APPLICATION GUIDE

FIELDCOMM GROUP™

Connecting the World of Process Automation

HART®

COMMUNICATION PROTOCOL

Using the HART Standard DD

FCG AG10101

Edition 2.1

27 Jan 2022

Document Distribution / Maintenance Control / Document Approval

To obtain information concerning document distribution control, maintenance control, and document approval please contact FieldComm Group at the address shown below.

Copyright © 2022 FieldComm Group

Trademark Information

FieldComm Group

Attention: President and CEO

9430 Research Boulevard

Suite 1-120

Austin, TX 78759, USA

Voice: (512) 792-2300

FAX: (512) 792-2310

http://www.fieldcommgroup.org

Intellectual Property Rights

The FieldComm Group (the Group) does not knowingly use or incorporate any information or data into the HART, FOUNDATION Fieldbus and FDI protocol standards, which the Group does not own or have lawful rights to use. Should the Group receive any notification regarding the existence of any conflicting private IPR, the Group 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 Group encourage implementers to infringe on any individual's or organization's IPR.

TABLE OF CONTENTS

1 Scope

This document describes the HART standard DDs, and how they can be used by DD developers.

2 Normative References

Specification	Document Number
HART Communication Protocol Specification	FCG TS20013
Universal Command Specification	FCG TS20127
Common Practice Command Specification	FCG TS20151
HART EDD Test Specification	FCG TS20019
EDDL Syntax and Semantics	FCG TS61804-3
EDD Interpretation	FCG TS61804-4
EDDL Builtin Library	FCG TS61804-5
HART Common Tables Specification	FCG TS20183
HART EDD Test Specification	FCG TT20019
Command Summary Specification	FCG TS20099

3 Definitions

Burst Mode A device operating mode which repeatedly sends the response to a selected HART command without the need for a request.

Device Revision The integer returned in byte 5 of Identity Commands (see the Command Summary Specification). This defines the revision level of the command set supported by the field device including the device-specific commands. The device revision returned by a HART device is used to select the DD to be used with that device.

Device Type The integer returned in bytes 1-2 of Identity Commands (see the Command Summary Specification). The device type returned by a HART device is used to select the DD to be used with that device.

Device Variable The process variables produced by the instrument or consumed by the instrument.

LIST A construct similar to a Value Array, except the number of elements in the list can change dynamically rather than being statically defined.

Manufacturer ID The integer returned in byte 17-18 of Identity Commands. The manufacturer ID returned by a HART device is used to select the DD to be used with that device.

Reference Array A construct that is built by referencing individually created elements, which can then be referenced by numeric index.

Value Array A construct that uses a single type definition, and the number of elements can be specified to automatically generate multiple instances of the same type. Each instance is then referenced by numeric index.

4 Abbreviations

ACK	Acknowledge
AO	Analog Output
BACK	Burst Acknowledge
COMM	Communication
CONFIG	Configuration
DD	Device Description
DDL	Device Description Language
DEV	Device
EDD	Electronic Device Description
EDDL	Electronic Device Description Language
FCG	FieldComm Group
FDI	Field Device Integration
HART	Highway Addressable Remote Transducer
ID	Identifier
IDE	Integrated Development Environment
IO	Input/Output
LCD	Liquid Crystal Display
MISC	Miscellaneous
MULTI	Multiple
PV	Primary Variable
QV	Quaternary Variable
REF	Reference
SI Units	International System of Units
STATS	Statistics

STX	Start of Transaction
SV	Secondary Variable
TV	Tertiary Variable
VAL	Value
VAR	Variable
VARS	Variables

5 Introduction

FCG creates and maintains HART standard DDs that can be used by DD developers. HART standard DDs provide standardized implementations of HART commands and features that can be imported into device vendors' DD implementations. HART standard DDs promote uniformity in the implementation of HART communications between multiple vendors and multiple devices. The HART standard DDs are modelled according to the HART Protocol specifications. Refer to Table 1 for the list of available HART standard DDs.

Table 1: Mapping of Standard DDs to HART Specification

DD Name	HART Specification
Standard DD Reference	Common Tables Specification (FCG TS20183)
Universal	Universal Command Specification (FCG TS20127)
Analog Channels	Common Practice Command Specification (FCG TS20151)
Burst Mode – Reference Arrays	Common Practice Command Specification (FCG TS20151)
Burst Mode - Value Arrays	Common Practice Command Specification (FCG TX20151)
Condensed Status	Common Practice Command Specification (FCG TS20151)
Device Variables	Common Practice Command Specification (FCG TS20151)
IO Systems List	Common Practice Command Specification (FCG TS20151)
IO Systems Reference Arrays	Common Practice Command Specification (FCG TS20151)
Miscellaneous	Common Practice Command Specification (FCG TS20151)
Wireless	Wireless Command Specification (FCG TS20155)
PV1	Universal Command Specification (FCG TS20127)
PV2	Universal Command Specification (FCG TS20127)

6 Guidelines for Using the Standard DD

Whether creating a new DD or updating an existing DD, the best practice is to import the latest revisions of applicable HART standard DDs into a product DD. In addition to making DDs consistent and conformant to the HART Communication standards, importing the HART standard DDs provides a jump start to the DD development process. After importing applicable HART standard DDs, the developer will already have most of the standardized command and variable definitions included in their product DD.

The current revisions of HART standard DDs are provided in Table 2, and are the revisions that should be imported into vendor DDs when they are applicable. The DEVICE_TYPE, DEVICE_REVISION, and DD_REVISION in the table correspond to the values that are necessary to include in an IMPORT statement whenever those DDs are imported. Example DD source snippets that shows the import of each of these Standard DDs is included in section 7.

When importing the HART standard DDs into a device DD, each standard DD can only be imported 1 time. For each of the Standard DDs that are imported into a device DD, the import declarations should appear in DD source code in the same order as is listed in Table 2.

Table 2: HART Standard DD Revisions

DEVICE_TYPE	DD Revision for HART 5 (Cmn Prac 7)	DD Revision for HART 6 (Cmn Prac 8)	DD Revision for HART 7.0-7.3 (Cmn Prac 9)	DD Revision for HART 7.4-7.6 (Cmn Prac10 or Cmn Prac 11)	DD Revision for HART 7.7 (Cmn Prac 12)
_TABLES	DEV_REV **25** DD_REV **2**				
_UNIVERSAL	DEV_REV **5** DD_REV **4**	DEV_REV **6** DD_REV **4**	DEV_REV **7** DD_REV **5**		
_PV (for Single Analog Channels)	DEV_REV **1** DD_REV **3**				
_PV (for multiple Analog Channels)	DEV_REV **2** DD_REV **3**			DEV_REV **2** DD_REV **4**	
_COMMON_PRACTICE_DEVICE_VARIABLES	DEV_REV **7** DD_REV **6**	DEV_REV **8** DD_REV **4**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	Dev Rev 12 DD Rev **2**
_COMMON_PRACTICE_ANALOG_CHANNELS	DEV_REV **7** DD_REV **6**	DEV_REV **8** DD_REV **4**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	Dev Rev 12 DD_REV **2**
_COMMON_PRACTICE_BURST_VAL_ARRAY	DEV_REV **7** DD_REV **6**	DEV_REV **8** DD_REV **4**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	Dev Rev 12 DD Rev **2**
_COMMON_PRACTICE_BURST_REF_ARRAY	DEV_REV **7** DD_REV **6**	DEV_REV **8** DD_REV **4**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	Dev Rev 12 DD Rev **2**
_COMMON_PRACTICE_CONDENSED_STATUS	**N/A**			DEV_REV **11** DD_REV **2**	
_COMMON_PRACTICE_IO_SYSTEM_LIST	**N/A**	**N/A**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	
_COMMON_PRACTICE_IO_SYSTEM_REF_ARRAY	**N/A**	**N/A**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	
_COMMON_PRACTICE_MISC	DEV_REV **7** DD_REV **6**	DEV_REV **8** DD_REV **4**	DEV_REV **9** DD_REV **4**	DEV_REV **11** DD_REV **2**	Dev Rev 12 DD Rev **2**
_WIRELESS	**N/A**		DEV_REV **1** DD_REV **4**	Dev Rev **2** DD Rev **2**	

The list of HART standard DDs necessary to import into a product DD is determined by a variety of conditions. The following sections are useful to describe which HART Standard DDs to import, and how to import the right set of items from them. Additionally, several sample DDs are available as a reference to provide example implementations of potential product DDs. These samples illustrate the proper way to import the HART standard DDs. The sample DDs are installed as part of the FDI IDE into the \HCF\DDL\dev folder by default. Please refer to one or more of the following sample DDs provided to illustrate the implementation of DDs for various classes of devices:

- Sample_HART5.ddl
- Sample_HART6.ddl
- Sample_HART7.ddl
- Sample_Burst_Ref_Array.ddl

- Sample_Burst_Val_Array.ddl
- Sample_Condensed_Status.ddl
- Sample_Fixed_Mapping.ddl
- Sample_IO_Adapter_List.ddl
- Sample_IO_Adapter_Ref_Array.ddl
- Sample_Multi_AO.ddl
- Sample_Wireless.ddl

The subsections of section 6 provide information to determine which HART Standard DDs are necessary to be included in a vendor's device DD. Use the descriptions in section 6 to determine what is needed for the device DD based on the implemented features of the associated device being modelled.

6.1 Fundamentals

All HART Device DDs must import the fundamental items to support universal HART communications and features. The necessary fundamentals include the _TABLES, _UNIVERSAL, and _PV DDs.

6.1.1 _TABLES

The Standard Tables DD provides standardized enumerations that are necessary to support all the other HART Standard DDs.

At a minimum, the IMPORT_TABLES_BASE() macro (see 7.13.1.1), IMPORT_TABLES_DEVELOPER() macro (see 7.13.1.5), and IMPORT_TABLES_HART(u) macro (see 7.13.1.11) must be included for all HART Device DDs. Additional items may also be required for import from the Standard Tables DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _TABLES to import.

There are a limited set of redefinitions of Standard Table items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of Standard Tables items are described in 7.13.3. Any redefinitions of items that are not described in 7.13.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _TABLES DD can be found in section 7.13.

6.1.2 _UNIVERSAL

All HART Device DDs must import one of the revisions of _UNIVERSAL DD. The device revision of the Universal DD imported is determined by which HART Universal revision the field device implements (5, 6, or 7).

At a minimum, the IMPORT_UNIVERSAL_BASE(u) macro must be included for all HART Device DDs (see 7.1.1.1). When implementing a DD with Universal revision 7, the IMPORT_UNIVERSAL_DEVICE_STATUS(n) macro (see 7.1.1.2) must be included in the universal import section, and IMPORT_TABLES_STANDARD_STATUS(u, n) macro (see 7.13.1.22) must be included in the tables import section. Additional items may also be required for import from the Universal DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _UNIVERSAL to import that corresponds to the HART universal revision being used.

There are a limited set of redefinitions of Universal items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of Universal items are described in 7.1.3. Any redefinitions of items that are not described in 7.1.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

For HART 5 or HART 6 devices, if commands 38 and/or 48 are supported, these commands are imported manually from the Universal DD (See 7.1.1.5).

Detailed reference material for the _UNIVERSAL DD can be found in section 7.1.

6.1.3 _PV

All HART Device DDs must import one of the revisions of _PV DD. PV device revision 1 is used to support devices with only a single analog output channel. PV device revision 2 is used to support devices with multiple analog output channels. At a minimum, the IMPORT_PV1_BASE(d), IMPORT_PV2_3_BASE(), or IMPORT_PV2_4_BASE() macro must be included for all HART Device DDs (see 7.2.1.1, 7.3.1.1, and 7.3.1.2). Refer to Table 2 to determine the latest DD revision of _PV to import that corresponds to the _PV device revision being used.

There are a limited set of redefinitions of PV items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of PV1 items are described in 7.2.3. Generally accepted redefinitions of PV2 items are described in 7.3.3. Any redefinitions of items that are not described in 7.2.3 or 7.3.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

The _PV DD contains additional infrequently used items that are also available to import individually. Reference 7.2.1.2, and 7.3.1.3 for a list of these items.

Detailed reference material for the _PV1 DD can be found in section 7.2. Detailed reference material for the _PV2 DD can be found in section 7.3.

6.2 Device Variables

HART device DDs that model devices that support more than just the 4 HART Dynamic Variables PV, SV, TV, and QV must import the _COMMON_PRACTICE_DEVICE_VARIABLES DD. At a minimum, the IMPORT_DEV_VARS_BASE() macro must be included (see 7.4.1.1). Additional items may also be required for import from the Common Practice Device Variables DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_DEVICE_VARIABLES to import.

There are a limited set of redefinitions of Common Practice Device Variables items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of device variable items are described in 7.4.3. Any redefinitions of items that are not described in 7.4.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

The _COMMON_PRACTICE_DEVICE_VARIABLES DD contains additional infrequently used items that are also available to import individually. Reference 7.4.1.7 for a list of these items.

Detailed reference material for the _COMMON_PRACTICE_DEVICE_VARIABLES DD can be found in section 7.4.

6.2.1 Device Variable Mapping

Device DDs that model devices that support the ability to read device variable mapping can include the IMPORT_DEV_VARS_MAPPING_READ() macro within the _COMMON_PRACTICE_DEVICE_VARIABLES_DD import section (see 7.4.1.4).

Device DDs that model devices that support the ability to read and remap device variable mapping can include the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro within the _COMMON_PRACTICE_DEVICE_VARIABLES_DD import section (see 7.4.1.5).

6.2.2 Device Variable Trim

HART 6 or later device DDs that model devices that support the ability to trim 1 or more device variables can include the IMPORT_DEV_VARS_TRIM() macro within the _COMMON_PRACTICE_DEVICE_VARIABLES_DD import section (see 7.4.1.6).

6.2.3 Variable Catch

HART 6 or later device DDs that model devices that support the ability to catch one or more device variables can include either the IMPORT_DEV_VARS_CATCH_V11() or IMPORT_DEV_VARS_CATCH_V8() macros within the _COMMON_PRACTICE_DEVICE_VARIABLES_DD import section (see 7.4.1.2 and 7.4.1.3). The older macro is used for the original implementation of device variable catching that supports 1 byte manufacturer code, device type, and command number. The latest version of variable catching supports the expanded device type and 2-byte command numbers.

6.3 Analog Channels

HART device DDs that model at least 1 analog output channel must import the _COMMON_PRACTICE_ANALOG_CHANNELS DD. Implementations of configuration and control of analog output channels may vary from one device to the next, so the items listed in 7.5.1.2 should be imported individually to match the set of commands implemented in the corresponding field device. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_ANALOG_CHANNELS to import.

There are a limited set of redefinitions of Common Practice Device Variables items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of analog channel items are described in 7.5.3. Any redefinitions of items that are not described in 7.5.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_DEVICE_VARIABLES DD can be found in section 7.5.

6.3.1 Multiple Analog Output Channels

Device DDs that model devices that support multiple analog outputs must include the IMPORT_ANALOG_CHANNELS_MULTI_AO() macro (see 7.5.1.1) within the _COMMON_PRACTICE_ANALOG_CHANNELS DD import section and IMPORT_TABLES_MULTI_AO() macro (see 7.13.1.17) in the _TABLES import section. Device DDs that model devices that support multiple analog outputs will also need to import either DD rev 1 or 2 of the PV2 DD (see 7.3).

6.4 Burst Operation

HART device DDs that model burst mode operation must include one of the standard burst DDs. There are 2 different variations of the standard burst DDs to choose from. One variation implements up to 4 individual burst messages and up to 2 individual event notifications that are collected into reference arrays. Reference arrays are supported by older host applications dating back several decades. The second variation implements burst messages and event notifications as a value array. Value arrays can be configured to have any number of instances, which allow support of as many burst messages and event notifications as necessary by simply supplying the number of instances desired. Value arrays were not originally supported in host applications over a decade ago.

When choosing whether to import the reference array model or the value array model, the following considerations should be made:

- New DDs that are only required to function on latest host applications (e.g. FDI compliant applications) or are going to be used with newer host applications that were designed in the last 10 years should import the value array model. The value array model is preferred because it is easily scalable to future revisions of the DD/device which may add additional burst messages or event notifications over time.
- Existing DDs that have already been implemented with a reference array burst message model may want to be designed to import the reference array model in order to maintain compatibility of transferring saved device configurations made with existing DDs to data sets created with an updated DD.

6.4.1 Burst using Value Array

HART device DDs that choose to import the burst message model using value arrays must import the _COMMON_PRACTICE_BURST_VAL_ARRAY DD. At a minimum, the IMPORT_BURST_VAL_ARRAY_BASE(u) macro

(see 7.7.1.1) must be imported in the burst value array import section, and IMPORT_TABLES_BURST_VAL_ARRAY(u) macro (see 7.13.1.3) must be imported in the tables import section. Additional items may also be required for import from the Common Practice Burst Value Array DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_BURST_VAL_ARRAY to import.

There are a limited set of redefinitions of Common Practice Burst items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of burst items are described in 7.7.3. Any redefinitions of items that are not described in 7.7.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_BURST_VAL_ARRAY DD can be found in section 7.7.

6.4.1.1 Event Notification

HART 7 or later device DDs using the burst value array model, that model devices supporting event notification must include the IMPORT_BURST_VAL_ARRAY_EVENT() macro (see 7.7.1.2) within the _COMMON_PRACTICE_BURST_VAL_ARRAY DD import section and IMPORT_TABLES_EVENT_VAL_ARRAY() macro (see 7.13.1.10) in the _TABLES import section.

6.4.1.2 Device Variable Trending

HART 7 or later device DDs using the burst value array model, that model devices supporting device variable trending must include the IMPORT_BURST_VAL_ARRAY_TREND() macro (see 7.7.1.5) within the _COMMON_PRACTICE_BURST_VAL_ARRAY DD import section and IMPORT_TABLES_TREND_VAL_ARRAY() macro (see 7.13.1.24) in the _TABLES import section.

6.4.1.3 Event Manager Registration

HART 7 or later device DDs using the burst value array model, that model devices supporting event manager registration must include the IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER() macro (see 7.7.1.3) within the _COMMON_PRACTICE_BURST_VAL_ARRAY DD import section and IMPORT_TABLES_EVENT_MANAGER() macro (see 7.13.1.8) in the _TABLES import section.

6.4.2 Burst using Reference Array

HART device DDs that choose to import the burst message model using reference arrays must import the _COMMON_PRACTICE_BURST_REF_ARRAY DD. At a minimum, the IMPORT_BURST_REF_ARRAY_BASE(u, b) macro (see 7.8.1.1) must be imported in the burst reference array import section, and IMPORT_TABLES_BURST_REF_ARRAY(u, b) macro (see 7.13.1.2) must be imported in the tables import section. Additional items may also be required for import from the Common Practice Burst Reference Array DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_BURST_REF_ARRAY to import.

There are a limited set of redefinitions of Common Practice Burst items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of burst items are described in 7.8.3. Any redefinitions of items that are not described in 7.8.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_BURST_REF_ARRAY DD can be found in section 7.8.

6.4.2.1 Event Notification

HART 7 or later device DDs using the burst reference array model, that model devices supporting event notification must include the IMPORT_BURST_REF_ARRAY_EVENT(e) macro (see 7.8.1.2) within the _COMMON_PRACTICE_BURST_REF_ARRAY DD import section and IMPORT_TABLES_EVENT_REF_ARRAY(e) macro (see 7.13.1.9) in the _TABLES import section.

6.4.2.2 Device Variable Trending

HART 7 or later device DDs using the burst reference array model, that model devices supporting device variable trending must include the IMPORT_BURST_REF_ARRAY_TREND() macro (see 7.8.1.6) within the _COMMON_PRACTICE_BURST_REF_ARRAY DD import section and IMPORT_TABLES_TREND_REF_ARRAY() macro (see 7.13.1.23) in the _TABLES import section.

6.4.2.3 Event Manager Registration

HART 7 or later device DDs using the burst reference array model, that model devices supporting event manager registration must include the IMPORT_BURST_REF_ARRAY_EVENT_REGISTER() macro (see 7.8.1.3) within the _COMMON_PRACTICE_BURST_VAL_ARRAY DD import section and IMPORT_TABLES_EVENT_MANAGER() macro (see 7.13.1.8) in the _TABLES import section.

6.5 Condensed Status

HART 7 or later device DDs that model devices that support condensed status commands can import the _COMMON_PRACTICE_CONDENSED_STATUS DD. At a minimum, the IMPORT_CONDENSED_STATUS_BASE() macro (see 7.6.1.1) must be included in the condensed status import section and IMPORT_TABLES_CONDENSED_STATUS(n) macro (see 7.13.1.4) must be included in the tables import section. Additional items may also be required for import from the Common Practice Device Variables DD and will be noted in each of the cases where applicable. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_CONDENSED_STATUS to import.

There are a limited set of redefinitions of Condensed Status items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of condensed status items are described in 7.6.3. Any redefinitions of items that are not described in 7.6.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_CONDENSED_STATUS DD can be found in section 7.6.

Even for devices that don't support the condensed status HART commands, a DD can include a getHealthStatus() method in order to provide condensed status capabilities via the DD. An example implementation of this method is available for review in example DDs (see 6).

6.5.1 Status Simulation Method

A reference implementation of a method to support status simulation is available as an option for DD developers to use. To import this reference method, include the IMPORT_CONDENSED_STATUS_METHOD_SIMULATE() macro within the _COMMON_PRACTICE_CONDENSED_STATUS DD import section (see 7.6.1.2).

6.6 Miscellaneous Features

There are several small miscellaneous features implement in the HART standard DD that does not fall into broader categories. The implementation of these features is included in a miscellaneous import DD. HART device DDs that implement one or more of these miscellaneous features can import the _COMMON_PRACTICE_MISC DD. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_MISC to import.

There are a limited set of redefinitions of miscellaneous items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of miscellaneous items are described in 7.11.3. Any redefinitions of items that are not described in 7.11.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

The _COMMON_PRACTICE_MISC DD contains additional infrequently used items that are also available to import individually. Reference 7.11.1.9 for a list of these items.

Detailed reference material for the _COMMON_PRACTICE_MISC DD can be found in section 7.11.

6.6.1 Lock

HART 6 or later device DDs that model devices supporting the HART device configuration lock can include the IMPORT_MISC_LOCK() macro (see 7.11.1.3) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_LOCK() macro (see 7.13.1.16) in the _TABLES import section.

6.6.2 Real-Time Clock

HART 7 or later device DDs that model devices supporting a read-only real-time clock can include the IMPORT_MISC_READ_TIME() macro (see 7.11.1.5) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_READ_TIME() macro (see 7.13.1.18) in the _TABLES import section.

HART device DDs that model devices supporting a readable and settable real-time clock can include the IMPORT_MISC_READ_WRITE_TIME() macro (see 7.11.1.6) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_READ_WRITE_TIME() macro (see 7.13.1.19) in the _TABLES import section.

6.6.3 Squawk

HART 7 or later device DDs that model devices supporting the HART Squawk mechanism can include the IMPORT_MISC_SQUAWK() macro (see 7.11.1.8) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_SQUAWK() macro (see 7.13.1.21) in the _TABLES import section.

6.6.4 Location

HART 7 or later device DDs that model devices supporting device location discovery can include the IMPORT_MISC_LOCATION() macro (see 7.11.1.2) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_LOCATION() macro (see 7.13.1.15) in the _TABLES import section.

6.6.5 SI Units

HART 7 or later device DDs that model devices supporting SI Unit Control can include the IMPORT_MISC_SI_UNIT() macro (see 7.11.1.7) within the _COMMON_PRACTICE_MISC DD import section and IMPORT_TABLES_SI_UNIT() macro (see 7.13.1.20) in the _TABLES import section.

6.6.6 Communication Statistics

HART 7 or later device DDs that model devices supporting monitoring of communication statistics can include the IMPORT_MISC_COMM_STATS() macro (see 7.11.1.1) within the _COMMON_PRACTICE_MISC DD import section.

6.6.7 Process Unit Tagging

HART device DDs that model devices supporting process unit tagging can include the IMPORT_MISC_PROCESS_UNIT_TAG() macro (see 7.11.1.4) within the _COMMON_PRACTICE_MISC DD import section.

6.7 IO Systems

HART 7 or later device DDs that model IO Systems with subdevices must include one of the standard IO System DDs. There are 2 different variations of the standard IO System DDs to choose from. One variation is implemented using reference arrays. The reference array model requires the execution of a METHOD to discover the list of subdevices that are attached to the IO System. Reference arrays are supported by older host applications dating back several decades. The second variation is implemented using a LIST. LISTs are arrays that can dynamically change size, which allow support of keeping track of subdevice lists dynamically during runtime as the number of subdevices attached to the IO System change over time. LISTs were not originally supported in host applications over a decade ago.

When choosing whether to import the reference array model or the LIST model, the following considerations should be made:

- New DDs that are only required to function on latest host applications (e.g. FDI compliant applications) or are going to be used with newer host applications that were designed in the last 10 years should import the LIST model. The LIST model is preferred because it is easily scalable to support any number of dynamically changing sets of subdevices.
- Existing DDs that have already been implemented with a reference array model may want to be designed to import the reference array model in order to maintain compatibility of transferring saved device configurations made with existing DDs to data sets created with an updated DD.

6.7.1 IO Systems using LIST

HART 7 or later device DDs that choose to import the IO System model using LISTs must import the _COMMON_PRACTICE_IO_SYSTEM_LIST DD. At a minimum, the IMPORT_IO_LIST_BASE_V11() or IMPORT_IO_LIST_BASE_V9() macro (see 7.9.1.2 and 7.9.1.3) must be imported in the IO system list import section. IMPORT_TABLES_IO_ADAPTER_V11() or IMPORT_TABLES_IO_ADAPTER_V9 macro (see 7.13.1.12 and 7.13.1.13) must be imported in the tables import section. IMPORT_BURST_VAL_ARRAY_SUBDEVICE() macro (see 7.7.1.4) must be imported into the burst import section. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_IO_SYSTEM_LIST to import. The xxx_V9 macros are used for the original implementation of IO System commands as they were defined in HART Common Practice Specification Version 9. The xxx_V11 macros are used for updated HART Common Practice version 10 or 11.

There are a limited set of redefinitions of IO System items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of IO System items are described in 7.9.3. Any redefinitions of items that are not described in 7.9.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_IO_SYSTEM_LIST DD can be found in section 7.9.

6.7.1.1 Subdevice Communication Statistics

HART 7 or later device DDs that model devices supporting the monitoring of subdevice communication statistics can include the IMPORT_IO_LIST_SYSTEM_COMM_STATS() macro (see 7.9.1.4) within the _COMMON_PRACTICE_IO_SYSTEM_LIST DD import section and IMPORT_TABLES_IO_ASSIGN() macro (see 7.13.1.14) in the _TABLES import section.

6.7.1.2 Subdevice List Assignment

HART 7 or later device DDs that model devices supporting subdevice list assignment can include the IMPORT_IO_LIST_ASSIGN() macro (see 7.9.1.1) within the _COMMON_PRACTICE_IO_SYSTEM_LIST DD import section.

6.7.2 IO Systems using Reference Array

HART 7 or later device DDs that choose to import the IO System model using reference arrays must import the _COMMON_PRACTICE_IO_SYSTEM_REF_ARRAY DD. At a minimum, the IMPORT_IO_REF_ARRAY_BASE_V11() or IMPORT_IO_REF_ARRAY_BASE_V9() macro (see 7.10.1.2 and 7.10.1.3) must be imported in the IO system reference array import section. IMPORT_TABLES_IO_ADAPTER_V11() or IMPORT_TABLES_IO_ADAPTER_V9() macro (see 7.13.1.12 and 7.13.1.13) must be imported in the tables import section. IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(b) macro (see 7.8.1.4) and IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) macro (see 7.8.1.5) must be imported into the burst import section. Refer to Table 2 to determine the latest DD revision of _COMMON_PRACTICE_REF_ARRAY_SYSTEM_LIST to import. The xxx_V9 macros are used for the original implementation of IO System commands as they were defined in HART Common Practice Specification Version 9. The xxx_V11 macros are used for updated HART Common Practice version 10 or 11.

There are a limited set of redefinitions of IO System items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of IO System items are described in 7.10.3. Any

redefinitions of items that are not described in 7.10.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

Detailed reference material for the _COMMON_PRACTICE_IO_SYSTEM_LIST DD can be found in section 7.10.

6.7.2.1 Subdevice Communication Statistics

HART 7 or later device DDs that model devices supporting the monitoring of subdevice communication statistics can include the IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS() macro (see 7.10.1.4) within the _COMMON_PRACTICE_IO_SYSTEM_REF_ARRAY DD import section and IMPORT_TABLES_IO_ASSIGN() macro (see 7.13.1.14) in the _TABLES import section.

6.7.2.2 Subdevice List Assignment

HART 7 or later device DDs that model devices supporting subdevice list assignment can include the IMPORT_IO_REF_ARRAY_ASSIGN() macro (see 7.10.1.1) within the _COMMON_PRACTICE_IO_SYSTEM_LIST DD import section.

6.8 Wireless

HART 7 or later device DDs that model devices that support wirelessHART must import the _WIRELESS DD. At a minimum, the IMPORT_WIRELESS_BASE() macro (see 7.12.1.1) must be included in the condensed status import section and IMPORT_TABLES_WIRELESS() macro (see 7.13.1.25) must be included in the tables import section. Refer to Table 2 to determine the latest DD revision of _WIRELESS to import.

There are a limited set of redefinitions of wireless items that may need to be redefined to match the implementation of a field device. Generally accepted redefinitions of wireless items are described in 7.12.3. Any redefinitions of items that are not described in 7.12.3 must be discussed with FCG to make sure they are compliant to HART protocol standards.

The _WIRELESS DD contains additional infrequently used items that are also available to import individually. Reference 7.12.1.2 for a list of these items.

Detailed reference material for the _WIRELESS DD can be found in section 7.12.

6.9 Device Variables Array

In addition to importing the HART standard DDs, an array of device variables needs to be defined by the DD developer (outside of importing the standard DDs). These items are required to be provided since they are referenced (but not defined) in the HART standard DDs.

HART DDs that model a field device need to provide a collection of attributes that are associated with each device variable that the field device supports. The attributes that must be provided for each device variable depends on the capabilities that the device supports for each variable. Refer to Table 3 for a list of attributes. The attributes are applicable and need to be defined by the DD developer whenever the associated command numbers shown in the table are imported into the DD.

Starting with this version of the HART Standard DDs, a new attribute “CONFIG_UNITS” has been added to the collection. The CONFIG_UNITS is used to allow the device variable units to be configured but remain separated from the DIGITAL_UNITS that are associated with the reported DIGITAL_VALUE. The attribute associated with DIGITAL_UNITS should be declared as read-only. If the CONFIG_UNITS are configured and saved to a field device, that field device will eventually report an updated value in the DIGITAL_UNITS. Depending on the communication topology, caching components present in the system and system update rates, it can take some time after configuring different unit codes before a new dynamic value in the new units is reported. Separating CONFIG_UNITS from DIGITAL_UNITS will prevent invalid unit codes from being associated with dynamic values. The DD developer should use CONFIG_UNITS in unit relations for all static configuration attributes associated with

those units. Use DIGITAL_UNITS in unit relations for all dynamic attributes associated with those units. An example is illustrated in Figure 3.

Table 3: Device Variable Attributes

Attribute	Data Type	Referenced by Standard DD	Associated Command Numbers
CAPTURE_MODE	*LIKE capture_mode_codes*	*Common Device Variables (HART 6 or later)*	113, 114
CLASSIFICATION	*LIKE* device_variable_classification_codes	*Universal DD (HART 6 or later)* *Common Device Variables*	8, 9 54, 522
CONFIG_UNITS	*LIKE units_code_xxxx*	*Universal DD* *Common Device Variables*	14 44, 53, 54
DAMPING_VALUE	*FLOAT*	*Universal DD* *Common Device Variables*	15 34, 54, 55
DATA_QUALITY	*LIKE process_data_status*	*Universal DD (HART 6 or later)* *Common Device Variables*	9 79
DEVICE_FAMILY	*LIKE device_variable_family_codes*	*Common Device Variables (HART 6 or later)*	54
DEVICE_FAMILY_STATUS	*LIKE device_family_status*	*Universal DD (HART 6 or later)* *Common Device Variables*	9 79
DIGITAL_UNITS	*LIKE units_code_xxxx*	*Universal DD* *Common Device Variables*	1, 3, 9 33, 79, 61, 110
DIGITAL_VALUE	*FLOAT*	*Universal DD* *Common Device Variables*	1, 3, 9 33, 61, 79, 82, 110
LIMIT_STATUS	*LIKE limit_status*	*Universal DD (HART 6 or later)* *Common Device Variables*	9 79
LOWER_SENSOR_LIMIT	*FLOAT*	*Universal DD* *Common Device Variables*	14 54
LOWER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	80
MAXIMUM_LOWER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	81
MAXIMUM_UPPER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	81
MINIMUM_LOWER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	81
MINIMUM_SPAN	*FLOAT*	*Universal DD* *Common Device Variables*	14 54
MINIMUM_TRIM_DIFFERENTIAL	*FLOAT*	*Common Device Variables (HART 6 or later)*	81

Attribute	Data Type	Referenced by Standard DD	Associated Command Numbers
MINIMUM_UPPER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	81
PROPERTIES	*LIKE dev_var_property_codes*	*Common Device Variables (HART 7, Common 11 or later)*	54
SENSOR_SERIAL_NUMBER	*UNSIGNED_INTEGER (3)*	*Universal DD* *Common Device Variables*	14 49, 54, 56
SHED_TIME	*FLOAT*	*Common Device Variables (HART 6 or later)*	113, 114
SIMULATED	*LIKE write_device_variable_codes*	*Common Device Variables (HART 6 or later)*	79, 534
SIMULATED_DATA_QUALITY	*LIKE process_data_status*	*Common Device Variables (rev 12.0 or later)*	534
SIMULATED_DEVICE_FAMILY_STATUS	*LIKE device_variable_family_codes*	*Common Device Variables (rev 12.0 or later)*	534
SIMULATED_DIGITAL_VALUE	*FLOAT*	*Common Device Variables (rev 12.0 or later)*	534
SIMULATED_LIMIT_STATUS	*LIKE limit_status*	*Common Device Variables (rev 12.0 or later)*	534
SOURCE_COMMAND_NUMBER	*UNSIGNED_INTEGER (1) (original)* *UNSIGNED_INTEGER (2) (Starting with Common 11)*	*Common Device Variables (HART 6 or later)*	113, 114
SOURCE_DEVICE_ID	*UNSIGNED_INTEGER (3)*	*Common Device Variables (HART 6 or later)*	113, 114
SOURCE_DEVICE_TYPE	*LIKE device_type_code*	*Common Device Variables (HART 6 or later)*	113, 114
SOURCE_MANUFACTURER	*LIKE manufacturer_id (ended with Common 11))*	*Common Device Variables (HART 6 or later)*	113, 114
SOURCE_SLOT_NUMBER	*UNSIGNED_INTEGER*	*Common Device Variables (HART 6 or later)*	113, 114
TRIM_POINT_SUPPORT	*LIKE trim_point_codes*	*Common Device Variables (HART 6 or later)*	81, 82
TRIM_POINT_UNITS	*LIKE units_code_xxxx*	*Common Device Variables (HART 6 or later)*	80, 81, 82
UPPER_SENSOR_LIMIT	*FLOAT*	*Universal DD* *Common Device Variables*	14 54
UPDATE_TIME_PERIOD	*TIME_VALUE*	*Common Device Variables (HART 7 or later)*	54
UPPER_TRIM_POINT	*FLOAT*	*Common Device Variables (HART 6 or later)*	80

A collection for each device variable supported by the device should be defined in the DD similar to that shown below. Collection member names are the names in the “Attribute” column of Table 3. If the device variable is not supported by any of the commands referenced in the “Associated Command Numbers” column, then that collection member need not be included in the collection. At a minimum, all member names that are referenced by the HART Universal standard DD must be included in the collection (see 7.1.2).

```
COLLECTION OF VARIABLE var0
{
    LABEL "Variable 0";
    HELP "Help for variable 0";
    MEMBERS
    {
        SENSOR_SERIAL_NUMBER,       var0_sn,         "Variable 0 Serial Number";
        CLASSIFICATION,             var0_class,      "Variable 0 Classification";
        DEVICE_FAMILY,              var0_family,     "Variable 0 Device Family";
        PROPERTIES,                 var0_properties, "Variable 0 Properties";
        UPPER_SENSOR_LIMIT,         var0_usl,        "Variable 0 Upper Sensor Limit";
        LOWER_SENSOR_LIMIT,         var0_lsl,        "Variable 0 Lower Sensor Limit";
        MINIMUM_SPAN,               var0_min_span,   "Variable 0 Minimum Span";
        DAMPING_VALUE,              var0_damping,    "Variable 0 Damping";
        CONFIG_UNITS,               var0_config_unit,"Variable 0 Units";
        DIGITAL_UNITS,              var0_unit,       "Variable 0 Units";
        DIGITAL_VALUE,              var0_value,      "Variable 0 Value";
        DATA_QUALITY,               var0_pdq,        "Variable 0 Quality";
        LIMIT_STATUS,               var0_lim_status, "Variable 0 Limit Status";
        DEVICE_FAMILY_STATUS,       var0_fam_status, "Variable 0 Family Status";
        UPDATE_TIME_PERIOD,         var0_update_time,"Variable 0 Update Period";
        SIMULATED,                  var0_simulate,   "Variable 0 Simulated";
        TRIM_POINT_SUPPORT,         var0_trim_sup,   "Variable 0 Trim Support";
        MINIMUM_LOWER_TRIM_POINT,   var0_min_lo_trim,"Variable 0 Minimum Lower Trim";
        MAXIMUM_LOWER_TRIM_POINT,   var0_max_lo_trim,"Variable 0 Maximum Lower Trim";
        MINIMUM_UPPER_TRIM_POINT,   var0_min_hi_trim,"Variable 0 Minimum Upper Trim";
        MAXIMUM_UPPER_TRIM_POINT,   var0_max_hi_trim,"Variable 0 Maximum Upper Trim";
        MINIMUM_TRIM_DIFFERENTIAL,  var0_trim_diff,  "Variable 0 Min Trim Difference";
        TRIM_POINT_UNITS,           var0_trim_units, "Variable 0 Trim Units";
        LOWER_TRIM_POINT,           var0_low_trim,   "Variable 0 Lower Trim";
        UPPER_TRIM_POINT,           var0_high_trim,  "Variable 0 Upper Trim";
        CAPTURE_MODE,               var0_cap_mode,   "Variable 0 Capture Mode";
        SOURCE_DEVICE_TYPE,         var0_dev_type,   "Variable 0 Device Type";
        SOURCE_DEVICE_ID,           var0_dev_id,     "Variable 0 Device ID";
        SOURCE_SLOT_NUMBER,         var0_slot_num,   "Variable 0 Slot Number";
        SOURCE_COMMAND_NUMBER,      var0_cmd_num,    "Variable 0 Command Number";
        SHED_TIME,                  var0_shed_time,  "Variable 0 Shed Time";
    }
}
```

Once a collection has been defined for each device variable supported in the device, an array of those collections named “deviceVariables” needs to be defined. This array has integer index references that match the indexes for the device variables that are implemented in the associated device’s firmware. The definition of the deviceVariables array should look similar to the following:

```
ARRAY OF COLLECTION deviceVariables
{
    ELEMENTS
    {
        0, var0, "Variable 0";
        1, var1, "Variable 1";
        ...
    }
}
```

NOTE: The deviceVariables array is referenced from some of the commands in the HART standard DDs that do not necessarily have to support all of the device variables implemented in the device. The deviceVariables array should include ALL of the device variables implemented in the corresponding device (Not including the standardized PV, SV, TV, and QV codes (246-249)). If the device only allows a subset of the device variables from being trimmed, or mapped, then additional arrays of collections should be defined that hold the subset of applicable variables. Refer to 7.4.3.1 and 7.4.3.3 for more details.

6.10 Standard Dictionary

The HART Standard DD uses a standard dictionary to manage all strings used in the standard DDs for attributes such as LABELs, HELPs, Descriptions, and Static Text. For each defined string, the HART standard dictionary includes a translation for at least English, German, Japanese, Russian, Chinese, French, Portuguese, Italian, and Spanish languages. The HART standard dictionary file is installed in \HCF\DDL\Library\standard.dc8 by default.

If a DD developer wants to reference an existing HART standard string from the dictionary, it can be done by referencing the dictionary handle inside of square brackets. The following is an example of referencing the standard dictionary label string for engineering units:

```
VARIABLE myUnits
{
    LABEL [Units];

...

}
```

In order for some EDD applications to be able to resolve the intended text string, they may need to have the correct version of the HART Standard Dictionary installed into their system. The integration of the HART standard dictionary is separate from the integration of vendor's DDs. Starting with the version 10 (*.fma) tokenized DD binaries (distributed within FDI packages), the strings used from the dictionary are included within the DD binary file. This helps to ensure that the desired strings are available to the EDD application when using the DD. For DD files that are in the older formats (e.g. *.fm8), the standard dictionary strings have been made available using MACROs that allow the strings to also be embedded right within the binary, which eliminates the dependency on synchronizing standard dictionaries between device vendors and EDD application vendors. The MACROs for accessing direct HART standard item strings are declared in the embedded_strings.h file. This file is installed in \HCF\DDL\dev\embedded_strings.h by default.

To reference a direct HART standard string using the MACROs, make sure to #include the embedded_strings.h file in your DD. The macro names follow the convention of *dictionary_handle*_temp, where *dictionary handle* is the handle that would have normally been referenced within the square brackets []. The following is an example of using the MACRO used in place of the previous library reference to get the same Units string. By using the MACRO, the version 8 binary file will be forced to contain the direct string content, while the version 10 binary file will use the dictionary as normal (and the v10 tokenizer will pull the string out of the dictionary and place it into the binary file).

```
#include "embedded_strings.h"

VARIABLE myUnits
{
    LABEL Units_temp;

...
}
```

7 Standard DD Reference

7.1 HART Universal DD

The universal commands are the basic commands that all HART instruments are required to implement. Furthermore, they are required to be implemented as defined by the FCG group.

The HART Universal DD includes all variables and commands necessary to implement the Universal HART commands. The HART Universal binary files are installed in \HCF\DDL\Library\000000\0002\0705.*, 0604.*, and 0504.*. The corresponding DDL source code for these files is located in HCF\DDL\HCFinfo\Standards\000000\Univ7_5.ddl, Univ6_4.ddl, and Univ5_4.ddl. Only one of the HART Universal DDs (either 5, 6, or 7) should be imported into a device DD depending on which HART universal revision is being modelled. To import the universal DD into a product DD, use one of the following IMPORT statements:

```
IMPORT STANDARD _UNIVERSAL, DEVICE_REVISION 7, DD_REVISION 5
{
    IMPORT_UNIVERSAL_BASE(7)
}

IMPORT STANDARD _UNIVERSAL, DEVICE_REVISION 6, DD_REVISION 4
{
    IMPORT_UNIVERSAL_BASE(6)
}

IMPORT STANDARD _UNIVERSAL, DEVICE_REVISION 5, DD_REVISION 4
{
    IMPORT_UNIVERSAL_BASE(5)
}
```

Import macros have been provided to make it easier to extract the correct items from the HART Universal DD. A description of each of these macros is described in 7.1.1.

7.1.1 IMPORTS

7.1.1.1 IMPORT_UNIVERSAL_BASE(*u*)

The IMPORT_UNIVERSAL_BASE(u) macro imports a list of all universal command items required to be included in all DDs. Different sets of tables are necessary depending on the HART universal revision that the DD is modeling. The parameter ‘u’ is used to specify which UNIVERSAL revision items to import. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used.

To import these required items, add one of the following line within the Universal import section:

```
IMPORT_UNIVERSAL_BASE(7)
IMPORT_UNIVERSAL_BASE(6)
IMPORT_UNIVERSAL_BASE(5)
```

The list of items imported using the IMPORT_UNIVERSAL_BASE(u) macro is described in Table 4. The Universal Revisions column indicates which items are imported for each of the HART universal revision macros.

Table 4: IMPORT_UNIVERSAL_BASE(u) items

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*comm_status*	5, 6, 7	Data item referenced from ALL HART commands.
VARIABLE	*config_change_counter*	6, 7	Data item referenced from command 38.
VARIABLE	*date*	5, 6, 7	Data referenced from commands 13 and 18.
VARIABLE	*descriptor*	5, 6, 7	Data referenced from commands 13 and 18.

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*device_id*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73.
VARIABLE	*device_type*	5, 6	Data item referenced from commands 0, and 11. Note – for HART 7, this variable is imported from the Common Tables DD See 7.13.
VARIABLE	*device_variable_code_1*	6, 7	Data referenced from commands 9 and 33.
VARIABLE	*device_variable_code_2*	6, 7	Data referenced from command 9 and 33.
VARIABLE	*device_variable_code_3*	6, 7	Data referenced from command 9 and 33.
VARIABLE	*device_variable_code_4*	6, 7	Data referenced from command 9 and 33.
VARIABLE	*device_variable_code_5*	7	Data referenced from command 9.
VARIABLE	*device_variable_code_6*	7	Data referenced from command 9.
VARIABLE	*device_variable_code_7*	7	Data referenced from command 9.
VARIABLE	*device_variable_code_8*	7	Data referenced from command 9.
VARIABLE	*final_assembly_number*	5, 6, 7	Data referenced from commands 16 and 19.
VARIABLE	*hardware_revision*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73.
VARIABLE	*longTag*	6, 7	Data referenced from commands 20, 21, and 22.
VARIABLE	*manufacturer_id*	5, 6	Data item referenced from commands 0, and 11, uses company_identification_code local variable for the enumeration list. Note – for HART 7, this variable is imported from the Common Tables DD See 7.13
VARIABLE	*max_num_device_variables*	6, 7	Data referenced from commands 0, 11, 21, and 73.
VARIABLE	*message*	5, 6, 7	Data referenced from commands 12 and 17.
VARIABLE	*polling_address*	5, 6, 7	Data referenced from commands 6 and 7.
VARIABLE	*private_label_distributor*	5, 6	Data item referenced from commands 0, and 11. Note – for HART 7, this variable is imported from the Common Tables DD See 7.13.
COMMAND	*read_additional_device_status*	7	Command Definition for command 48.
COMMAND	*read_device_variables_and_status*	6, 7	Command Definition for command 9.
COMMAND	*read_dynamic_variable_classification*	6, 7	Command Definition for command 8.
COMMAND	*read_dynamic_variables_and_pv_current*	5, 6, 7	Command Definition for command 3.
COMMAND	*read_final_assembly_number*	5, 6, 7	Command Definition for command 16.
COMMAND	*read_long_tag*	6, 7	Command Definition for command 20.
COMMAND	*read_loop_configuration*	6, 7	Command Definition for command 7.
COMMAND	*read_message*	5, 6, 7	Command Definition for command 12.
COMMAND	*read_pv*	5, 6, 7	Command Definition for command 1.
COMMAND	*read_pv_current_and_percent_range*	5, 6, 7	Command Definition for command 2.

Item Type	Item Name	Universal Revisions	Description
COMMAND	*read_pv_output_info*	5, 6, 7	Command Definition for command 15.
COMMAND	*read_pv_sensor_info*	5, 6, 7	Command Definition for command 14.
COMMAND	*read_tag_descriptor_date*	5, 6, 7	Command Definition for command 13.
COMMAND	*read_unique_identifier*	5, 6, 7	Command Definition for command 0.
COMMAND	*read_unique_identifier_with_long_tag*	6, 7	Command Definition for command 21.
COMMAND	*read_unique_identifier_with_tag*	5, 6, 7	Command Definition for command 11.
VARIABLE	*request_preambles*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73 .
COMMAND	*reset_configuration_change_flag*	7	Command Definition for command 38.
VARIABLE	*response_code*	5, 6, 7	Data referenced from ALL HART commands.
VARIABLE	*response_preambles*	6, 7	Data referenced from commands 0, 11, 21, 59, and 73.
VARIABLE	*software_revision*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73.
VARIABLE	*tag*	5, 6, 7	Data referenced from commands 11, 13, and 18.
VARIABLE	*time_stamp*	7	Data referenced from command 9.
VARIABLE	*transmitter_revision*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73.
VARIABLE	*universal_revision*	5, 6, 7	Data referenced from commands 0, 11, 21, and 73.
COMMAND	*write_final_assembly_number*	5, 6, 7	Command Definition for command 19.
COMMAND	*write_long_tag*	6, 7	Command Definition for command 22.
COMMAND	*write_message*	5, 6, 7	Command Definition for command17.
COMMAND	*write_polling_address*	5, 6, 7	Command Definition for command 6.
COMMAND	*write_tag_descriptor_date*	5, 6, 7	Command Definition for command 18.

7.1.1.2 IMPORT_UNIVERSAL_DEVICE_STATUS(*n*)

The IMPORT_UNIVERSAL_DEVICE_STATUS(n) macro imports a list of all items required for implementation of command 48 additional status. Different sets of items are necessary depending on how many bytes are modelled in the command 48 response. When using the IMPORT_UNIVERSAL_DEVICE_STATUS(n) macro, the parameter ‘n’ is used that specify the number of command 48 bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter ‘n’ to be used.

To import these required tables, a line similar to the following within the Universal import section (Note – A device implementing 10 “additional status” bytes in command 48 is shown in this example):

```
IMPORT_UNIVERSAL_DEVICE_STATUS(10)
```

The list of items imported using the IMPORT_UNIVERSAL_DEVICE_STATUS(n) macro is described in Table 5. The “n Values” column identifies for which value of ‘n’ a variable will be imported or not.

Table 5: IMPORT_DEVICE_STATUS(n) items

Item Type	Item Name	n Values	Description
VARIABLE	*device_specific_status_0*	≥ 1	Data referenced from command 48.
VARIABLE	*device_specific_status_1*	≥ 2	Data referenced from command 48.
VARIABLE	*device_specific_status_2*	≥ 3	Data referenced from command 48.
VARIABLE	*device_specific_status_3*	≥ 4	Data referenced from command 48.
VARIABLE	*device_specific_status_4*	≥ 5	Data referenced from command 48.
VARIABLE	*device_specific_status_5*	≥ 6	Data referenced from command 48.
VARIABLE	*device_specific_status_14*	≥ 15	Data referenced from command 48.
VARIABLE	*device_specific_status_15*	≥ 16	Data referenced from command 48.
VARIABLE	*device_specific_status_16*	≥ 17	Data referenced from command 48.
VARIABLE	*device_specific_status_17*	≥ 18	Data referenced from command 48.
VARIABLE	*device_specific_status_18*	≥ 19	Data referenced from command 48.
VARIABLE	*device_specific_status_19*	≥ 20	Data referenced from command 48.
VARIABLE	*device_specific_status_20*	≥ 21	Data referenced from command 48.
VARIABLE	*device_specific_status_21*	≥ 22	Data referenced from command 48.
VARIABLE	*device_specific_status_22*	≥ 23	Data referenced from command 48.
VARIABLE	*device_specific_status_23*	≥ 24	Data referenced from command 48.
VARIABLE	*device_specific_status_24*	25	Data referenced from command 48.

7.1.1.3 IMPORT_UNIVERSAL_EVENT_REF_ARRAY(*e, n*)

The IMPORT_UNIVERSAL_EVENT_REF_ARRAY(*e,n*) macro imports a list of all items required for implementation of event notification.

While event notifications are not a HART universal function, the device status masks and latched values derived from the universal command 48 data items and are used for event notifications are included in the universal DD to keep the definitions coupled to their source in a single location. The remainder of the event notification items are included in one of the common practice DDs (referenced later in this document).

Event notification in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the reference array model, which is limited to a maximum of 2 event notifications. The reference array model limit of 2 event notifications was chosen based on the requirement for a minimum of 2 event notifications to be supported by WirelessHART adapters. To support more than 2 event notifications, the Value Array implementation should be used (see 7.1.1.4). The parameter 'e' is used to specify how many event notifications are to be supported (allowable values are from 1 to 2). The reference array model may be preferred for compatibility in older host applications.

NOTE: If this macro is used, then do NOT use the IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(n) macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

Different sets of items are necessary depending on how many bytes are modelled in the command 48 response. When using the IMPORT_UNIVERSAL_EVENT_REF_ARRAY(e, n) macro, the parameter 'e' is used to specify the number of event notifications to be implemented, and the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the response code and device status). The same values for e and n have to be used in all macros that are used that call for parameters 'e' or 'n' to be used.

To import these required tables, add the following line within the Universal import section (Note – A device implementing 1 event notification message, and 10 "additional status" bytes in command 48 is shown in this example):

```
IMPORT_UNIVERSAL_EVENT_REF_ARRAY(1,10)
```

The list of items imported using the IMPORT_UNIVERSAL_EVENT_REF_ARRAY(*e, n*) macro is described in Table 6. The "e Values" and "n Values" columns identify for which values of 'e' and 'n' a variable will be imported or not.

Table 6: IMPORT_UNIVERSAL_EVENT_REF_ARRAY(e, n) items

Item Type	Item Name	e Values	n Values	Description
VARIABLE	*config_change_counter_latched_value*	≥ 1	≥ 1	Data referenced from command 119.
VARIABLE	*config_change_counter_latched_value_1*	2	≥ 1	Data referenced from command 119.
VARIABLE	*device_specific_status_0_latched_value*	≥ 1	≥ 1	Data referenced from command 119.
VARIABLE	*device_specific_status_0_latched_value_1*	2	≥ 1	Data referenced from command 119.
VARIABLE	*device_specific_status_0_mask*	≥ 1	≥ 1	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_0_mask_1*	2	≥ 1	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_1_latched_value*	≥ 1	≥ 2	Data referenced from command 119.
VARIABLE	*device_specific_status_1_latched_value_1*	2	≥ 2	Data referenced from command 119.
VARIABLE	*device_specific_status_1_mask*	≥ 1	≥ 2	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_1_mask_1*	2	≥ 2	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_2_latched_value*	≥ 1	≥ 3	Data referenced from command 119.
VARIABLE	*device_specific_status_2_latched_value_1*	2	≥ 3	Data referenced from command 119.
VARIABLE	*device_specific_status_2_mask*	≥ 1	≥ 3	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_2_mask_1*	2	≥ 3	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_3_latched_value*	≥ 1	≥ 4	Data referenced from command 119.
VARIABLE	*device_specific_status_3_latched_value_1*	2	≥ 4	Data referenced from command 119.
VARIABLE	*device_specific_status_3_mask*	≥ 1	≥ 4	Data referenced from commands 115,116.

Item Type	Item Name	e Values	n Values	Description
VARIABLE	*device_specific_status_3_mask_1*	2	≥ 4	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_4_latched_value*	≥ 1	≥ 5	Data referenced from command 119.
VARIABLE	*device_specific_status_4_latched_value_1*	2	≥ 5	Data referenced from command 119.
VARIABLE	*device_specific_status_4_mask*	≥ 1	≥ 5	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_4_mask_1*	2	≥ 5	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_5_latched_value*	≥ 1	≥ 6	Data referenced from command 119.
VARIABLE	*device_specific_status_5_latched_value_1*	2	≥ 6	Data referenced from command 119.
VARIABLE	*device_specific_status_5_mask*	≥ 1	≥ 6	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_5_mask_1*	2	≥ 6	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_14_latched_value*	≥ 1	≥ 15	Data referenced from command 119.
VARIABLE	*device_specific_status_14_latched_value_1*	2	≥ 15	Data referenced from command 119.
VARIABLE	*device_specific_status_14_mask*	≥ 1	≥ 15	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_14_mask_1*	2	≥ 15	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_15_latched_value*	≥ 1	≥ 16	Data referenced from command 119.
VARIABLE	*device_specific_status_15_latched_value_1*	2	≥ 16	Data referenced from command 119.
VARIABLE	*device_specific_status_15_mask*	≥ 1	≥ 16	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_15_mask_1*	2	≥ 16	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_16_latched_value*	≥ 1	≥ 17	Data referenced from command 119.
VARIABLE	*device_specific_status_16_latched_value_1*	2	≥ 17	Data referenced from command 119.
VARIABLE	*device_specific_status_16_mask*	≥ 1	≥ 17	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_16_mask_1*	2	≥ 17	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_17_latched_value*	≥ 1	≥ 18	Data referenced from command 119.
VARIABLE	*device_specific_status_17_latched_value_1*	2	≥ 18	Data referenced from command 119.
VARIABLE	*device_specific_status_17_mask*	≥ 1	≥ 18	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_17_mask_1*	2	≥ 18	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_18_latched_value*	≥ 1	≥ 19	Data referenced from command 119.
VARIABLE	*device_specific_status_18_latched_value_1*	2	≥ 19	Data referenced from command 119.

Item Type	Item Name	e Values	n Values	Description
VARIABLE	*device_specific_status_18_mask*	≥ 1	≥ 19	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_18_mask_1*	2	≥ 19	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_19_latched_value*	≥ 1	≥ 20	Data referenced from command 119.
VARIABLE	*device_specific_status_19_latched_value_1*	2	≥ 20	Data referenced from command 119.
VARIABLE	*device_specific_status_19_mask*	≥ 1	≥ 20	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_19_mask_1*	2	≥ 20	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_20_latched_value*	≥ 1	≥ 21	Data referenced from command 119.
VARIABLE	*device_specific_status_20_latched_value_1*	2	≥ 21	Data referenced from command 119.
VARIABLE	*device_specific_status_20_mask*	≥ 1	≥ 21	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_20_mask_1*	2	≥ 21	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_21_latched_value*	≥ 1	≥ 22	Data referenced from command 119.
VARIABLE	*device_specific_status_21_latched_value_1*	2	≥ 22	Data referenced from command 119.
VARIABLE	*device_specific_status_21_mask*	≥ 1	≥ 22	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_21_mask_1*	2	≥ 22	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_22_latched_value*	≥ 1	≥ 23	Data referenced from command 119.
VARIABLE	*device_specific_status_22_latched_value_1*	2	≥ 23	Data referenced from command 119.
VARIABLE	*device_specific_status_22_mask*	≥ 1	≥ 23	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_22_mask_1*	2	≥ 23	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_23_latched_value*	≥ 1	≥ 24	Data referenced from command 119.
VARIABLE	*device_specific_status_23_latched_value_1*	2	≥ 24	Data referenced from command 119.
VARIABLE	*device_specific_status_23_mask*	≥ 1	≥ 24	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_23_mask_1*	2	≥ 24	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_24_latched_value*	≥ 1	≥ 25	Data referenced from command 119.
VARIABLE	*device_specific_status_24_latched_value_1*	2	≥ 25	Data referenced from command 119.
VARIABLE	*device_specific_status_24_mask*	≥ 1	≥ 25	Data referenced from commands 115,116.
VARIABLE	*device_specific_status_24_mask_1*	2	≥ 25	Data referenced from commands 115,116.

7.1.1.4 IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(*n*)

The IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(n) macro imports a list of all items required for implementation of event notification.

While event notifications are not a HART universal function, the device status masks and latched values derived from the universal command 48 data items and are used for event notifications are included in the universal DD to keep the definitions coupled to their source in a single location. The remainder of the event notification items are included in one of the common practice DDs (referenced later in this document).

Event notification in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the value array model, which is recommended because it allows any number of event notifications to be modelled without creating unique variable copies for each supported event notification.

NOTE: If this macro is used, then do NOT use the IMPORT_UNIVERSAL_EVENT_REF_ARRAY(n_x) macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

Different sets of items are necessary depending on how many bytes are modelled in the command 48 response. When using the IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(n) macro, the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To import these required tables, add the following line within the Universal import section (Note – A device implementing 10 "additional status" bytes in command 48 is shown in this example):

```
IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(10)
```

The list of items imported using the IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(*n*) macro is described in Table 7. The "n Values" column identifies for which values of 'n' a variable will be imported or not.

Table 7: IMPORT_UNIVERSAL_EVENT_VAL_ARRAY(n) items

Item Type	Item Name	n Values	Description
VARIABLE	*config_change_counter_latched_value*	≥ 1	Data referenced from command 119.
VARIABLE	*device_specific_status_0_latched_value*	≥ 1	Data referenced from command 119.
VARIABLE	*device_specific_status_0_mask*	≥ 1	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_1_latched_value*	≥ 2	Data referenced from command 119.
VARIABLE	*device_specific_status_1_mask*	≥ 2	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_2_latched_value*	≥ 3	Data referenced from command 119.
VARIABLE	*device_specific_status_2_mask*	≥ 3	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_3_latched_value*	≥ 4	Data referenced from command 119.
VARIABLE	*device_specific_status_3_mask*	≥ 4	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_4_latched_value*	≥ 5	Data referenced from command 119.
VARIABLE	*device_specific_status_4_mask*	≥ 5	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_5_latched_value*	≥ 6	Data referenced from command 119.
VARIABLE	*device_specific_status_5_mask*	≥ 6	Data referenced from commands 115 and 116.

Item Type	Item Name	n Values	Description
VARIABLE	*device_specific_status_14_latched_value*	≥ 15	Data referenced from command 119.
VARIABLE	*device_specific_status_14_mask*	≥ 15	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_15_latched_value*	≥ 16	Data referenced from command 119.
VARIABLE	*device_specific_status_15_mask*	≥ 16	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_16_latched_value*	≥ 17	Data referenced from command 119.
VARIABLE	*device_specific_status_16_mask*	≥ 17	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_17_latched_value*	≥ 18	Data referenced from command 119.
VARIABLE	*device_specific_status_17_mask*	≥ 18	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_18_latched_value*	≥ 19	Data referenced from command 119.
VARIABLE	*device_specific_status_18_mask*	≥ 19	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_19_latched_value*	≥ 20	Data referenced from command 119.
VARIABLE	*device_specific_status_19_mask*	≥ 20	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_20_latched_value*	≥ 21	Data referenced from command 119.
VARIABLE	*device_specific_status_20_mask*	≥ 21	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_21_latched_value*	≥ 22	Data referenced from command 119.
VARIABLE	*device_specific_status_21_mask*	≥ 22	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_22_latched_value*	≥ 23	Data referenced from command 119.
VARIABLE	*device_specific_status_22_mask*	≥ 23	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_23_latched_value*	≥ 24	Data referenced from command 119.
VARIABLE	*device_specific_status_23_mask*	≥ 24	Data referenced from commands 115 and 116.
VARIABLE	*device_specific_status_24_latched_value*	≥ 25	Data referenced from command 119.
VARIABLE	*device_specific_status_24_mask*	≥ 25	Data referenced from commands 115 and 116.

7.1.1.5 Additional items

Starting with HART Revision 7, commands 38 and 48 were moved from common practice commands to universal commands. Because these commands are not mandatory for HART 5 and HART 6 devices to implement, they are not imported when using the IMPORT_UNIVERSAL_BASE(u) macro when u is specified to be either 5 or 6. However, commands 38 and 48 are strongly recommended to be implemented in all HART devices. If these commands are implemented in HART 5 or HART 6 devices, they need to be independently imported into the DD.

To import an individual item, add the item within the Universal import section (example shown below):

```
COMMAND reset_configuration_change_flag;
COMMAND read_additional_device_status;
```

The list of additional items (not covered in any of the previous macros) is described in Table 8.

Table 8: Other Universal items

Item Type	Item Name	Description
COMMAND	*read_additional_device_status*	Command Definition for command 48.

Item Type	Item Name	Description
COMMAND	*reset_configuration_change_flag*	Command Definition for command 38.

7.1.2 DEPENDENCIES

When using the IMPORT_UNIVERSAL_BASE(u) import, the following standard tables must also be imported:

- IMPORT_TABLES_BASE() Described in 7.13.1.1
- IMPORT_TABLES_DEVELOPER() Described in 7.13.1.5
- IMPORT_TABLES_HART(u) Described in 7.13.1.11

When using the IMPORT_UNIVERSAL_DEVICE_STATUS(n) import, the following standard tables must also be imported:

- IMPORT_TABLES_STANDARD_STATUS(u, n) Described in 7.13.1.22

When using IMPORT_UNIVERSAL_BASE(u) macro, the deviceVariable collection for all device variables must include the following members (see 6.9):

- CONFIG_UNITS
- DAMPING_VALUE
- DIGITAL_UNITS
- DIGITAL_VALUE
- LOWER_SENSOR_LIMIT
- MINIMUM_SPAN
- SENSOR_SERIAL_NUMBER
- UPPER_SENSOR_LIMIT

If the universal revision is rev 6 or greater, the following members must also be included:

- CLASSIFICATION
- DATA_QUALITY
- DEVICE_FAMILY_STATUS
- LIMIT_STATUS

7.1.3 REDEFINITIONS

7.1.3.1 REDEFINE_COMMAND_3_TRANSACTIONS(d)

The REDEFINE_COMMAND_3_TRANSACTIONS(d) macro is used in the REDEFINITIONS section of the Universal DD import. It is used to truncate the number of bytes used in command 3 to match the number of process variables implemented in the corresponding device. Devices that support all 4 of the process variables do not need to use this macro because the standard DD is defined to use all 4 variables by default. When using the REDEFINE_COMMAND_3_TRANSACTIONS(d) macro, the parameter ‘d’ is used to specify the number of process variables to be implemented. If only PV is implemented, then d=1. PV, and SV d=2, PV, SV, and TV d=3. The same value of d has to be used in all macros that are used that call for a parameter ‘d’ to be used.

To redefine the command 3 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing only PV and SV returned in command 3 is shown in this example):

```
REDEFINITIONS
{
```

```
    COMMAND read_dynamic_variables_and_pv_current
    {
        REDEFINE_COMMAND_3_TRANSACTIONS(2)
    }
}
```

7.1.3.2 REDEFINE_COMMAND_8_TRANSACTIONS(d)

The REDEFINE_COMMAND_8_TRANSACTIONS(d) macro is used in the REDEFINITIONS section of the Universal DD import. It is used to adjust the bytes used in command 8 according to the number of process variables implemented in the corresponding device. Devices that support all 4 of the process variables do not need to use this macro because the standard DD is defined to use all 4 variables by default. When using the REDEFINE_COMMAND_8_TRANSACTIONS(d) macro, the parameter 'd' is used to specify the number of process variables to be implemented. If only PV is implemented, then d=1. PV, and SV d=2, PV, SV, and TV d=3. The same value of d has to be used in all macros that are used that call for a parameter 'd' to be used.

To redefine the command 8 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing only PV and SV returned in command 8 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_dynamic_variable_classification
    {
        REDEFINE_COMMAND_8_TRANSACTIONS(2)
    }
}
```

7.1.3.3 REDEFINE_COMMAND_9_TRANSACTIONS(s)

The REDEFINE_COMMAND_9_TRANSACTIONS(s) macro is used in the REDEFINITIONS section of the Universal DD import. It is used to truncate the maximum number of variable slots supported in command 9 to match what is implemented in the corresponding device. Devices that support all 8 slots in command 9 do not need to use this macro because the standard DD is defined to use 8 slots by default. When using the REDEFINE_COMMAND_9_TRANSACTIONS(s) macro, the parameter 's' is used to specify the number of variable slots in command 9 to be implemented. The value of s used must be at least 4, and no more than 8. The same value of s has to be used in all macros that are used that call for a parameter 's' to be used.

NOTE: Only use this redefinition macro when the DD is modeling Universal revision 7 (see 7.1.1.1). HART Universal revision 6 has a fixed number of 4 slots used in command 9. This command does not need to be redefined for HART rev 6.

To redefine the command 9 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing only 4 slots in command 9 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_device_variables_and_status
    {
        REDEFINE_COMMAND_9_TRANSACTIONS(4)
    }
}
```

7.1.3.4 REDEFINE_H5_COMMAND_48_TRANSACTIONS(n)

The REDEFINE_H5_COMMAND_48_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Universal DD import when modeling universal HART revision 5. It is used to truncate the number of bytes used in command 48 to match the number of command 48 status bytes implemented in the corresponding device. Devices that support all 25 bytes of status in command 48 do not need to use this macro because the standard DD is defined

to use all 25 bytes by default. When using the REDEFINE_H5_COMMAND_48_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

NOTE: Only use this redefinition macro when the DD is modeling Universal revision 5 (see 7.1.1.1). See 7.1.3.5 and 7.1.3.6 for redefining command 48 for HART revs 6 and 7 respectively.

To redefine the command 48 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing 10 "additional status" bytes in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_additional_device_status
    {
        REDEFINE_H5_COMMAND_48_TRANSACTIONS(10)
    }
}
```

7.1.3.5 REDEFINE_H6_COMMAND_48_TRANSACTIONS(n)

The REDEFINE_H6_COMMAND_48_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Universal DD import when modeling universal HART revision 6. It is used to truncate the number of bytes used in command 48 to match the number of command 48 status bytes implemented in the corresponding device. Devices that support all 25 bytes of status in command 48 do not need to use this macro because the standard DD is defined to use all 25 bytes by default. When using the REDEFINE_H6_COMMAND_48_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

NOTE: Only use this redefinition macro when the DD is modeling Universal revision 6 (see 7.1.1.1). See 7.1.3.3 and 7.1.3.6 for redefining command 48 for HART revs 5 and 7 respectively.

To redefine the command 48 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing 10 "additional status" bytes in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_additional_device_status
    {
        REDEFINE_H6_COMMAND_48_TRANSACTIONS(10)
    }
}
```

7.1.3.6 REDEFINE_H7_COMMAND_48_TRANSACTIONS(n)

The REDEFINE_H7_COMMAND_48_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Universal DD import when modeling universal HART revision 7. It is used to truncate the number of bytes used in command 48 to match the number of command 48 status bytes implemented in the corresponding device. Devices that support all 25 bytes of status in command 48 do not need to use this macro because the standard DD is defined to use all 25 bytes by default. When using the REDEFINE_H7_COMMAND_48_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

NOTE: Only use this redefinition macro when the DD is modeling Universal revision 7 (see 7.1.1.1). See 7.1.3.3 and 7.1.3.5 for redefining command 48 for HART revs 5 and 6 respectively.

To redefine the command 48 transaction to match the device, add the following lines within the Universal import redefinition section (Note – A device implementing 10 “additional status” bytes in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_additional_device_status
    {
        REDEFINE_H7_COMMAND_48_TRANSACTIONS(10)
    }
}
```

7.1.3.7 Writable Response Preambles

HART DDs that model a device that supports writing the number of response preambles, the handling of response_preambles needs to be redefined to READ & WRITE. This redefinition should be included when the device supports command 59 to write the response preambles. In this case, the COMMAND write_number_of_response_preambles must be imported from the common practice miscellaneous DD.

To redefine response_preambles to match the device, add the following lines within the Universal import redefinition section:

```
REDEFINITIONS
{
    VARIABLE response_preambles
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.1.3.8 Device Specific Status

Command 48 to read additional device status is defined by HART protocol specifications as a universal command. Command 48 includes multiple bit-enumerated bytes whose bit meanings are device-specific and can vary from one device to the next. Consequently, the bit-enumerations of the device specific status bytes needs to be redefined for each product DD.

For DDs that are modeling event notifications, the device_specific_status_xx_mask and device_specific_status_xx_latched_value also needs to be redefined to match the device specific names used for each of the additional status bits.

To redefine the device-specific status bits, add lines similar to the following for each of the supported device-specific status bytes:

```
REDEFINITIONS
{
    VARIABLE device_specific_status_0
    {
        REDEFINE LABEL "My Sensor Status";
        REDEFINE HELP "My Sensor Status– all status that may be at the sensor level.";
        REDEFINE TYPE BIT_ENUMERATED
        {
            {
                0x80, "My Device Failure", "My Help", HARDWARE
            },
            {
                0x40, "My Configuration Failure”, "My Help", SOFTWARE
            },
            …
```

```
            }
        }

        VARIABLE device_specific_status_0_mask
        {
            REDEFINE LABEL "My Sensor Status Mask";
            REDEFINE HELP "My Sensor Status– all status that may be at the sensor level.";
            REDEFINE TYPE BIT_ENUMERATED
            {
                {
                    0x80, "My Device Failure"
                },
                {
                    0x40, "My Configuration Failure"
                },
                …
            }
        }

        VARIABLE device_specific_status_0_latched_value
        {
            REDEFINE LABEL "My Sensor Status Event";
            REDEFINE HELP "My Sensor Status– all status that may be at the sensor level.";
            REDEFINE TYPE BIT_ENUMERATED
            {
                {
                    0x80, "My Device Failure"
                },
                {
                    0x40, "My Configuration Failure"
                },
                …
            }
        }
    }
```

7.1.3.9 Time Formats

Depending on the measurement types and capabilities of the field device, the time format of certain variables may need to be adjusted to match characteristics of the field device. Variables such as time_stamp, and others may need to have their time formats adjusted.

To redefine the time format for a variable, add lines similar to the following for all applicable variables:

```
    REDEFINITIONS
    {
        VARIABLE time_stamp
        {
            TYPE TIME_VALUE
            {
                REDEFINE TIME_FORMAT %H:%M";
            }
        }
    }
```

7.1.3.10 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter ‘x’ is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE transmitter_revision
    {
        SET_DEFAULT(2);
    }
}
```

7.1.3.11 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE polling_address
    {
        REDEFINE HELP "Address used by the host to identify a Field Device. This address is
changeable by the user to allow multiple devices on the loop to be assigned unique polling
addresses. This device defaults to address 63.";
    }
}
```

7.2 HART PV1 DD

The PV1 DD includes all items needed to implement the process variables for single Analog Output channel devices. The HART PV1 binary files are installed in \HCF\DDL\Library\000000\0016\0103.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\PV1_3.ddl.

To import the PV1 DD into a product DD, use the following IMPORT statement:

```
IMPORT STANDARD _PV, DEVICE_REVISION 1, DD_REVISION 3
{
    IMPORT_PV1_BASE()
}
```

NOTE: If this DD is imported, do NOT import the PV2 DD.

Import macros have been provided to make it easier to extract the correct items from the HART PV DD. A description of each of these macros is described in 7.2.1.

7.2.1 IMPORTS

7.2.1.1 IMPORT_PV1_BASE(*d*)

The IMPORT_PV1_BASE(d) macro imports all items required to implement the set of dynamic process variables for single analog output channel devices. When using the IMPORT_PV1_BASE(d) macro, the parameter 'd' is used to specify how many of the HART dynamic variables are to be supported (1 = PV only, 2 = PV and SV, 3 = PV, SV, and TV, 4 = PV, SV, TV, and QV). The same value of d has to be used in all macros that are used that call for a parameter 'd' to be used.

To import these required tables, add the following line within the PV import section (Note – A device implementing only PV, and SV is shown in this example):

IMPORT_PV1_BASE(2)

The list of items imported using the IMPORT_PV1_BASE(d) macro is described in Table 9. The "d Values" column identifies for which value of 'd' a variable will be imported or not.

Table 9: IMPORT_PV1_BASE(d) items

Item Type	Item Name	d Values	Description
COLLECTION	*analog_io*	≥ 1	Referenced from commands 2, 3, 15, 40, 45, 46, 61, 100.
ARRAY OF COLLECTION	*dynamic_variables*	≥ 1	Referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 61, 100, 110.
VARIABLE	*loopCurrent*	≥ 1	Data referenced from commands 2 and 3.
VARIABLE	*lowerRange_value*	≥ 1	Data referenced from commands 15 and 35.
VARIABLE	*percentRange*	≥ 1	Data referenced from command 2.
COLLECTION	*primary_variable*	≥ 1	Referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 61, 100, 110.
COLLECTION	*quaternary_variable*	= 4	Referenced from commands 3, 8, 61, 110.
COLLECTION	*scaling*	≥ 1	Referenced from commands 2, 15, 35, 47.
UNIT	*scaling_units_relation*	≥ 1	Unit relation that associates upper and lower range value to an engineering unit variable.
COLLECTION	*secondary_variable*	≥ 2	Referenced from commands 3, 8, 61, 110.
COLLECTION	*tertiary_variable*	≥ 3	Referenced from commands 3, 8, 61, 110.

Item Type	Item Name	d Values	Description
VARIABLE	*upperRange_value*	≥ 1	Data referenced from commands 15 and 35.

7.2.1.2 Additional items

Some additional less often to be used tables are also included in the standard PV DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually.

To import an individual item, add the item within the PV import section (example shown below):

```
WRITE_AS_ONE scaling_wao;
```

The list of additional items (not covered in any of the previous macros) is described in Table 10.

Table 10: Other PV1 items

Item Type	Item Name	Description
WRITE_AS_ONE	*scaling_wao*	A relation that forces the upper and lower range values to be edited together as a pair.

7.2.2 DEPENDENCIES

When using the IMPORT_PV1_BASE(d) import, there is a dependency on common practice device variable imports:

- IMPORT_DEV_VARS_MAPPING_READ() Described in 7.4.1.4
- IMPORT_DEV_VARS_MAPPING_READ_WRITE() Described in 7.4.1.5

7.2.3 REDEFINITIONS

7.2.3.1 REDEFINE_DYNAMIC_VAR_SIZE(*d*)

The REDEFINE_DYNAMIC_VAR_SIZE(d) macro removes the unsupported elements from the dynamic_variables array. Devices that do not support all 4 of the HART dynamic variables (PV, SV, TV, and QV) need to redefine the dynamic_variables array using the REDEFINE_DYNAMIC_VAR_SIZE(d) macro. The parameter ‘d’ is used to specify the number of process variables to be implemented. If only PV is implemented, then d=1. PV, and SV d=2, PV, SV, and TV d=3. The same value of d has to be used in all macros that are used that call for a parameter ‘d’ to be used.

To redefine the dynamic_variables array, add the following lines to the redefinitions section of the PV1 import (Note – A device implementing only PV, and SV is shown in this example):

```
REDEFINITIONS
{
    ARRAY OF COLLECTION dynamic_variables
    {
        ELEMENTS
        {
            REDEFINE_DYNAMIC_VAR_SIZE(2)
        }
    }
}
```

7.2.3.2 Writeable loop_current

The variable loop_current is defined to be READ only, but can be redefined to READ & WRITE if the device supports a device specific way to control the loop output from a menu on the UI. In most cases, the loop output is either calculated by the field device or consumed by an actuation device. In these cases, loop_current need not be handled with a WRITE command. To redefine the loop_current to be writeable, add the following to the redefinition section of the PV import:

```
REDEFINITIONS
{
    VARIABLE loop_current
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.2.3.3 Fixed Mapping

The PV1 HART standard DD implements discoverable mapping where the PV, SV, TV, and QV variable mapping can be read from the field device using Common Practice command 50. This is the recommended implementation to allow all host applications to discover a device's variable mapping – even if the mapping can change over the life of a device (e.g. using command 51 to configure a different mapping, or receiving new devices of the same type from the manufacturer with different mapping). Even if a field device does not implement the ability to change the variable mapping using command 51, it is recommended to implement the ability to read the mapping using command 50. For devices that do not implement the device variable mapping (i.e. the IMPORT_DEV_VARS_MAPPING_READ() or IMPORT_DEV_VARS_MAPPING_READ_WRITE() macros are not imported from the Common Practice Device Variables DD), then the variable mapping in the DD needs to be redefined to a fixed mapping.

To redefine the mapping to a fixed mapping, the following lines should be added to the redefinitions section of the PV1 import section where "FIXED_PRIMARY_INDEX", "FIXED_SECONDARY_INDEX", "FIXED_TERTIARY_INDEX", and "FIXED_QUATERNARY_INDEX" are defined to be the values of the variable indexes into the deviceVariables array. For devices that only support a subset of the 4 dynamic variables, only those that are applicable to the device need be redefined.

```
REDEFINITIONS
{
    COLLECTION OF VARIABLE scaling
    {
        MEMBERS
        {
            REDEFINE RANGE_UNITS, deviceVariables[FIXED_PRIMARY_INDEX].CONFIG_UNITS,
                                                  range_unit_temp, [range_units_help];
        }
    }

    COLLECTION OF COLLECTION primary_variable
    {
        MEMBERS
        {
            REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_PRIMARY_INDEX];
        }
    }

    COLLECTION OF COLLECTION secondary_variable
    {
        MEMBERS
        {
            REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_SECONDARY_INDEX];
        }
    }

    COLLECTION OF COLLECTION tertiary_variable
    {
        MEMBERS
        {
            REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_TERTIARY_INDEX];
        }
    }
```

```
    COLLECTION OF COLLECTION quaternary_variable
    {
        MEMBERS
        {
            REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_QUATERNARY_INDEX];
        }
    }

    REDEFINE UNIT scaling_units_relation
    {
        deviceVariables[FIXED_PRIMARY_INDEX].CONFIG_UNITS:
            upperRange_value,
            lowerRange_value
    }
}
```

7.2.3.4 DELETE ANALOG_CHANNEL_FLAGS

HART Universal rev 5 did not have analog channel flags. For DDs modeling a HART rev 5 device, the additional collection member should be deleted as follows:

```
COLLECTION OF VARIABLE analog_io
{
    MEMBERS
    {
        DELETE ANALOG_CHANNEL_FLAGS;
    }
}
```

7.2.3.5 Display and Edit Formats

Depending on the measurement types and capabilities of the field device, the display format and edit format of certain variables may need to be adjusted to match characteristics of the field device. Variables such as loopCurrent, percentRange, upperRange_value, lowerRange_value, and others may need to have their display and/or edit formats adjusted.

To redefine the display or edit format for a variable, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        TYPE FLOAT
        {
            REDEFINE DISPLAY_FORMAT ".2f";
            REDEFINE EDIT_FORMAT ".2f";
        }
    }
}
```

7.2.3.6 Voltage Devices

For devices that are voltage output devices instead of current loop devices, the constant units for some items may need to be redefined to voltage units.

To redefine the output to be voltage based, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        REDEFINE CONSTANT_UNIT volts_temp;
    }
```

```
}
```

7.2.3.7 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE lowerRange_value
    {
        SET_DEFAULT(-40);
    }
}
```

7.2.3.8 Minimum and Maximum Values

It is best practice for all numeric values that are writeable to define an upper and lower limit. This helps to prevent an out of bounds entry from being attempted to be sent to a field device.

To define minimum and maximum values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE lowerRange_value
    {
        REDEFINE MIN_VALUE -200.0;
        REDEFINE MAX_VALUE 850.0;
    }
}
```

7.2.3.9 Label

In order to maintain consistency of standardized items, the LABEL of HART Standard DD item should not normally be redefined. The exception to this is for voltage devices (instead of loop current devices) may need to change the labels of standardized output variables to be indicative of being voltage driven instead of current driven.

To redefine the label of a loop current specific item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        REDEFINE LABEL "Primary Output Voltage";
    }
}
```

7.2.3.10 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        REDEFINE HELP "Primary Output Voltage- The voltage level calculated to control the
analog channel output. This output uses terminals 1 and 2 located under the main access
panel.";
    }
}
```

7.3 HART PV2 DD

The PV2 DD includes all items needed to implement the process variables for multiple Analog Output channel devices. The HART PV2 binary files are installed in \HCF\DDL\Library\000000\0016\0203.* and 0204.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\PV2_3.ddl and PV2_4.dll.

There are 2 revisions of the PV2 DD. DD revision 3 is implemented according to Common Practice rev 7. DD revision 4 implements the enhancements that were added to specification in Common Practice rev 11. Import the PV2 DD revision corresponding to the Common Practice Revision that the device is designed to. Common Practice rev 11 DD appends the analog limit values to command 70.

To import the PV2 DD into a product DD, use one of the following IMPORT statements:

```
IMPORT STANDARD _PV, DEVICE_REVISION 2, DD_REVISION 3
{
    IMPORT_PV2_3_BASE()
}
IMPORT STANDARD _PV, DEVICE_REVISION 2, DD_REVISION 4
{
    IMPORT_PV2_4_BASE()
}
```

NOTE: If this DD is imported, do NOT import the PV1 DD.

Import macros have been provided to make it easier to extract the correct items from the HART PV DD. A description of each of these macros is described in 7.3.1.

7.3.1 IMPORTS

7.3.1.1 IMPORT_PV2_3_BASE()

The IMPORT_PV2_3_BASE() macro imports all items required to implement the set of process variables for multiple analog output channel devices as was originally defined in Common Practice rev 7 before the analog upper and lower limits were appended to the end of command 70.

To import these required tables, add the following line within the PV import section:

```
IMPORT_PV2_3_BASE()
```

The list of items imported using the IMPORT_PV2_3_BASE() macro is described in Table 12.

Table 11: IMPORT_PV2_3_BASE() items

Item Type	Item Name	Description
COLLECTION	*analog_io*	Referenced from commands 2, 3, 15, 40, 45, 46, 60, 61, 62, 63, 64, 66, 67, 68, 70, 100, 535.
ARRAY OF COLLECTION	*dynamic_variables*	Data referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 100, 110, 535.
VARIABLE	*loopCurrent*	Data referenced from commands 2, 3, 40, 45, 46, 60, 61, 62, 66, 67, 68.
VARIABLE	*lowerRange_value*	Data referenced from commands 15, 35, 63, 65.
VARIABLE	*percentRange*	Data referenced from commands 2 and 60.
COLLECTION	*primary_variable*	Referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 61, 100, 110.
VARIABLE	*PVloopCurrentLowerEndPoint*	Data referenced from command 70 and 535.

Item Type	Item Name	Description
VARIABLE	*PVloopCurrentUpperEndPoint*	Data referenced from command 70 and 535.
VARIABLE	*PVloopDamping*	Data referenced from commands 63 and 64.
VARIABLE	*PVloopUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
COLLECTION	*quaternary_variable*	Referenced from commands 3, 8, 61 and 110.
COLLECTION	*QVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.
VARIABLE	*QVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*QVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*QVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
VARIABLE	*QVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*QVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*QVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*QVpercentRange*	Data referenced from command 60.
COLLECTION	*QVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*QVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*QVupperRange_value*	Data referenced from commands 63 and 65.
COLLECTION	*scaling*	Referenced from commands 2, 15, 35, 47, 60, 63, 65, 69.
UNIT	*scaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
COLLECTION	*secondary_variable*	Referenced from commands 3, 8, 61, 110.
COLLECTION	*SVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.
VARIABLE	*SVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*SVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*SVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
VARIABLE	*SVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*SVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*SVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*SVpercentRange*	Data referenced from command 60.
COLLECTION	*SVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*SVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*SVupperRange_value*	Data referenced from commands 63 and 65.
COLLECTION	*tertiary_variable*	Referenced from commands 3, 8, 61 and 110.
COLLECTION	*TVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.
VARIABLE	*TVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*TVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*TVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.

Item Type	Item Name	Description
VARIABLE	*TVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*TVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*TVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*TVpercentRange*	Data referenced from command 60.
COLLECTION	*TVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*TVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*TVupperRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*upperRange_value*	Data referenced from commands 15, 35, 63, 65.

7.3.1.2 IMPORT_PV2_4_BASE()

The IMPORT_PV2_4_BASE() macro imports all items required to implement the set of process variables for multiple analog output channel devices as was updated starting in Common Practice rev 10.

To import these required tables, add the following line within the PV import section:

```
IMPORT_PV2_4_BASE()
```

The list of items imported using the IMPORT_PV2_4_BASE() macro is described in Table 12.

Table 12: IMPORT_PV2_4_BASE() items

Item Type	Item Name	Description
COLLECTION	*analog_io*	Referenced from commands 2, 3, 15, 40, 45, 46, 60, 61, 62, 63, 64, 66, 67, 68, 70, 100, 535.
ARRAY OF COLLECTION	*dynamic_variables*	Referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 100, 110, 535.
VARIABLE	*loopCurrent*	Data referenced from commands 2, 3, 40, 45, 46, 60, 61, 62, 66, 67, 68.
VARIABLE	*lowerRange_value*	Data referenced from commands 15, 35, 63, 65.
VARIABLE	*percentRange*	Data referenced from commands 2 and 60.
COLLECTION	*primary_variable*	Referenced from commands 1, 2, 3, 8, 14, 15, 34, 35, 40, 44, 45, 46, 47, 49, 61, 100, 110.
VARIABLE	*PVloopCurrentLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*PVloopCurrentLowerLimit*	Data referenced from command 70.
VARIABLE	*PVloopCurrentUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*PVloopCurrentUpperLimit*	Data referenced from command 70.
VARIABLE	*PVloopDamping*	Data referenced from commands 63 and 64.
VARIABLE	*PVloopUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
COLLECTION	*quaternary_variable*	Referenced from commands 3, 8, 61 and 110.
COLLECTION	*QVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.

Item Type	Item Name	Description
VARIABLE	*QVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*QVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*QVanalogChannelLowerLimit*	Data referenced from command 70.
VARIABLE	*QVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
VARIABLE	*QVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*QVanalogChannelUpperLimit*	Data referenced from command 70.
VARIABLE	*QVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*QVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*QVpercentRange*	Data referenced from command 60.
COLLECTION	*QVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*QVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*QVupperRange_value*	Data referenced from commands 63 and 65.
COLLECTION	*scaling*	Referenced from commands 2, 15, 35, 47, 60, 63, 65, 69.
UNIT	*scaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
COLLECTION	*secondary_variable*	Referenced from commands 3, 8, 61, 110.
COLLECTION	*SVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.
VARIABLE	*SVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*SVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*SVanalogChannelLowerLimit*	Data referenced from command 70.
VARIABLE	*SVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.
VARIABLE	*SVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*SVanalogChannelUpperLimit*	Data referenced from command 70.
VARIABLE	*SVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*SVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*SVpercentRange*	Data referenced from command 60.
COLLECTION	*SVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*SVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*SVupperRange_value*	Data referenced from commands 63 and 65.
COLLECTION	*tertiary_variable*	Referenced from commands 3, 8, 61 and 110.
COLLECTION	*TVanalog_io*	Referenced from commands 60, 61, 62, 63, 64, 66, 67, 68, 70, 535.
VARIABLE	*TVanalogChannelDamping*	Data referenced from commands 63 and 64.
VARIABLE	*TVanalogChannelLowerEndPoint*	Data referenced from command 70, 535.
VARIABLE	*TVanalogChannelLowerLimit*	Data referenced from command 70.
VARIABLE	*TVanalogChannelUnits*	Data referenced from commands 60, 61, 62, 66, 67, 68, 70, 535.

Item Type	Item Name	Description
VARIABLE	*TVanalogChannelUpperEndPoint*	Data referenced from command 70, 535.
VARIABLE	*TVanalogChannelUpperLimit*	Data referenced from command 70.
VARIABLE	*TVanalogChannelValue*	Data referenced from commands 60, 61, 62, 66, 67, 68.
VARIABLE	*TVlowerRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*TVpercentRange*	Data referenced from command 60.
COLLECTION	*TVscaling*	Referenced from commands 60, 63, 65, 69.
UNIT	*TVscaling_units_relation*	Unit relation that associates upper and lower range value to an engineering unit variable.
VARIABLE	*TVupperRange_value*	Data referenced from commands 63 and 65.
VARIABLE	*upperRange_value*	Data referenced from commands 15, 35, 63, 65.

7.3.1.3 Additional items

Some additional less often to be used tables are also included in the standard tables DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually.

To import an individual item, add the item within the PV import section (example shown below):

```
WRITE_AS_ONE scaling_wao;
```

The list of additional items (not covered in any of the previous macros) is described in Table 13.

Table 13: Other PV2 items

Item Type	Item Name	Description
WRITE_AS_ONE	*QVscaling_wao*	A relation that forces the upper and lower range values to be edited together as a pair.
WRITE_AS_ONE	*scaling_wao*	A relation that forces the upper and lower range values to be edited together as a pair.
WRITE_AS_ONE	*SVscaling_wao*	A relation that forces the upper and lower range values to be edited together as a pair.
WRITE_AS_ONE	*TVscaling_wao*	A relation that forces the upper and lower range values to be edited together as a pair.

7.3.2 DEPENDENCIES

When using the IMPORT_PV2_*_BASE() import, the following standard tables must also be imported:

- IMPORT_TABLES_MULTI_AO() Described in 7.13.1.17

When using the IMPORT_PV2_*_BASE() import, there is a dependency on common practice device variable imports:

- IMPORT_DEV_VARS_MAPPING_READ() Described in 7.4.1.4
- IMPORT_DEV_VARS_MAPPING_READ_WRITE() Described in 7.4.1.5

7.3.3 REDEFINITIONS

7.3.3.1 Writeable loop_current

The variables loop_current, SVanalogChannelValue, TVanalogChannelValue and QVanalogChannelValue are defined to be READ only, but can be redefined to READ & WRITE if the device supports a device specific way to control the loop output from a menu on the UI. In most cases, the loop outputs are either calculated by the field device or consumed by an actuation device. In these cases, loop_current, SVanalogChannelValue, TVanalogChannelValue and QVanalogChannelValue do not need to be handled with a WRITE command. To redefine the loop_current, SVanalogChannelValue, TVanalogChannelValue and QVanalogChannelValue to be writeable, add the following to the redefinition section of the PV import. For devices that only support a subset of the 4 dynamic variables, only those that are applicable to the device need be redefined.

```
REDEFINITIONS
{
    VARIABLE loop_current
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE SVanalogChannelValue
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE TVanalogChannelValue
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE QVanalogChannelValue
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.3.3.2 Writable analog endpoint values

For devices that implement command 535 write_analog_channel_endpoint_values (and are imported from the Common Practice Analog Channels DD see 7.5.1.2), the analog endpoint variables need to be made writeable.

To redefine the analog channel endpoint values to be writeable, add the following to the redefinition section of the PV import.

```
REDEFINITIONS
{
    VARIABLE PVloopCurrentUpperEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE PVloopCurrentLowerEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE SVanalogChannelUpperEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE SVanalogChannelLowerEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE TVanalogChannelUpperEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
```

```
    }
    VARIABLE TVanalogChannelLowerEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE QVanalogChannelUpperEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
    VARIABLE QVanalogChannelLowerEndPoint
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.3.3.3 Fixed Mapping

The PV2 HART standard DD implements discoverable mapping where the PV, SV, TV, and QV variable mapping can be read from the field device using Common Practice command 50. This is the recommended implementation to allow all host applications to discover a device's variable mapping – even if the mapping can change over the life of a device (e.g. using command 51 to configure a different mapping, or receiving new devices of the same type from the manufacturer with different mapping). Even if a field device does not implement the ability to change the variable mapping using command 51, it is recommended to implement the ability to read the mapping using command 50. For devices that do not implement the device variable mapping (i.e. the IMPORT_DEV_VARS_MAPPING_READ() or IMPORT_DEV_VARS_MAPPING_READ_WRITE() macros are not imported from the Common Practice Device Variables DD), then the variable mapping in the DD needs to be redefined to a fixed mapping.

To redefine the mapping to a fixed mapping, the following lines should be added to the redefinitions section of the PV2 import section where "FIXED_PRIMARY_INDEX", "FIXED_SECONDARY_INDEX", "FIXED_TERTIARY_INDEX", and "FIXED_QUATERNARY_INDEX" are defined to be the values of the variable indexes into the deviceVariables array. For devices that only support a subset of the 4 dynamic variables, only those that are applicable to the device need be redefined.

```
REDEFINITIONS
{
    COLLECTION OF VARIABLE scaling
    {
        MEMBERS
        {
            REDEFINE RANGE_UNITS, deviceVariables[FIXED_PRIMARY_INDEX].CONFIG_UNITS,
                                                  range_unit_unit, [range_units_help];
        }
    }

    COLLECTION OF VARIABLE SVscaling
    {
        MEMBERS
        {
            REDEFINE RANGE_UNITS, deviceVariables[FIXED_SECONDARY_INDEX].CONFIG_UNITS,
                                                  range_unit_temp, [range_units_help];
        }
    }

    COLLECTION OF VARIABLE TVscaling
    {
        MEMBERS
        {
            REDEFINE RANGE_UNITS, deviceVariables[FIXED_TERTIARY_INDEX].CONFIG_UNITS,
                                                  range_unit_temp, [range_units_help];
        }
```

```
}

COLLECTION OF VARIABLE QVscaling
{
    MEMBERS
    {
        REDEFINE RANGE_UNITS,deviceVariables[FIXED_QUATERNARY_INDEX].CONFIG_UNITS,
                                              range_unit_temp, [range_units_help];
    }
}

COLLECTION OF COLLECTION primary_variable
{
    MEMBERS
    {
        REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_PRIMARY_INDEX];
    }
}

COLLECTION OF COLLECTION secondary_variable
{
    MEMBERS
    {
        REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_SECONDARY_INDEX];
    }
}

COLLECTION OF COLLECTION tertiary_variable
{
    MEMBERS
    {
        REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_TERTIARY_INDEX];
    }
}

COLLECTION OF COLLECTION quaternary_variable
{
    MEMBERS
    {
        REDEFINE DEVICE_VARIABLE,        deviceVariables[FIXED_QUATERNARY_INDEX];
    }
}

REDEFINE UNIT scaling_units_relation
{
    deviceVariables[FIXED_PRIMARY_INDEX].CONFIG_UNITS:
        upperRange_value,
        lowerRange_value
}

REDEFINE UNIT SVscaling_units_relation
{
    deviceVariables[FIXED_SECONDARY_INDEX].CONFIG_UNITS:
        upperRange_value,
        lowerRange_value
}

REDEFINE UNIT TVscaling_units_relation
{
    deviceVariables[FIXED_TERTIARY_INDEX].CONFIG_UNITS:
        upperRange_value,
        lowerRange_value
}
```

```
    REDEFINE UNIT QVscaling_units_relation
    {
        deviceVariables[FIXED_QUATERNARY_INDEX].CONFIG_UNITS:
            upperRange_value,
            lowerRange_value
    }
}
```

7.3.3.4 DELETE ANALOG_CHANNEL_FLAGS

HART Universal rev 5 did not have analog channel flags. For DDs modeling a HART rev 5 device, the additional collection member should be deleted as follows:

```
COLLECTION OF VARIABLE analog_io
{
    MEMBERS
    {
        DELETE ANALOG_CHANNEL_FLAGS;
    }
}
COLLECTION OF VARIABLE SVanalog_io
{
    MEMBERS
    {
        DELETE ANALOG_CHANNEL_FLAGS;
    }
}
COLLECTION OF VARIABLE TVanalog_io
{
    MEMBERS
    {
        DELETE ANALOG_CHANNEL_FLAGS;
    }
}
COLLECTION OF VARIABLE QVanalog_io
{
    MEMBERS
    {
        DELETE ANALOG_CHANNEL_FLAGS;
    }
}
```

7.3.3.5 Display and Edit Formats

Depending on the measurement types and capabilities of the field device, the display format and edit format of certain variables may need to be adjusted to match characteristics of the field device. Variables such as loopCurrent, percentRange, upperRange_value, lowerRange_value, and others may need to have their display and/or edit formats adjusted.

To redefine the display or edit format for a variable, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        TYPE FLOAT
        {
            REDEFINE DISPLAY_FORMAT ".2f";
            REDEFINE EDIT_FORMAT ".2f";
        }
    }
```

```
}
```

7.3.3.6 Voltage Devices

For devices that are voltage output devices instead of current loop devices, the constant units for some items may need to be redefined to voltage units.

To redefine the output to be voltage based, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE PVloopUnits
    {
        REDEFINE TYPE ENUMERATED
        {
            UNITS_CODE(58)
        }
    }
}
```

7.3.3.7 Configurable Analog Channels

For devices that have configurable analog channels (e.g. can be configured to be voltage, current, or frequency), the engineering units for the channels may need to adjust to match the type of analog channel being configured.

To redefine the analog channels to use units code variables (rather than constant units), add lines similar to the following for all applicable analog channel values.

```
REDEFINITIONS
{
    VARIABLE SVanalogChannelValue
    {
        DELETE CONSTANT_UNIT;
    }
}

UNIT SVanalogChannelUnitRelation
{
    SVanalogChannelUnits: SVanalogChannelValue
}
```

7.3.3.8 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter ‘x’ is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE lowerRange_value
    {
        SET_DEFAULT(-40);
    }
}
```

7.3.3.9 Minimum and Maximum Values

It is best practice for all numeric values that are writeable to define an upper and lower limit. This helps to prevent an out of bounds entry from being attempted to be sent to a field device.

To define minimum and maximum values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE lowerRange_value
    {
        REDEFINE MIN_VALUE -200.0;
        REDEFINE MAX_VALUE 850.0;
    }
}
```

7.3.3.10 Label

In order to maintain consistency of standardized items, the LABEL of HART Standard DD item should not normally be redefined. The exception to this is for voltage devices (instead of loop current devices) may need to change the labels of standardized output variables to be indicative of being voltage driven instead of current driven.

To redefine the label of a loop current specific item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        REDEFINE LABEL "Primary Output Voltage";
    }
}
```

7.3.3.11 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE loopCurrent
    {
        REDEFINE HELP "Primary Output Voltage- The voltage level calculated to control the
analog channel output. This output uses terminals 1 and 2 located under the main access
panel.";
    }
}
```

7.4 HART Common Practice Device Variables DD

The common practice commands do not have to be implemented by every instrument and DD, but they must be implemented and defined exactly as defined by the FCG group if they are used.

The HART Common Practice Device Variables DD includes all items needed to implement HART device variables. The HART Common Practice Device Variables binary files are installed in \HCF\DDL\Library\000000\0019\0706.*, 0804.*, 0904.*, 0b02.* and 0c02.*. The corresponding DDL source code for these files is located in HCF\DDL\HCFinfo\Standards\000000\Common7_6_Device_Vars.ddl, Common8_4_Device_Vars.ddl, Common9_4_Device_Vars.ddl, Common11_2_Device_Vars.ddl and Common12_2_Device_Vars.ddl.

Only one device revision of the DD (either 7, 8, 9, 11 or 12) should be imported into a device DD depending on which HART universal revision is being modelled. To import the common practice device variables DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 5 */
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 7, DD_REVISION 6
{
    IMPORT_DEV_VARS_BASE()
}

/* For HART Rev 6 */
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 8, DD_REVISION 4
{
    IMPORT_DEV_VARS_BASE()
}

/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_DEV_VARS_BASE()
}

/* For HART Rev 7 Common Practice rev 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_DEV_VARS_BASE()
}

/* For HART Rev 7 Common Practice rev 12 */
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 12, DD_REVISION 2
{
    IMPORT_DEV_VARS_BASE()
}
```

Import macros have been provided to make it easier to extract the correct items from the HART Common Practice Device Variables DD. A description of each of these macros is described in 7.4.1.

7.4.1 IMPORTS

7.4.1.1 IMPORT_DEV_VARS_BASE()

The IMPORT_DEV_VARS_BASE() macro imports all items required to implement device variable functionality.

To import these required items, add the following line within the common practice device variable import section:

```
IMPORT_DEV_VARS_BASE()
```

The list of items imported using the IMPORT_DEV_VARS_BASE() macro is described in Table 14.

Table 14: IMPORT_DEV_VARS_BASE() items

Item Type	Item Name	Description
VARIABLE	*device_variable_code*	Data referenced from command 52, 53, 54, 55, 56, 79, 522, and 534.
COMMAND	*read_device_variable_information*	Command Definition for command 54.

7.4.1.2 IMPORT_DEV_VARS_CATCH_V11()

The IMPORT_DEV_VARS_CATCH_V11() macro imports a list of all items required to implement the device variable catch mechanism as was updated starting with Common Practice rev 11 DD.

NOTE: If the IMPORT_DEV_VARS_CATCH_V8() macro is used, do NOT use the IMPORT_DEV_VARS_CATCH_V11() macro.

NOTE: This macro can only be used in DDs modeling HART Universal rev 7 and Common Practice rev 10 or later.

To import these items, add the following line within the common practice device variable import section:

```
IMPORT_DEV_VARS_CATCH_V11()
```

The list of items imported using the IMPORT_DEV_VARS_CATCH_V11() macro is described in Table 15.

Table 15: IMPORT_DEV_VARS_CATCH_V11() items

Item Type	Item Name	Description
COMMAND	*catch_device_variable*	Command Definition for command 113.
VARIABLE	*destination_device_variable_code*	Data referenced from commands 113 and 114.
COMMAND	*read_caught_device_variable*	Command Definition for command 114.
VARIABLE	*source_command_number_byte*	Data referenced from command 114.

7.4.1.3 IMPORT_DEV_VARS_CATCH_V8()

The IMPORT_DEV_VARS_CATCH_V8() macro imports a list of all items required to implement the device variable catch mechanism as it was originally defined starting in common practice revision 8.

NOTE: If the IMPORT_DEV_VARS_CATCH_V11() macro is used, do NOT use the IMPORT_DEV_VARS_CATCH_V8() macro.

NOTE: This macro can only be used in DDs modeling HART 6 or later, and prior to Common Practice rev 10.

To import these items, add the following line within the common practice device variable import section:

```
IMPORT_DEV_VARS_CATCH_V8()
```

The list of items imported using the IMPORT_DEV_VARS_CATCH_V8() macro is described in Table 16.

Table 16: IMPORT_DEV_VARS_CATCH_V8() items

Item Type	Item Name	Description
COMMAND	*catch_device_variable*	Command Definition for command 113.

Item Type	Item Name	Description
VARIABLE	*destination_device_variable_code*	Data referenced from commands 113 and 114.
COMMAND	*read_caught_device_variable*	Command Definition for command 114.

7.4.1.4 IMPORT_DEV_VARS_MAPPING_READ()

The IMPORT_DEV_VARS_MAPPING_READ() macro imports a list of all items needed to support the reading of device variable mapping.

To import these required tables, add the following line within the common practice device variables import section:

```
IMPORT_DEV_VARS_MAPPING_READ()
```

NOTE: If the IMPORT_DEV_VARS_MAPPING_READ() macro is used, do NOT use the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro.

NOTE: If mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), then do NOT use the IMPORT_DEV_VARS_MAPPING_READ() macro.

The list of items imported using the IMPORT_DEV_VARS_MAPPING_READ() macro is described in Table 17.

Table 17: IMPORT_DEV_VARS_MAPPING_READ() items

Item Type	Item Name	Description
VARIABLE	*primary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), primary_variable_code will be indirectly referenced from commands 1, 3, 8, 14, 15, 34, 35, 44, 49, 61, 63, 65, and 110. primary_variable_code is always referenced from commands 50 and 51.
VARIABLE	*quaternary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), quaternary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. quaternary_variable_code is always referenced from commands 50 and 51.
COMMAND	*read_device_variable_assignments*	Command Definition for command 50.
VARIABLE	*secondary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), secondary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. secondary_variable_code is always referenced from commands 50 and 51.
VARIABLE	*tertiary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), tertiary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. tertiary_variable_code is always referenced from commands 50 and 51. Data referenced from commands.

Item Type	Item Name	Description
METHOD	*warning_message*	POST_EDIT_ACTION METHOD used by variable mapping indexes.

7.4.1.5 IMPORT_DEV_VARS_MAPPING_READ_WRITE()

The IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro imports a list of all items needed to support the reading and writing of device variable mapping.

To import these required tables, add the following line within the common practice device variables import section:

```
IMPORT_DEV_VARS_MAPPING_READ_WRITE()
```

NOTE: If the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro is used, do NOT use the IMPORT_DEV_VARS_MAPPING_READ() macro.

NOTE: If mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), then do NOT use the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro.

The list of items imported using the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro is described in Table 18.

Table 18: IMPORT_DEV_VARS_MAPPING_READ_WRITE() items

Item Type	Item Name	Description
VARIABLE	*primary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), primary_variable_code will be indirectly referenced from commands 1, 3, 8, 14, 15, 34, 35, 44, 49, 61, 63, 65, and 110. primary_variable_code is always referenced from commands 50 and 51.
VARIABLE	*quaternary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), quaternary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. quaternary_variable_code is always referenced from commands 50 and 51.
COMMAND	*read_device_variable_assignments*	Command Definition for command 50.
VARIABLE	*secondary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), secondary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. secondary_variable_code is always referenced from commands 50 and 51.
VARIABLE	*tertiary_variable_code*	Unless the variable mapping has been redefined to fixed mapping (see 7.2.3.3 and 7.3.3.2), tertiary_variable_code will be indirectly referenced from commands 3, 8, 61, 63, 65, and 110. tertiary_variable_code is always referenced from commands 50 and 51. Data referenced from commands.

Item Type	Item Name	Description
METHOD	*warning_message*	POST_EDIT_ACTION METHOD used by variable mapping indexes.
COMMAND	*write_device_variable_assignments*	Command Definition for command 51.

7.4.1.6 IMPORT_DEV_VARS_TRIM()

The IMPORT_DEV_VARS_TRIM() macro imports all items needed to implement device variable trimming.

NOTE: This macro can only be used in DDs modeling HART 6 or later.

To import these required tables, add the following line within the common practice device variables import section:

```
IMPORT_DEV_VARS_TRIM()
```

The list of items imported using the IMPORT_DEV_VARS_TRIM() macro is described in Table 19.

Table 19: IMPORT_DEV_VARS_TRIM() items

Item Type	Item Name	Description
VARIABLE	*device_variable_trim_code*	Data referenced from commands 80, 81, 82, and 83.
COMMAND	*read_device_variable_trim_guidelines*	Command Definition for command 81.
COMMAND	*read_device_variable_trim_point*	Command Definition for command 80.
COMMAND	*reset_device_variable_trim*	Command Definition for command 83.
COMMAND	*write_device_variable_trim_point*	Command Definition for command 82.

7.4.1.7 Additional items

Some additional less often to be used tables are also included in the standard tables DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually. To import an individual item, add the item within the common practice device variable import section (example shown below):

```
COMMAND write_pv_units;
```

The list of additional items (not covered in any of the previous macros) is described in Table 20.

Table 20:Other Device Variable items

Item Type	Item Name	Description
METHOD	*catchDeviceVariable*	A reference METHOD for catching device variables from other devices. (Available with HART6 and later only)
METHOD	*device_variable_trim*	A reference METHOD using commands 80, 81, and 82 to trim device variables. (Available with HART6 and later only)
METHOD	*device_variable_trim_reset*	A reference METHOD using commands 80, and 83 to reset device variable trims. (Available with HART6 and later only)

Item Type	Item Name	Description
METHOD	*device_variable_zero_trim*	A reference METHOD using command 52 to zero device variables.
COMMAND	*read_all_dynamic_variables*	Deprecated Command Definition for command 110.
COMMAND	*read_device_variables*	Command Definition for command 33.
COMMAND	*read_device_variable_command_code*	Command Definition for command 534. (Available with Common Practice v12 or later only)
COMMAND	*read_dynamic_variables_and_pv_analog_output*	Deprecated Command Definition for command 61.
METHOD	*return_to_normal*	ABORT METHOD to remind users to put loop back to auto control.
COMMAND	*set_device_variable_zero*	Command Definition for command 52.
COMMAND	*set_pv_zero*	Command Definition for command 43.
METHOD	*trim_warning*	ABORT METHOD used by trim methods. (Available with HART6 and later only)
COMMAND	*write_device_variable*	Command Definition for command 79. (Available with HART6 and later only)
COMMAND	*write_device_variable_damping_value*	Command Definition for command 55.
COMMAND	*write_device_variable_sensor_serial_number*	Command Definition for command 56.
COMMAND	*write_device_variable_units*	Command Definition for command 53.
COMMAND	*write_pv_damping_value*	Command Definition for command 34.
COMMAND	*write_pv_sensor_serial_number*	Command Definition for command 49.
COMMAND	*write_pv_transfer_function*	Command Definition for command 47.
COMMAND	*write_pv_units*	Command Definition for command 44.
COMMAND	*write_volumetric_flow_classification*	Command Definition for command 522. (Available with common practice 11 or later only)
METHOD	*zero_trim*	A reference METHOD using command 43 to zero PV.

7.4.2 DEPENDENCIES

When using IMPORT_DEV_VARS_BASE() macro, the deviceVariable collection for all device variables must include the following members (see 6.9). These members may be the same or in addition to the members that are accessed from the universal base import (see 7.1.2):

- CONFIG_UNITS
- DAMPING_VALUE
- LOWER_SENSOR_LIMIT
- MINIMUM_SPAN
- SENSOR_SERIAL_NUMBER
- UPPER_SENSOR_LIMIT

If the universal revision is rev 6 or greater, the following members must also be included:

- CLASSIFICATION
- DEVICE_FAMILY

If the universal revision is rev 7 or greater, the following members must also be included:

- PROPERTIES
- UPDATE_TIME_PERIOD

When using IMPORT_DEV_VARS_TRIM() macro, the deviceVariable collection for all device variables supporting trim must include the following members (see 6.9):

- DIGITAL_VALUE
- LOWER_TRIM_POINT
- MAXIMUM_LOWER_TRIM_POINT
- MAXIMUM_UPPER_TRIM_POINT
- MINIMUM_LOWER_TRIM_POINT
- MINIMUM_TRIM_DIFFERENTIAL
- MINIMUM_UPPER_TRIM_POINT
- TRIM_POINT_SUPPORT
- TRIM_POINT_UNITS
- UPPER_TRIM_POINT

When using IMPORT_DEV_VARS_CATCH_V11()macro, the deviceVariable collection for all device variables that support the catch mechanism must include the following members (see 6.9):

- CAPTURE_MODE
- SHED_TIME
- SOURCE_COMMAND_NUMBER
- SOURCE_DEVICE_ID
- SOURCE_DEVICE_TYPE
- SOURCE_SLOT_NUMBER

When using IMPORT_DEV_VARS_CATCH_V8() or IMPORT_DEV_VARS_CATCH_V11() macro, the deviceVariable collection for all device variables that support the catch mechanism must include the following members (see 6.9):

- CAPTURE_MODE
- SHED_TIME
- SOURCE_COMMAND_NUMBER
- SOURCE_DEVICE_ID
- SOURCE_MANUFACTURER
- SOURCE_DEVICE_TYPE
- SOURCE_SLOT_NUMBER

When importing additional individual items (from list in 7.4.1.7), there can be dependencies on the developers deviceVariables collections. Certain members of the device variables' collections need to be present depending on the supported command numbers. Ensure that all applicable device variable attributes from Table 3 (reference the 'Associated Command Numbers' column) are present for imported items in Table 20 (reference the 'Description' column to find associated command numbers).

When manually importing COMMAND read_device_variables in a DD using the HART rev 5 model, the variables device_variable_code_1, device_variable_code_2, device_variable_code_3, and device_variable_code_4 must be

manually imported from the universal DD. HART rev 6 and 7 will already have these variables imported by using the IMPORT_UNIVERSAL_BASE(u) for values of u at 6 or greater.

7.4.3 REDEFINITIONS

7.4.3.1 Read-only Variable Mapping

When using the IMPORT_DEV_VARS_MAPPING_READ() macro, the primary_variable_code, secondary_variable_code, tertiary_variable_code, and quaternary_variable_code will need to be redefined to be read-only.

To redefine the mapping variables to be read-only, add the following lines to the redefinition section of the Common Practice Device variables DD import:

```
REDEFINITIONS
{
    VARIABLE primary_variable_code
    {
        REDEFINE HANDLING READ;
    }
    VARIABLE secondary_variable_code
    {
        REDEFINE HANDLING READ;
    }
    VARIABLE tertiary_variable_code
    {
        REDEFINE HANDLING READ;
    }
    VARIABLE quaternary_variable_code
    {
        REDEFINE HANDLING READ;
    }
}
```

7.4.3.2 Mappable Device Variables

When importing the variables primary_variable_code, secondary_variable_code, tertiary_variable_code, and quaternary_variable_code (i.e. if using the IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro) from the HART Common Practice Device Variables DD, the type INDEX may need to be redefined to point to only a subset of device variables that can be mapped to PV, SV, TV, or QV using command 51. If the device being modelled does not support mapping of ALL device variables contained in the deviceVariables array to PV, SV, TV, and/or QV, then a vendor specific array containing only the mappable variable elements should be defined, and the primary, secondary, tertiary, and quaternary variable codes should be redefined to index into it.

To redefine the variable mapping codes to point to a subset of device variables, add the following lines to the redefinition section of the Common Practice Device variables DD import:

```
REDEFINITIONS
{
    VARIABLE primary_variable_code
    {
        REDEFINE TYPE INDEX vendorSpecificMappingArray;
    }
}
```

The definition of the vendor specific mapping array defined in the vendor specific part of DD source may look similar to the following:

```
ARRAY OF COLLECTION vendorSpecificMappingArray
{
    ELEMENTS
    {
```

```
            0, var0, "Variable 0";
        }
    }
```

7.4.3.3 Trimmable device variables

When importing VARIABLE device_variable_trim_code (i.e. if using the IMPORT_DEV_VARS_TRIM() macro) from the HART Common Practice Device Variables DD, the type INDEX may need to be redefined to point to only a subset of device variables that can be trimmed using the common practice trim commands. If the device being modelled does not support trimming of ALL device variables contained in the deviceVariables array, then a vendor specific array containing only the trimmable variable elements should be defined, and the device_variable_trim_code should be redefined to index into it.

To redefine the device_variable_trim_code to point to a subset of device variables, add the following lines to the redefinition section of the Common Practice Device variables DD import:

```
REDEFINITIONS
{
    VARIABLE device_variable_trim_code
    {
        REDEFINE TYPE INDEX vendorSpecificTrimArray;
    }
}
```

The definition of the vendor specific trim array defined in the vendor specific part of DD source may look similar to the following:

```
ARRAY OF COLLECTION vendorSpecificTrimArray
{
    ELEMENTS
    {
        0, var0, "Variable 0";
    }
}
```

7.4.3.4 Catchable device variables

When importing VARIABLE destination_device_variable_code (i.e. if using the IMPORT_DEV_VARS_CATCH_V11() or IMPORT_DEV_VARS_CATCH_V8() macro) from the HART Common Practice Device Variables DD, the type INDEX may need to be redefined to point to only a subset of device variables that can be captured using the common practice “catch” commands. If the device being modelled does not support catching ALL device variables contained in the deviceVariables array, then a vendor specific array containing only the catchable variable elements should be defined, and the destination_device_variable_code should be redefined to index into it.

To redefine the destination_device_variable_code to point to a subset of device variables, add the following lines to the redefinition section of the Common Practice Device variables DD import:

```
REDEFINITIONS
{
    VARIABLE destination_device_variable_code
    {
        REDEFINE TYPE INDEX vendorSpecificCatchArray;
    }
}
```

The definition of the vendor specific catch array defined in the vendor specific part of DD source may look similar to the following:

```
ARRAY OF COLLECTION vendorSpecificCatchArray
{
    ELEMENTS
    {
```

```
            0, var0, "Variable 0";
        }
    }
```

7.4.3.5 Remove actions

The HART standard DDs include a warning message to be displayed as a POST_EDIT_ACTION for the variables primary_variable_code, secondary_variable_code, tertiary_variable_code, and quaternary_variable_code. If the device being modeled does not require any sort of user warning (i.e. because the user is warned in another way or changing the variable mapping does not constitute any warning message to be displayed), then the actions can be deleted from these variables.

To delete the post edit actions on the mapping variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE primary_variable_code
    {
        DELETE POST_EDIT_ACTIONS;
    }
}
```

7.4.3.6 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter ‘x’ is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE primary_variable_code
    {
        SET_DEFAULT(0);
    }
}
```

7.4.3.7 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD device_variable_trim_reset
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.4.3.8 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on

the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE destination_device_variable_code
    {
        REDEFINE HELP "Destination Device Variable- An index that identifies a Field Device
Variable that supports the capturing of a device variable from another field device. For this
device, the destination device variable can be set to either 0 for external temperature or 1
for external pressure.";
    }
}
```

7.5 HART Common Practice Analog Channels DD

The Common Practice Analog Channels DD includes items necessary support analog channels in a field device. The common practice analog channel binary files are installed in \HCF\DDL\Library\000000\001a\0706.*, 0804.*, 0904.*, 0b02.*, and 0c02.*. The corresponding DDL source code for these files is located in HCF\DDL\HCFinfo\Standards\000000\Common7_6_Analog_Channels.ddl, Common8_4_Analog_Channels.ddl, Common9_4_Analog_Channels.ddl, Common11_2_Analog_Channels.ddl, and Common12_2_Analog_Channels.ddl.

Only one device revision of the DD (either 7, 8, 9, 11, or 12) should be imported into a device DD depending on which HART universal revision is being modelled. To import the common practice analog channel DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 5 */
IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 7, DD_REVISION 6
{
    IMPORT_ANALOG_CHANNELS_MULTI_AO()
}

/* For HART Rev 6 */
IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 8, DD_REVISION 4
{
    IMPORT_ANALOG_CHANNELS_MULTI_AO()
}

/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_ANALOG_CHANNELS_MULTI_AO()
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_ANALOG_CHANNELS_MULTI_AO()
}

/* For HART Rev 7, Common Practice 12 */
IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 12, DD_REVISION 2
{
    IMPORT_ANALOG_CHANNELS_MULTI_AO()
}
```

Import macros have been provided to make it easier to extract the correct items from the Common Practice Analog Channels DD. A description of each of these macros is described in 7.5.1.

7.5.1 IMPORTS

7.5.1.1 IMPORT_ANALOG_CHANNELS_MULTI_AO()

The IMPORT_ANALOG_CHANNELS_MULTI_AO() macro imports all items required to support multiple analog signaling channels.

To import these items, add the following line within the common practice analog channels import section:

```
IMPORT_ANALOG_CHANNELS_MULTI_AO()
```

The list of items imported using the IMPORT_ANALOG_CHANNELS_MULTI_AO() macro is described in Table 21.

Table 21: IMPORT_ANALOG_CHANNELS_MULTI_AO() items

Item Type	Item Name	Description
VARIABLE	*analog_channel_number_code*	Data referenced from commands 60, 63, 64, 65, 66, 67, 68, 69, 70, and 535.
VARIABLE	*analog_channel_number_code_1*	Data referenced from command 62.
VARIABLE	*analog_channel_number_code_2*	Data referenced from command 62.
VARIABLE	*analog_channel_number_code_3*	Data referenced from command 62.
VARIABLE	*analog_channel_number_code_4*	Data referenced from command 62.
COMMAND	*enter_exit_fixed_analog_channel_mode*	Command Definition for command 66.
COMMAND	*read_analog_channel_endpoint_values*	Command Definition for command 70.
COMMAND	*read_analog_channel_information*	Command Definition for command 63.
COMMAND	*read_analog_channels*	Command Definition for command 62.
COMMAND	*read_analog_output_and_percent_of_range*	Command Definition for command 60.
COMMAND	*trim_analog_channel_gain*	Command Definition for command 68.
COMMAND	*trim_analog_channel_zero*	Command Definition for command 67.
COMMAND	*write_analog_channel_additional_damping_value*	Command Definition for command 64.
COMMAND	*write_analog_channel_range_values*	Command Definition for command 65.
COMMAND	*write_analog_channel_transfer_function*	Command Definition for command 69.

7.5.1.2 Additional items

Some additional items are also included in the common practice analog channels DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually.

To import an individual item, add the item within the common practice analog channels import section (example shown below):

```
COMMAND enter_exit_fixed_pv_current_mode;
```

The list of additional items (not covered in any of the previous macros) is described in Table 22.

Table 22: Other Analog Channel items

Item Type	Item Name	Description
METHOD	*analog_dac_trim*	METHOD for trimming an Analog Channel using commands 66, 67, 68.
METHOD	*applied_rerange*	METHOD for adjusting the range values using commands 1, 15, 36, and 37.
COMMAND	*enter_exit_fixed_pv_current_mode*	Command Definition for command 40.
METHOD	*fixed_analog_output*	METHOD for manually controlling loop output level using command 66.
METHOD	*leave_analog_fixed_current_mode*	ABORT METHOD used by fixed_analog_output METHOD.
METHOD	*leave_fixed_current_mode*	ABORT METHOD used by enter_exit_fixed_pv_current_mode METHOD.
COMMAND	*set_pv_lower_range_value*	Command Definition for command 37.

Item Type	Item Name	Description
COMMAND	*set_pv_upper_range_value*	Command Definition for command 36.
METHOD	*transmitter_dac_trim*	METHOD for trimming the PV loop current using commands 40, 45, and 46.
METHOD	*transmitter_loop_test*	METHOD for validating loop current output using command 40.
COMMAND	*trim_pv_current_dac_gain*	Command Definition for command 46.
COMMAND	*trim_pv_current_dac_zero*	Command Definition for command 45.
METHOD	*valve_adc_trim*	METHOD for trimming loop input using commands 2, 45, and 46.
COMMAND	*write_analog_channel_endpoint_values*	Command Definition for command 535.
COMMAND	*write_pv_alarm_code*	Command Definition for command 100. (Available with Common 11 or later only)
COMMAND	*write_pv_range_values*	Command Definition for command 35.

7.5.2 DEPENDENCIES

When using the Common Practice Analog Channels DD, the PV2 DD must also be imported. See (7.3).

When using the IMPORT_ANALOG_CHANNELS_MULTI_AO() import, there is a dependency on standard tables:

- IMPORT_TABLES_MULTI_AO() Described in 7.13.1.17

7.5.3 REDEFINITIONS

7.5.3.1 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD analog_dac_trim
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.5.3.2 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    METHOD analog_dac_trim
    {
        REDEFINE HELP "Trim Analog Output-  Allows the Digital to Analog calibration of a
selected Analog Channel with an external reference at its operating endpoints. For this
device, the calibration performed at the factory can be recalled by navigating to the
maintenance menu.";
    }
}
```

7.6 HART Common Practice Condensed Status DD

The Common Practice Condensed Status DD includes items necessary to model the condensed status functionality. The Common Practice Condensed Status DD binary files are installed in \HCF\DDL\Library\000000\001d\0b02.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Common11_2_Condensed_Status.ddl.

NOTE: This DD can only be imported when modeling HART Universal Revision 7, Common Practice 10 or later. HART revisions 5 and 6 do not have support for condensed status commands.

To import the common practice condensed status DD into a product DD, use the following IMPORT statement:

```
IMPORT STANDARD _COMMON_PRACTICE_CONDENSED_STATUS, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_CONDENSED_STATUS_BASE()
}
```

Import macros have been provided to make it easier to extract the correct items from the Common Practice Condensed Status DD. A description of each of these macros is described in 7.6.1.

7.6.1 IMPORTS

7.6.1.1 IMPORT_CONDENSED_STATUS_BASE()

The IMPORT_CONDENSED_STATUS_BASE() macro imports items required to model the condensed status functionality.

To import these items, add the following line within the common practice condensed status import section:

```
IMPORT_CONDENSED_STATUS_BASE()
```

The list of items imported using the IMPORT_CONDENSED_STATUS_BASE() macro is described in Table 23.

Table 23: IMPORT_CONDENSED_STATUS_BASE() items

Item Type	Item Name	Description
COMMAND	*read_condensed_status_map*	Command Definition for command 523.
COMMAND	*reset_condensed_status_map*	Command Definition for command 525.
METHOD	*Reset_Condensed_Status_Map*	METHOD to reset condensed status mapping back to default values.
VARIABLE	*simulate_bit_index*	Local variable referenced from command 527.
COMMAND	*simulate_status_bit*	Command Definition for command 527.
COMMAND	*write_condensed_status_map*	Command Definition for command 524.
COMMAND	*write_status_simulation_mode*	Command Definition for command 526.

7.6.1.2 IMPORT_CONDENSED_STATUS_METHOD_SIMULATE()

The CONDENSED_STATUS_METHOD_SIMULATE() macro imports items to support an example METHOD that allows the generic simulation of condensed status.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice condensed status import section:

```
IMPORT_CONDENSED_STATUS_METHOD_SIMULATE()
```

The list of items imported using the IMPORT_CONDENSED_STATUS_METHOD_SIMULATE() macro is described in Table 24.

Table 24: IMPORT_CONDENSED_STATUS_METHOD_SIMULATE() items

Item Type	Item Name	Description
METHOD	*abort_disable_status_simulation*	Abort method used by the set_status_simulation method.
METHOD	*set_status_simulation*	METHOD to simulate condensed status.
ARRAY OF VARIABLE	*simulation_items*	Data referenced from the set_status_simulation method.
VARIABLE	*status_simulation_byte*	Local variable referenced from the set_status_simulation method.
VARIABLE	*userdefined_label_matches*	Local variable referenced from the set_status_simulation method.

7.6.2 DEPENDENCIES

When using the IMPORT_CONDENSED_STATUS_BASE() import, there is a dependency on standard tables:

- IMPORT_TABLES_CONDENSED_STATUS(n) Described in 7.13.1.4

7.6.3 REDEFINITIONS

7.6.3.1 REDEFINE_CONDENSED_STATUS_TRANSACTIONS(n)

The REDEFINE_CONDENSED_STATUS_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Condensed Status import. It is used to truncate the number of condensed status mapping parameters to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_CONDENSED_STATUS_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the command 523 and 524 condensed status mapping command transactions to match the device, add the following lines within the Condensed Status import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    REDEFINE_CONDENSED_STATUS_TRANSACTIONS(10)
}
```

7.6.3.2 REDEFINE_SIMULATION_ITEM_SIZE(n)

The REDEFINE_SIMULATION_ITEM_SIZE(n) macro is used in the REDEFINITIONS section of the Common Practice Condensed Status import. It is used to truncate the number of condensed status mapping parameters to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_SIMULATION_ITEM_SIZE(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the condensed status mapping array to match the device, add the following lines within the Condensed Status import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    ARRAY OF VARIABLE simulation_items
    {
        REDEFINE_SIMULATION_ITEM_SIZE(10)
    }
}
```

7.6.3.3 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD Reset_Condensed_Status_Map
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.6.3.4 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    METHOD set_status_simulation
    {
        REDEFINE HELP "Set Status Simulation- Allows detailed status of the device to be
simulated, so that the condensed status mapping configuration can be verified for desired
operation. For this product, the simulated status will also affect the LEDs on the local
panel.";
    }
}
```

7.7 HART Common Practice Burst Value Array DD

The Common Practice Burst Value Array DD includes all items needed to implement the burst mode functionality. Burst mode can be modelled using either value arrays or reference arrays. This DD implements the value array model, which is recommended because it allows any number of burst messages to be modelled without creating unique variable copies for each supported burst message. Simply changing the number of elements in the value array can change how many unique burst messages are being modelled in the DD.

The Common Practice Burst Value Array binary files are installed in \HCF\DDL\Library\000000\001b\0706.*, 0804.*, 0904.*, 0b02.*, and 0c02.*. The corresponding DDL source code for these files are located in HCF\DDL\HCFinfo\Standards\000000\Common7_6_Burst_Val_Array.ddl, Common8_4_Burst_Val_Array.ddl, Common9_4_Burst_Val_Array.ddl, Common11_2_Burst_Val_Array.ddl, and Common12_2_Burst_Val_Array.ddl.

Only one device revision of the DD (either 7, 8, 9, 11, or 12) should be imported into a device DD depending on which HART universal revision is being modelled. To import the common practice burst value array DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 5 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_VAL_ARRAY, DEVICE_REVISION 7, DD_REVISION 6
{
    IMPORT_BURST_VAL_ARRAY_BASE(5)
}

/* For HART Rev 6 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_VAL_ARRAY, DEVICE_REVISION 8, DD_REVISION 4
{
    IMPORT_BURST_VAL_ARRAY_BASE(6)
}

/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_VAL_ARRAY, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_BURST_VAL_ARRAY_BASE(7)
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_VAL_ARRAY, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_BURST_VAL_ARRAY_BASE(7)
}

/* For HART Rev 7, Common Practice 12 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_VAL_ARRAY, DEVICE_REVISION 12, DD_REVISION 2
{
    IMPORT_BURST_VAL_ARRAY_BASE(7)
}
```

NOTE: If this DD is imported, do NOT import the Common Practice Burst Reference Array DD.

Import macros have been provided to make it easier to extract the correct items from the HART Common Practice Burst Value Array DD. A description of each of these macros is described in 7.7.1.

7.7.1 IMPORTS

7.7.1.1 IMPORT_BURST_VAL_ARRAY_BASE(u)

The IMPORT_BURST_VAL_ARRAY_BASE(u) macro imports a list of all items required to implement burst mode configuration. Different sets of variables are necessary depending on the HART universal revision that the DD is modeling. The parameter “u” is used to specify which UNIVERSAL revision items to import. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used.

To import these required items, add one of the following lines within the common practice burst value array import section:

```
IMPORT_BURST_VAL_ARRAY_BASE(5)
IMPORT_BURST_VAL_ARRAY_BASE(6)
IMPORT_BURST_VAL_ARRAY_BASE(7)
```

The list of items imported using the IMPORT_BURST_VAL_ARRAY_BASE(u) macro is described in Table 25. The Universal Revisions column indicates which items are imported for each of the HART universal revision macros.

Table 25: IMPORT_BURST_VAL_ARRAY_BASE(u) items

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*burst_command_number*	5, 6, 7	Data referenced from commands 105 and 108.
COLLECTION	*std_burst_message_collection*	5, 6, 7	Referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
VARIABLE	*burst_message_number*	7	Data referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
ARRAY	*burst_messages*	5, 6, 7	Data referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
COMMAND	*burst_mode_control*	5, 6, 7	Command Definition for command 109.
VARIABLE	*burst_variable_code*	5, 6, 7	Data referenced from commands 105 and 107.
ARRAY	*burst_variable_codes*	5, 6, 7	Data referenced from commands 105 and 107.
ARRAY OF COLLECTION	*burstDeviceVariables*	5, 6, 7	Data Type used indirectly for commands 105 and 107.
COLLECTION	*dummy_collection*	5, 6, 7	Data Type used indirectly for commands 105 and 107.
VARIABLE	*dummy_variable*	5, 6, 7	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_float*	5, 6, 7	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_time_int*	7	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_unit*	5, 6, 7	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_unsigned_3*	5, 6, 7	Stub variable used for burst indexing type.
VARIABLE	*max_update_period*	7	Data Type used indirectly for commands 103 and 105.
COMMAND	*read_burst_mode_configuration*	6, 7	Command Definition for command 105.
VARIABLE	*total_number_burst_messages*	7	Data referenced from command 105.
VARIABLE	*trigger_level*	7	Data referenced from commands 104 and 105.
VARIABLE	*update_period*	7	Data referenced from commands 103 and 105.
COMMAND	*write_burst_command_number*	5, 6, 7	Command Definition for command 108.
COMMAND	*write_burst_device_variables*	5, 6, 7	Command Definition for command 107.
COMMAND	*write_burst_period*	7	Command Definition for command 103.
COMMAND	*write_burst_trigger*	7	Command Definition for command 104.

7.7.1.2 IMPORT_BURST_VAL_ARRAY_EVENT()

The IMPORT_BURST_VAL_ARRAY_EVENT() macro imports a list of all items required to implement event notification configuration.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these required items, add the following line within the common practice burst value array import section:

```
IMPORT_BURST_VAL_ARRAY_EVENT()
```

The list of items imported using the IMPORT_BURST_VAL_ARRAY_EVENT() macro is described in Table 26.

Table 26: IMPORT_BURST_VAL_ARRAY_EVENT() items

Item Type	Item Name	Description
COMMAND	*acknowledge_event_notification*	Command Definition for command 119.
COLLECTION	*event*	Referenced from commands 101, 102, 115, 116, 117, 118 and 119.
COLLECTION	*event_control*	Referenced from commands 101, 102, 115, 117, 118 and 119.
VARIABLE	*event_debounce_interval_event*	Data referenced from commands 115 and 117.
COLLECTION	*event_mask*	Referenced from commands 115 and 116.
COMMAND	*event_notification_control*	Command Definition for command 118.
VARIABLE	*event_notification_retry_time_event*	Data referenced from commands 115 and 117.
COLLECTION	*event_report*	Referenced from command 119.
VARIABLE	*eventNumber*	Data referenced from commands 101, 102, 115, 116, 117, 118 and 119.
ARRAY	*events*	Data referenced from commands 101, 102, 115, 116, 117, 118 and 119.
VARIABLE	*max_update_time_event*	Data referenced from commands 115 and 117.
VARIABLE	*number_events_supported*	Data referenced from command 115.
COMMAND	*read_event_notification_summary*	Command Definition for command 115.
VARIABLE	*time_of_first_unack_event*	Data referenced from commands 115 and 119.
COMMAND	*write_event_notification_bit_mask*	Command Definition for command 116.
COMMAND	*write_event_notification_timing*	Command Definition for command 117.

7.7.1.3 IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER()

The IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER() macro imports items that are required to implement Event system registration.

NOTE: This macro can only be used in DDs modeling HART rev 7 and Common Practice rev 10 or later.

To import these items, add the following line within the common practice burst value array import section:

```
IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER()
```

The list of items imported using the IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER() macro is described in Table 27.

Table 27: IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER() items

Item Type	Item Name	Description
COMMAND	*read_event_manager_registration_status*	Command Definition for command 515.
COMMAND	*register_event_manager*	Command Definition for command 514.

7.7.1.4 IMPORT_BURST_VAL_ARRAY_SUBDEVICE()

The IMPORT_BURST_VAL_ARRAY_SUBDEVICE() macro imports items necessary to support IO Adapter bursting.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst value array import section:

```
IMPORT_BURST_VAL_ARRAY_SUBDEVICE()
```

The list of items imported using the IMPORT_BURST_VAL_ARRAY_SUBDEVICE() macro is described in Table 28.

Table 28: IMPORT_BURST_VAL_ARRAY_SUBDEVICE() items

Item Type	Item Name	Description
VARIABLE	*burst_subdevice_index*	Data referenced from commands 101 and 102.
VARIABLE	*constOne*	Data referenced from commands 101 and 102.
VARIABLE	*constZero*	Data referenced from commands 101 and 102.
VARIABLE	*event_subdevice_index*	Data referenced from commands 101 and 102.
COMMAND	*read_subdevice_burst_map*	Command Definition for command 101.
VARIABLE	*subdevice_missing*	Data referenced from commands 101 and 102.
COMMAND	*write_subdevice_burst_map*	Command Definition for command 102.

7.7.1.5 IMPORT_BURST_VAL_ARRAY_TREND()

The IMPORT_BURST_VAL_ARRAY_TREND() macro imports all items required to implement device variable trending.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst value array import section:

```
IMPORT_BURST_VAL_ARRAY_TREND()
```

The list of items imported using the IMPORT_BURST_VAL_ARRAY_TREND() macro is described in Table 29.

Table 29: IMPORT_BURST_VAL_ARRAY_TREND() items

Item Type	Item Name	Description
VARIABLE	*number_trends_supported*	Data referenced from command 91.
COMMAND	*read_trend*	Command Definition for command 93.

Item Type	Item Name	Description
COMMAND	*read_trend_configuration*	Command Definition for command 91.
METHOD	*readTrendFromDevice*	Method that sends command 93 to read the trend values.
COLLECTION	*std_trend_collection*	Referenced from commands 91, 92 and 93.
VARIABLE	*trend_0_date_stamp*	Data referenced from command 93.
VARIABLE	*trend_0_time_stamp*	Data referenced from command 93.
ARRAY	*trend_array*	Data referenced from commands 91, 92 and 93.
COLLECTION	*trend_data*	Referenced from command 93.
VARIABLE	*trend_device_variable_code*	Data referenced from commands 91, 92 and 93.
VARIABLE	*trend_number*	Data referenced from commands 91, 92 and 93.
VARIABLE	*trend_sample_interval*	Data referenced from commands 91, 92 and 93.
VARIABLE	*trend_value*	Data referenced from command 93.
ARRAY	*trend_values*	Data referenced from command 93.
COMMAND	*write_trend_configuration*	Command Definition for command 92.

7.7.2 DEPENDENCIES

When using the IMPORT_BURST_VAL_ARRAY_BASE(u) import, there is a dependency on standard tables:

- IMPORT_TABLES_BURST_VAL_ARRAY(u) Described in 7.13.1.3

When using the IMPORT_BURST_VAL_ARRAY_EVENT() import, there is a dependency on standard tables:

- IMPORT_TABLES_EVENT_VAL_ARRAY() Described in 7.13.1.10

When using the IMPORT_BURST_VAL_ARRAY_EVENT_REGISTER() import, there is a dependency on standard tables:

- IMPORT_TABLES_EVENT_MANAGER() Described in 7.13.1.8

When using the IMPORT_BURST_VAL_ARRAY_TREND() import, there is a dependency on standard tables:

- IMPORT_TABLES_TREND_VAL_ARRAY() Described in 7.13.1.24

7.7.3 REDEFINITIONS

7.7.3.1 REDEFINE_BURST_VAL_ARRAY_SIZE(b)

The REDEFINE_BURST_VAL_ARRAY_SIZE(b) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to adjust the number of burst messages supported in the corresponding device. When using the REDEFINE_BURST_VAL_ARRAY_SIZE(b) macro, the parameter ‘b’ is used to specify the number of burst messages to be implemented. The same value of b has to be used in all macros that are used that call for a parameter ‘b’ to be used.

To redefine the burst value array to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 3 burst messages is shown in this example):

```
REDEFINITIONS
{
    ARRAY burst_messages
    {
        REDEFINE_BURST_VAL_ARRAY_SIZE(3)
    }
}
```

7.7.3.2 REDEFINE_EVENT_VAL_ARRAY_SIZE(e)

The REDEFINE_EVENT_VAL_ARRAY_SIZE(e) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to adjust the number of event notification messages supported in the corresponding device. When using the REDEFINE_EVENT_VAL_ARRAY_SIZE(e) macro, the parameter 'e' is used to specify the number of event notification messages to be implemented. The same value of e has to be used in all macros that are used that call for a parameter 'e' to be used.

To redefine the event value array to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 1 event notification message is shown in this example):

```
REDEFINITIONS
{
    ARRAY events
    {
        REDEFINE_EVENT_VAL_ARRAY_SIZE(1)
    }
}
```

7.7.3.3 REDEFINE_EVENT_MASK_SIZE(n)

The REDEFINE_EVENT_MASK_SIZE(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to adjust the number of event notification mask bytes supported in the corresponding device. When using the REDEFINE_EVENT_MASK_SIZE(n) macro, the parameter 'n' is used to specify the number of status bytes implemented in command 48 (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the event mask size to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 10 command 48 bytes is shown in this example):

```
REDEFINITIONS
{
    COLLECTION event_mask
    {
        REDEFINE_EVENT_MASK_SIZE(10)
    }
}
```

7.7.3.4 REDEFINE_EVENT_REPORT_SIZE(n)

The REDEFINE_EVENT_REPORT_SIZE(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to adjust the number of event notification report bytes supported in the corresponding device. When using the REDEFINE_EVENT_REPORT_SIZE(n) macro, the parameter 'n' is used to specify the number of status bytes implemented in command 48 (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the event report size to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 10 command 48 bytes is shown in this example):

```
REDEFINITIONS
{
    COLLECTION event_report
    {
        REDEFINE_EVENT_REPORT_SIZE(10)
    }
}
```

7.7.3.5 REDEFINE_COMMAND_115_VAL_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_115_VAL_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to truncate the number of event notification status masks to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_115_VAL_ARRAY_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the command 115 command transactions to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_event_notification_summary
    {
        REDEFINE_COMMAND_115_VAL_ARRAY_TRANSACTIONS(10)
    }
}
```

7.7.3.6 REDEFINE_COMMAND_116_VAL_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_116_VAL_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to truncate the number of event notification status masks to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_116_VAL_ARRAY_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the command 116 command transactions to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND write_event_notification_bit_mask
    {
        REDEFINE_COMMAND_116_VAL_ARRAY_TRANSACTIONS(10)
    }
}
```

7.7.3.7 REDEFINE_COMMAND_119_VAL_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_119_VAL_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to truncate the number of event notification status bytes to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_119_VAL_ARRAY_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the command 119 command transactions to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND acknowledge_event_notification
    {
        REDEFINE_COMMAND_119_VAL_ARRAY_TRANSACTIONS(10)
    }
}
```

7.7.3.8 REDEFINE_TREND_VAL_ARRAY_SIZE(t)

The REDEFINE_TREND_VAL_ARRAY_SIZE(t) macro is used in the REDEFINITIONS section of the Common Practice Burst Value Array import. It is used to adjust the number of variable trends supported in the corresponding device. When using the REDEFINE_TREND_VAL_ARRAY_SIZE(t) macro, the parameter 't' is used to specify the number of variable trends to be implemented. The same value of t has to be used in all macros that are used that call for a parameter 't' to be used.

To redefine the trend value array to match the device, add the following lines within the Burst Value Array import redefinition section (Note – A device implementing 2 burst messages is shown in this example):

```
REDEFINITIONS
{
    ARRAY trend_array
    {
        REDEFINE_TREND_VAL_ARRAY_SIZE(2)
    }
}
```

7.7.3.9 ARRAY OF COLLECTION burstDeviceVariables

The burstDeviceVariables array imported for HART5 and HART6 devices only contains the variable code 250 "Not Used". This variable code must be supported in the burst variable slots and cannot be deleted. However, the burstDeviceVariables array imported for HART 7 contains all the standardized HART variable codes from 242 through 250. Some devices may not support all the standardized device variable codes (e.g. 242 and 243 for battery voltage and battery life), so those codes may need to be deleted from the burstDeviceVariables array. Additionally, when importing the burstDeviceVariables ARRAY from the HART common practice Burst Value Array DD, the array needs to be redefined to include all the device specific variables that can be bursted from the device.

There may also be a need to affect the ordering of burst device variables in the list (so the end user sees the enumerated selections appear in a certain order). ADDing enumerations causes them to be placed at the bottom of the list. To adjust the order of enumerated items, it may be necessary to DELETE some existing enumerations, and then ADD them back in at the right order in th list. The example below shows how device specific variables can be added to the list to appear before the standardized variables that are already in the list.

NOTE: The HART protocol specification states that a field device will return a failure response code if an attempt is made to write the burst slot#0 to 250 – "Not Used". However, the HART standard DD treats all 8 burst slots the same, as implementing the first slot to have a different set of enumerations compared to the other slots adds a layer of complexity, and may cause some compatibility issues for field devices developed prior to the addition of this protocol requirement. End users will unlikely intentionally set the first slot to 250 anyway – but if they do, some field devices will return an error so the user can change their selection and try again. No attempt should be made to remove code 250 from the list of burst slots because it should be possible to select for any one or more of the last 7 burst slots. Some DD developers may choose to add an action to warn the user if slot #0 is set to '250'.

To redefine the burstDeviceVariables array, add lines similar to the following lines to the redefinition section of the Common Practice Burst Value Array DD import:

```
REDEFINITIONS
{
    ARRAY OF COLLECTION burstDeviceVariables
    {
        ELEMENTS
        {
            DELETE 242; //Deleted because not supported
            DELETE 243; //Deleted because not supported
            DELETE 244; //Add later to preserve order
            DELETE 245; //Add later to preserve order
            DELETE 246; //Add later to preserve order
            DELETE 247; //Add later to preserve order
            DELETE 248; //Add later to preserve order
            DELETE 249; //Add later to preserve order
            DELETE 250; //Add later to preserve order
            ADD 0, deviceVariables[0], "Pressure";
            ADD 1, deviceVariables[1], "Temperature";
            ...
            ADD {244, dummy_collection, device_variable_code_codes(244); // Reorder
            ADD {245, dummy_collection, device_variable_code_codes(245); // Reorder
            ADD {246, dummy_collection, device_variable_code_codes(246); // Reorder
            ADD {247, dummy_collection, device_variable_code_codes(247); // Reorder
            ADD {248, dummy_collection, device_variable_code_codes(248); // Reorder
            ADD {249, dummy_collection, device_variable_code_codes(249); // Reorder
            ADD {250, dummy_collection, device_variable_code_codes(250); // Reorder
        }
    }
}
```

Some devices may need to declare a conditional set of burst slot variables based on varying support for different device variables based on certain device operating conditions. The example below shows how conditional support of variables can be defined to appear in the list.

To redefine the burstDeviceVariables array to have conditional items, add lines similar to the following lines to the redefinition section of the Common Practice Burst Value Array DD import:

```
REDEFINITIONS
{
    ARRAY OF COLLECTION burstDeviceVariables
    {
        REDEFINE ELEMENTS
        {
            0, deviceVariables[0], “devVar 0”;
            IF (condition)
            {
                1, deviceVariables[1],”devVar 1”;
            }
        }
    }
}
```

7.7.3.10 HART5 Write-only

In the HART revision 5 protocol specification, there is not a command to read back burst mode settings. Because of this, the variable burst_mode_select should have its HANDLING redefined to be write-only as follows:

```
REDEFINITIONS
{
    VARIABLE burst_mode_select
    {
        REDEFINE HANDLING WRITE;
```

```
        }
    }
```

7.7.3.11 Vendor-specific burst command list

When importing the burst command number variables, the list of command numbers in the enumeration is defined depending on which universal revision was used in the import macro as follows:

HART 7: Commands, 1, 2, 3, 9, and 48.

HART 6: Commands 1, 2, 3, 9.

HART 5: Commands 1, 2, 3.

In some cases, a device may allow for bursting of additional HART commands. In this case, the burst command number variables need to add the additional supported choices.

There may also be a need to affect the ordering of burst messages in the list (so the end user sees the enumerated selections appear in a certain order). ADDing enumerations causes them to be placed at the bottom of the list. To adjust the order of enumerated items, it may be necessary to DELETE some existing enumerations, and then ADD them back in at the right order in the list. The example below shows how optional command 33 can be added to the list to appear before command 48 that is already in the list.

To redefine the burst command number variables to support additional HART commands, add lines similar to the following to the Common Practice Burst Value Array redefinition section:

```
REDEFINITIONS
{
    VARIABLE burst_command_number
    {
        TYPE ENUMERATED
        {
            DELETE 48;
            ADD {33, Read_Device_Variables_temp,     Read_Device_Variables_help_temp }
            ADD {48, read_additional_device_status_temp,
                     read_additional_device_status_help_temp}
            ADD {178, "My Special Burst Command",     "My Help"}
        }
    }
}
```

7.7.3.12 Burst and Event Notification Message Collections

When importing std_burst_message_collection, and event_control collections from the HART common practice Burst Value Array DD, the collections need to be redefined to remove IO system specific members if the DD is not being developed for an IO System.

All devices that don't connect to subdevices (are not IO Systems) should have the following lines in the redefinition section of the Common Practice Burst Value Array DD import:

```
REDEFINITIONS
{
    COLLECTION std_burst_message_collection
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }
```

```
COLLECTION event_control
{
    MEMBERS
    {
        DELETE SUB_DEVICE_MISSING;
        DELETE SUB_DEVICE_MAPPING;
    }
}
```

7.7.3.13 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE update_period
    {
        SET_DEFAULT(192000);
    }
}
```

7.7.3.14 Minimum and Maximum Values

It is best practice for all numeric values that are writeable to define an upper and lower limit. This helps to prevent an out of bounds entry from being attempted to be sent to a field device.

To define minimum and maximum values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE trigger_level
    {
        REDEFINE MIN_VALUE -200.0;
        REDEFINE MAX_VALUE 850.0;
    }
}
```

7.7.3.15 Display, Edit, and Time Formats

Depending on the measurement types and capabilities of the field device, the display format, edit format, and/or time format of certain variables may need to be adjusted to match characteristics of the field device. Variables such as trigger_level, trend_value, and others may need to have their display, edit, and/or time formats adjusted.

To redefine the display, edit, or time format for a variable, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE trigger_level
    {
        TYPE FLOAT
        {
            REDEFINE DISPLAY_FORMAT ".2f";
            REDEFINE EDIT_FORMAT ".2f";
        }
    }
```

```
    VARIABLE trend_0_time_stamp
    {
        TYPE TIME_VALUE
        {
            REDEFINE TIME_FORMAT %H:%M";
        }
    }
}
```

7.7.3.16 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD readTrendFromDevice
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.7.3.17 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE burst_variable_code
    {
        REDEFINE HELP "Burst Variable Slot- Device variable code assigned to the slot to be
read in burst mode. For this device, variable code setting of 0 is for the main pressure
value, and variable code setting of 1 is for the external temperature value.";
    }
}
```

7.8 HART Common Practice Burst Reference Array DD

The Common Practice Burst Reference Array DD includes all items needed to implement the burst mode functionality. Burst mode can be modelled using either value arrays or reference arrays. This DD implements the reference array model, which is limited to a maximum of 4 burst messages and 2 event notifications. The reference array model may be preferred for compatibility in older host applications.

The Common Practice Burst Reference Array binary files are installed in \HCF\DDL\Library\000000\001c\0706.*, 0804.*, 0904.*, 0b02.*, and 0c02.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Common7_6_Burst_Ref_Array.ddl, Common8_4_Burst_Ref_Array.ddl, Common9_4_Burst_Ref_Array.ddl, Common11_2_Burst_Ref_Array.ddl, and Common12_2_Burst_Ref_Array.ddl.

Only one device revision of the DD (either 7, 8, 9, 11, or 12) should be imported into a device DD depending on which HART universal revision is being modelled. To import the common practice burst reference array DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 5 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 7, DD_REVISION 6
{
    IMPORT_BURST_REF_ARRAY_BASE(5,1)
}

/* For HART Rev 6 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 8, DD_REVISION 4
{
    IMPORT_BURST_REF_ARRAY_BASE(6,1)
}

/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_BURST_REF_ARRAY_BASE(7,3)
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_BURST_REF_ARRAY_BASE(7,3)
}

/* For HART Rev 7, Common Practice 12 */
IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 12, DD_REVISION 2
{
    IMPORT_BURST_REF_ARRAY_BASE(7,3)
}
```

NOTE: If this DD is imported, do NOT import the Common Practice Burst Value Array DD.

Import macros have been provided to make it easier to extract the correct items from the HART Common Practice Burst Reference Array DD. A description of each of these macros is described in 7.8.1.

7.8.1 IMPORTS

7.8.1.1 IMPORT_BURST_REF_ARRAY_BASE(u, *b*)

The IMPORT_BURST_REF_ARRAY_BASE(u, b) macro imports a list of all items required to implement burst mode configuration. Different sets of burst related items are necessary depending on the HART universal revision that the DD is modeling. The parameter ‘u’ is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used. This macro imports the reference array model, which is limited to a maximum of 4 burst messages. The parameter ‘b’ is used

to specify how many burst messages are to be supported (allowable values are from 1 to 4). The same value of b has to be used in all macros that are used that call for a parameter 'b' to be used.

NOTE: When Universal Revision (u) is less than 7, only 1 burst message (b) is allowed to be modeled.

To import these required items, add the following line within the common practice burst reference array import section (Note – the example shows an import for a HART7 device that supports 3 total burst messages (index 0, 1, and 2):

```
IMPORT_BURST_REF_ARRAY_BASE(7,3)
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_BASE(u, b) macro are described in Table 30. The "Universal Revisions" column identifies for which value of u a variable will be imported or not. The "b Values" column identifies for which values of 'b' a variable will be imported or not.

Table 30: IMPORT_BURST_REF_ARRAY_BASE(u, b) items

Item Type	Item Name	Universal Revisions	b Values	Description
VARIABLE	*burst_command_1*	7	≥ 2	Data referenced from commands 105 and 108.
VARIABLE	*burst_command_2*	7	≥ 3	Data referenced from commands 105 and 108.
VARIABLE	*burst_command_3*	7	4	Data referenced from commands 105 and 108.
VARIABLE	*burst_command_number*	5, 6, 7	≥ 1	Data referenced from commands 105 and 108.
COLLECTION	*burst_message_1*	7	≥ 2	Referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
COLLECTION	*burst_message_2*	7	≥ 3	Referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
COLLECTION	*burst_message_3*	7	4	Referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
ARRAY OF COLLECTION	*burst_message_array*	5, 6, 7	≥ 1	Data referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
COLLECTION	*std_burst_message_collection*	5, 6, 7	≥ 1	Referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
VARIABLE	*burst_message_number*	7	≥ 1	Data referenced from commands 101, 102, 103, 104, 105, 107, 108 and 109.
REFRESH	*burst_message_0_relation*	7	≥ 1	Relation to update the burst trigger attributes.
REFRESH	*burst_message_1_relation*	7	≥ 2	Relation to update the burst trigger attributes.
REFRESH	*burst_message_2_relation*	7	≥ 3	Relation to update the burst trigger attributes.
REFRESH	*burst_message_3_relation*	7	4	Relation to update the burst trigger attributes.
COMMAND	*burst_mode_control*	5, 6, 7	≥ 1	Command Definition for command 109.
UNIT	*burst_trigger_0_units_relation*	7	≥ 1	Unit relation for burst trigger levels.

Item Type	Item Name	Universal Revisions	b Values	Description
UNIT	*burst_trigger_1_units_relation*	7	≥ 2	Unit relation for burst trigger levels.
UNIT	*burst_trigger_2_units_relation*	7	≥ 3	Unit relation for burst trigger levels.
UNIT	*burst_trigger_3_units_relation*	7	4	Unit relation for burst trigger levels.
VARIABLE	*burst_trigger_level_1*	7	≥ 2	Data referenced from commands 104 and 105.
VARIABLE	*burst_trigger_level_2*	7	≥ 3	Data referenced from commands 104 and 105.
VARIABLE	*burst_trigger_level_3*	7	4	Data referenced from commands 104 and 105.
VARIABLE	*burst_variable_code_0_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_0_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_0_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_1*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_1_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_1_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_1_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_2*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_2_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_2_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_2_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_3*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_3_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_3_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_3_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_4*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.

Item Type	Item Name	Universal Revisions	b Values	Description
VARIABLE	*burst_variable_code_4_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_4_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_4_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_5*	7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_5_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_5_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_5_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_6*	7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_6_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_6_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_6_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_7*	7	≥ 1	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_7_message_1*	7	≥ 2	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_7_message_2*	7	≥ 3	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_7_message_3*	7	4	Data referenced from commands 105 and 107.
VARIABLE	*burst_variable_code_8*	7	≥ 1	Data referenced from commands 105 and 107.
ARRAY OF ARRAY	*burst_variables*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.
ARRAY OF VARIABLE	*burst_variables_0*	5, 6, 7	≥ 1	Data referenced from commands 105 and 107.
ARRAY OF VARIABLE	*burst_variables_1*	7	≥ 2	Data referenced from commands 105 and 107.
ARRAY OF VARIABLE	*burst_variables_2*	7	≥ 3	Data referenced from commands 105 and 107.
ARRAY OF VARIABLE	*burst_variables_3*	7	4	Data referenced from commands 105 and 107.

Item Type	Item Name	Universal Revisions	b Values	Description
ARRAY OF COLLECTION	*burstDeviceVariables*	5, 6, 7	≥ 1	Data Type used indirectly for commands 105 and 107.
COLLECTION	*dummy_collection*	5, 6, 7	≥ 1	Data Type used indirectly for commands 105 and 107.
VARIABLE	*dummy_variable*	5, 6, 7	≥ 1	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_float*	5, 6, 7	≥ 1	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_time_int*	7	≥ 1	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_unit*	5, 6, 7	≥ 1	Stub variable used for burst indexing type.
VARIABLE	*dummy_variable_unsigned_3*	5, 6, 7	≥ 1	Stub variable used for burst indexing type.
VARIABLE	*max_update_period*	7	≥ 1	Data referenced from commands 103 and 105.
VARIABLE	*max_update_period_1*	7	≥ 2	Data referenced from commands 103 and 105.
VARIABLE	*max_update_period_2*	7	≥ 3	Data referenced from commands 103 and 105.
VARIABLE	*max_update_period_3*	7	4	Data referenced from commands 103 and 105.
COMMAND	*read_burst_mode_configuration*	6, 7	≥ 1	Command Definition for command 105.
VARIABLE	*total_number_burst_messages*	7	≥ 1	Data referenced from command 105.
VARIABLE	*trigger_level*	7	≥ 1	Data referenced from commands 104 and 105.
VARIABLE	*update_period*	7	≥ 1	Data referenced from commands 103 and 105.
VARIABLE	*update_period_1*	7	≥ 2	Data referenced from commands 103 and 105.
VARIABLE	*update_period_2*	7	≥ 3	Data referenced from commands 103 and 105.
VARIABLE	*update_period_3*	7	4	Data referenced from commands 103 and 105.
COMMAND	*write_burst_command_number*	5, 6, 7	≥ 1	Command Definition for command 108.
COMMAND	*write_burst_device_variables*	5, 6, 7	≥ 1	Command Definition for command 107.
COMMAND	*write_burst_period*	7	≥ 1	Command Definition for command 103.
COMMAND	*write_burst_trigger*	7	≥ 1	Command Definition for command 104.

7.8.1.2 IMPORT_BURST_REF_ARRAY_EVENT(*e*)

The IMPORT_BURST_REF_ARRAY_EVENT(e) macro imports a list of all items required to implement event notification configuration. This macro imports the reference array model, which is limited to a maximum of 2 event notifications. The parameter 'e' is used to specify how many event notifications are to be supported (allowable values are from 1 to 2). The same value of e has to be used in all macros that are used that call for a parameter 'e' to be used.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst reference array import section (Note – the example shows an import for a device that supports 1 event notification (index 0):

```
IMPORT_BURST_REF_ARRAY_EVENT(1)
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_EVENT(e) macro are described in Table 31. The "e Values" column identifies for which values of 'e' a variable will be imported or not.

Table 31: IMPORT_BURST_REF_ARRAY_EVENT(e) items

Item Type	Item Name	e Values	Description
COMMAND	*acknowledge_event_notification*	≥ 1	Command Definition for command 119.
COLLECTION	*event*	≥ 1	Referenced from commands 115, 116, 117, 118 and 119.
ARRAY OF COLLECTION	*event_array*	≥ 1	Data referenced from commands 101, 102, 115, 116, 117, 118 and 119.
COLLECTION	*event_control*	≥ 1	Referenced from commands 101, 102, 115, 117, 118 and 119.
COLLECTION	*event_control_1*	≥ 2	Referenced from commands 101, 102, 115, 117, 118 and 119.
VARIABLE	*event_debounce_interval_event*	≥ 1	Data referenced from commands 115, 117.
VARIABLE	*event_debounce_interval_event_1*	≥ 2	Data referenced from commands 115, 117.
COLLECTION	*event_mask*	≥ 1	Referenced from commands 115, 116.
COLLECTION	*event_mask_1*	≥ 2	Referenced from commands 115, 116.
COMMAND	*event_notification_control*	≥ 1	Command Definition for command 118.
VARIABLE	*event_notification_retry_time_event*	≥ 1	Data referenced from commands 115, 117.
VARIABLE	*event_notification_retry_time_event_1*	≥ 2	Data referenced from commands 115, 117.
COLLECTION	*event_report*	≥ 1	Referenced from command 119.
COLLECTION	*event_report_1*	≥ 2	Referenced from command 119.
COLLECTION	*event1*	≥ 2	Referenced from commands 115, 116, 117, 118 and 119.
VARIABLE	*eventNumber*	≥ 1	Data referenced from commands 101, 102, 115, 116, 117, 118 and 119.
VARIABLE	*max_update_time_event*	≥ 1	Data referenced from commands 115, 117.
VARIABLE	*max_update_time_event_1*	≥ 2	Data referenced from commands 115, 117.
VARIABLE	*number_events_supported*	≥ 1	Data referenced from command 115.

Item Type	Item Name	e Values	Description
COMMAND	*read_event_notification_summary*	≥ 1	Command Definition for command 115.
VARIABLE	*time_of_first_unack_event*	≥ 1	Data referenced from commands 115, 119.
VARIABLE	*time_of_first_unack_event_1*	≥ 2	Data referenced from commands 115, 119.
COMMAND	*write_event_notification_bit_mask*	≥ 1	Command Definition for command 116.
COMMAND	*write_event_notification_timing*	≥ 1	Command Definition for command 117.

7.8.1.3 IMPORT_BURST_REF_ARRAY_EVENT_REGISTER()

The IMPORT_BURST_REF_ARRAY_EVENT_REGISTER() macro imports a list of items required to implement event manager registration.

NOTE: This macro can only be used in DDs modeling HART rev 7 and Common Practice rev 10 or later.

To import these items, add the following line within the common practice burst reference array import section:

```
IMPORT_BURST_REF_ARRAY_EVENT_REGISTER()
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_EVENT_REGISTER() macro is described in Table 32.

Table 32: IMPORT_BURST_REF_ARRAY_EVENT_REGISTER() items

Item Type	Item Name	Description
COMMAND	*read_event_manager_registration_status*	Command Definition for command 515.
COMMAND	*register_event_manager*	Command Definition for command 514.

7.8.1.4 IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(*b*)

The IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(b) macro imports a list of all items required to implement IO system subdevices to burst messages. This macro imports the reference array model, which is limited to a maximum of 4 burst messages. The parameter ‘b’ is used to specify how many burst messages are to be supported (allowable values are from 1 to 4). The same value of b has to be used in all macros that are used that call for a parameter ‘b’ to be used.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst reference array import section (Note – the example shows an import for a device that supports 3 total burst messages (index 0, 1, and 2):

```
IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(3)
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_SUBDEVICE(b) macro is described in Table 33. The “b Values” column identifies for which values of ‘b’ a variable will be imported or not.

Table 33: IMPORT_BURST_REF_ARRAY_SUBDEVICE(b) items

Item Type	Item Name	b Values	Description
VARIABLE	*burst_subdevice_index_1*	≥ 2	Data referenced from commands 101 and 102.
VARIABLE	*burst_subdevice_index_2*	≥ 3	Data referenced from commands 101 and 102.

Item Type	Item Name	b Values	Description
VARIABLE	*burst_subdevice_index_3*	≥ 4	Data referenced from commands 101 and 102.
VARIABLE	*burst_subdevice_index_int*	≥ 1	Data referenced from commands 101 and 102.
VARIABLE	*constOne*	≥ 1	Data referenced from commands 101 and 102.
VARIABLE	*constZero*	≥ 1	Data referenced from commands 101 and 102.
COMMAND	*read_subdevice_burst_map*	≥ 1	Command Definition for command 101.
VARIABLE	*subdevice_missing*	≥ 1	Data referenced from commands 101 and 102.
VARIABLE	*subdevice_missing_1*	≥ 2	Data referenced from commands 101 and 102.
VARIABLE	*subdevice_missing_2*	≥ 3	Data referenced from commands 101 and 102.
VARIABLE	*subdevice_missing_3*	≥ 4	Data referenced from commands 101 and 102.
COMMAND	*write_subdevice_burst_map*	≥ 1	Command Definition for command 102.

7.8.1.5 IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(*e*)

The IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) macro imports a list of all items required to implement IO system subdevice mapping to event notifications. This macro imports the reference array model, which is limited to a maximum of 2 event notifications. The parameter ‘e’ is used to specify how many event notifications are to be supported (allowable values are from 1 to 2). The same value of e has to be used in all macros that are used that call for a parameter ‘e’ to be used.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst reference array import section (Note – the example shows an import for a device that supports 2 total event notifications (index 0 and 1):

```
IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(2)
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) macro is described in Table 34. The “e Values” column identifies for which values of ‘e’ a variable will be imported or not.

Table 34: IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) items

Item Type	Item Name	e Values	Description
VARIABLE	*event_subdevice_index_int*	≥ 1	Data referenced from commands 101, 102.
VARIABLE	*event_subdevice_index_int_1*	≥ 2	Data referenced from commands 101, 102.

7.8.1.6 IMPORT_BURST_REF_ARRAY_TREND()

The IMPORT_BURST_REF_ARRAY_TREND() macro imports all items required to implement variable trending. The reference array model supports capability for only a single variable trend. To support multiple variable trends in the same device, the Value Array Model of the burst DD needs to be used (see 7.7).

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice burst reference array import section:

```
IMPORT_BURST_REF_ARRAY_TREND()
```

The list of items imported using the IMPORT_BURST_REF_ARRAY_TREND() macro are described in Table 35.

Table 35: IMPORT_BURST_REF_ARRAY_TREND() items

Item Type	Item Name	Description
VARIABLE	*number_trends_supported*	Data referenced from command 91.
COMMAND	*read_trend*	Command Definition for command 93.
COMMAND	*read_trend_configuration*	Command Definition for command 91.
METHOD	*readTrendFromDevice*	Method that sends command 93 to read the trend values.
COLLECTION	*std_trend_0*	Referenced from commands 91, 92, and 93.
VARIABLE	*trend_0_date_stamp*	Data referenced from command 93.
VARIABLE	*trend_0_time_stamp*	Data referenced from command 93.
UNIT	*std_trend_0_units_relation*	Relation that associates trend values with engineering units.
COLLECTION	*std_trend_data_0*	Referenced from command 93.
COLLECTION	*std_trend_data_1*	Referenced from command 93.
COLLECTION	*std_trend_data_2*	Referenced from command 93.
COLLECTION	*std_trend_data_3*	Referenced from command 93.
COLLECTION	*std_trend_data_4*	Referenced from command 93.
COLLECTION	*std_trend_data_5*	Referenced from command 93.
COLLECTION	*std_trend_data_6*	Referenced from command 93.
COLLECTION	*std_trend_data_7*	Referenced from command 93.
COLLECTION	*std_trend_data_8*	Referenced from command 93.
COLLECTION	*std_trend_data_9*	Referenced from command 93.
COLLECTION	*std_trend_data_10*	Referenced from command 93.
COLLECTION	*std_trend_data_11*	Referenced from command 93.
ARRAY OF COLLECTION	*std_trend_data_array*	Referenced from command 93.
VARIABLE	*trend_device_variable_code*	Data referenced from commands 91, 92, and 93.
VARIABLE	*trend_number*	Data referenced from commands 91, 92, and 93.
ARRAY OF COLLECTION	*std_trend_rarray*	Referenced from commands 91, 92, and 93.
VARIABLE	*trend_sample_interval*	Data referenced from commands 91, 92, and 93.
VARIABLE	*trend_value*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_1*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_2*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_3*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_4*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_5*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_6*	Data referenced from command 93.

Item Type	Item Name	Description
VARIABLE	*std_trend_value_0_7*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_8*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_9*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_10*	Data referenced from command 93.
VARIABLE	*std_trend_value_0_11*	Data referenced from command 93.
COMMAND	*write_trend_configuration*	Command Definition for command 92.

7.8.2 DEPENDENCIES

When using the IMPORT_BURST_REF_ARRAY_BASE(u, b) import, there is a dependency on standard tables:

- IMPORT_TABLES_BURST_REF_ARRAY(u, b) Described in 7.13.1.2

When using the IMPORT_BURST_REF_ARRAY_EVENT(e) import, there is a dependency on standard tables:

- IMPORT_TABLES_EVENT_REF_ARRAY(e) Described in 7.13.1.9

When using the IMPORT_BURST_REF_ARRAY_EVENT_REGISTER() import, there is a dependency on standard tables:

- IMPORT_TABLES_EVENT_MANAGER() Described in 7.13.1.8

When using the IMPORT_BURST_REF_ARRAY_TREND() import, there is a dependency on standard tables:

- IMPORT_TABLES_TREND_REF_ARRAY() Described in 7.13.1.23

7.8.3 REDEFINITIONS

7.8.3.1 REDEFINE_BURST_REF_ARRAY_SIZE(b)

The REDEFINE_BURST_REF_ARRAY_SIZE(b) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to adjust the number of burst messages supported in the corresponding device. When using the REDEFINE_BURST_REF_ARRAY_SIZE(b) macro, the parameter ‘b’ is used to specify the number of burst messages to be implemented. The same value of b has to be used in all macros that are used that call for a parameter ‘b’ to be used.

To redefine the burst reference arrays to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 3 burst messages is shown in this example):

```
REDEFINITIONS
{
    ARRAY OF COLLECTION burst_message_array
    {
        REDEFINE_BURST_REF_ARRAY_SIZE(3)
    }

    ARRAY OF COLLECTION burst_variables
    {
        REDEFINE_BURST_REF_ARRAY_SIZE(3)
    }
}
```

7.8.3.2 REDEFINE_EVENT_REF_ARRAY_SIZE(e)

The REDEFINE_EVENT_REF_ARRAY_SIZE(e) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to adjust the number of event notification messages supported in the corresponding device. When using the REDEFINE_EVENT_REF_ARRAY_SIZE(e) macro, the parameter ‘e’ is used to specify the number of event notification messages to be implemented. The same value of e has to be used in all macros that are used that call for a parameter ‘e’ to be used.

To redefine the event reference array to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 1 event notification message is shown in this example):

```
REDEFINITIONS
{
    ARRAY OF COLLECTION event_array
    {
        REDEFINE_EVENT_REF_ARRAY_SIZE(1)
    }
}
```

7.8.3.3 REDEFINE_EVENT_MASK_SIZE(n)

The REDEFINE_EVENT_MASK_SIZE(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to adjust the number of event notification mask bytes supported in the corresponding device. When using the REDEFINE_EVENT_MASK_SIZE(n) macro, the parameter ‘n’ is used to specify the number of status bytes implemented in command 48 (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter ‘n’ to be used.

To redefine the event mask size to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 10 command 48 bytes is shown in this example):

```
REDEFINITIONS
{
    COLLECTION event_mask
    {
        REDEFINE_EVENT_MASK_SIZE(10)
    }

    COLLECTION event_mask_1
    {
        REDEFINE_EVENT_MASK_SIZE(10)
    }
}
```

7.8.3.4 REDEFINE_EVENT_REPORT_SIZE(n)

The REDEFINE_EVENT_REPORT_SIZE(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to adjust the number of event notification report bytes supported in the corresponding device. When using the REDEFINE_EVENT_REPORT_SIZE(n) macro, the parameter ‘n’ is used to

specify the number of status bytes implemented in command 48 (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter ‘n’ to be used.

To redefine the event report size to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 10 command 48 bytes, and 2 event notification messages is shown in this example. If only 1 event notification message is supported, then the COLLECTION event_report_1 would not be redefined):

```
REDEFINITIONS
{
    COLLECTION event_report
    {
        REDEFINE_EVENT_REPORT_SIZE(10)
    }

    COLLECTION event_report_1
    {
        REDEFINE_EVENT_REPORT_SIZE(10)
    }
}
```

7.8.3.5 REDEFINE_COMMAND_115_REF_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_115_REF_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to truncate the number of event notification status masks to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_115_REF_ARRAY_TRANSACTIONS(n) macro, the parameter ‘n’ is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter ‘n’ to be used.

To redefine the command 115 command transactions to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND read_event_notification_summary
    {
        REDEFINE_COMMAND_115_REF_ARRAY_TRANSACTIONS(10)
    }
}
```

7.8.3.6 REDEFINE_COMMAND_116_REF_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_116_REF_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to truncate the number of event notification status masks to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_116_REF_ARRAY_TRANSACTIONS(n) macro, the parameter ‘n’ is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter ‘n’ to be used.

To redefine the command 116 command transactions to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND write_event_notification_bit_mask
    {
        REDEFINE_COMMAND_116_REF_ARRAY_TRANSACTIONS(10)
    }
}
```

7.8.3.7 REDEFINE_COMMAND_119_REF_ARRAY_TRANSACTIONS(*n*)

The REDEFINE_COMMAND_119_REF_ARRAY_TRANSACTIONS(n) macro is used in the REDEFINITIONS section of the Common Practice Burst Reference Array import. It is used to truncate the number of event notification status bytes to match the number of command 48 diagnostic bytes supported in the corresponding device. When using the REDEFINE_COMMAND_119_REF_ARRAY_TRANSACTIONS(n) macro, the parameter 'n' is used to specify the number of command 48 status bytes to be implemented (not including the response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To redefine the command 119 command transactions to match the device, add the following lines within the Burst Reference Array import redefinition section (Note – A device implementing 10 bytes of status in command 48 is shown in this example):

```
REDEFINITIONS
{
    COMMAND acknowledge_event_notification
    {
        REDEFINE_COMMAND_119_REF_ARRAY_TRANSACTIONS(10)
    }
}
```

7.8.3.8 ARRAY OF COLLECTION burstDeviceVariables

The burstDeviceVariables array imported for HART5 and HART6 devices only contains the variable code 250 "Not Used". This variable code must be supported in the burst variable slots and cannot be deleted. However, the burstDeviceVariables array imported for HART 7 contains all the standardized HART variable codes from 242 through 250. Some devices may not support all the standardized device variable codes (e.g. 242 and 243 for battery voltage and battery life), so those codes may need to be deleted from the burstDeviceVariables array. Additionally, when importing the burstDeviceVariables ARRAY from the HART common practice Burst Reference Array DD, the array needs to be redefined to include all the device specific variables that can be bursted from the device.

There may also be a need to affect the ordering of burst device variables in the list (so the end user sees the enumerated selections appear in a certain order). ADDing enumerations causes them to be placed at the bottom of the list. To adjust the order of enumerated items, it may be necessary to DELETE some existing enumerations, and then ADD them back in at the right order in th list. The example below shows how device specific variables can be added to the list to appear before the standardized variables that are already in the list.

NOTE: The HART protocol specification states that a field device will return a failure response code if an attempt is made to write the burst slot#0 to 250 – "Not Used". However, the HART standard DD treats all 8 burst slots the same, as implementing the first slot to have a different set of enumerations compared to the other slots adds a layer of complexity, and may cause some compatibility issues for field devices developed prior to the addition of this protocol requirement. End users will unlikely intentionally set the first slot to 250 anyway – but if they do, some field devices will return an error so the user can change their selection and try again. No attempt should be made to remove code 250 from the list of burst slots because it should be possible to select for any one or more of the last 7 burst slots. Some DD developers may choose to add an action to warn the user if slot #0 is set to '250'.

To redefine the burstDeviceVariables array, add lines similar to the following lines to the redefinition section of the Common Practice Burst Reference Array DD import:

```
REDEFINITIONS
{
    ARRAY OF COLLECTION burstDeviceVariables
    {
        ELEMENTS
        {
            DELETE 242; //Deleted because not supported
            DELETE 243; //Deleted because not supported
            DELETE 244; //Add later to preserve order
            DELETE 245; //Add later to preserve order
            DELETE 246; //Add later to preserve order
            DELETE 247; //Add later to preserve order
            DELETE 248; //Add later to preserve order
            DELETE 249; //Add later to preserve order
            DELETE 250; //Add later to preserve order
            ADD 0, deviceVariables[0], "Pressure";
            ADD 1, deviceVariables[1], "Temperature";
            ...
            ADD {244, dummy_collection, device_variable_code_codes(244); // Reorder
            ADD {245, dummy_collection, device_variable_code_codes(245); // Reorder
            ADD {246, dummy_collection, device_variable_code_codes(246); // Reorder
            ADD {247, dummy_collection, device_variable_code_codes(247); // Reorder
            ADD {248, dummy_collection, device_variable_code_codes(248); // Reorder
            ADD {249, dummy_collection, device_variable_code_codes(249); // Reorder
            ADD {250, dummy_collection, device_variable_code_codes(250); // Reorder
        }
    }
}
```

Some devices may need to declare a conditional set of burst slot variables based on varying support for different device variables based on certain device operating conditions. The example below shows how conditional support of variables can be defined to appear in the list.

To redefine the burstDeviceVariables array to have conditional items, add lines similar to the following lines to the redefinition section of the Common Practice Burst Value Array DD import:

```
REDEFINITIONS
{
    ARRAY OF COLLECTION burstDeviceVariables
    {
        REDEFINE ELEMENTS
        {
            0, deviceVariables[0], “devVar 0”;
            IF (condition)
            {
                1, deviceVariables[1],”devVar 1”;
            }
        }
    }
}
```

7.8.3.9 HART5 Write-only

In the HART revision 5 protocol specification, there is not a command to read back burst mode settings. Because of this, the variable burst_mode_select_0 should have its HANDLING redefined to be write-only as follows:

```
REDEFINITIONS
{
    VARIABLE burst_mode_select_0
```

```
        {
            REDEFINE HANDLING WRITE;
        }
    }
```

7.8.3.10 Vendor-specific burst command list

When importing the burst command number variables, the list of command numbers in the enumeration is defined depending on which universal revision was used in the import macro as follows:

HART 7: Commands, 1, 2, 3, 9, and 48.

HART 6: Commands 1, 2, 3, 9.

HART 5: Commands 1, 2, 3.

In some cases, a device may allow for bursting of additional HART commands. In this case, the burst command number variables need to add the additional supported choices.

There may also be a need to affect the ordering of burst messages in the list (so the end user sees the enumerated selections appear in a certain order). ADDing enumerations causes them to be placed at the bottom of the list. To adjust the order of enumerated items, it may be necessary to DELETE some existing enumerations, and then ADD them back in at the right order in the list. The example below shows how optional command 33 can be added to the list to appear before command 48 that is already in the list.

To redefine the burst command number variables to support additional HART commands, add lines similar to the following to the Common Practice Burst Reference Array redefinition section:

```
REDEFINITIONS
{
    VARIABLE burst_command_number
    {
        TYPE ENUMERATED
        {
            DELETE 48;
            ADD {33, Read_Device_Variables_temp,     Read_Device_Variables_help_temp }
            ADD {48, read_additional_device_status_temp,
                     read_additional_device_status_help_temp}
            ADD {178, "My Special Burst Command", "My Help"}
        }
    }

    VARIABLE burst_command_1
    {
        TYPE ENUMERATED
        {
            DELETE 48;
            ADD {33, Read_Device_Variables_temp,     Read_Device_Variables_help_temp }
            ADD {48, read_additional_device_status_temp,
                     read_additional_device_status_help_temp}
            ADD {178, "My Special Burst Command", "My Help"}
        }
    }

    VARIABLE burst_command_2
    {
        TYPE ENUMERATED
        {
            DELETE 48;
            ADD {33, Read_Device_Variables_temp,     Read_Device_Variables_help_temp }
            ADD {48, read_additional_device_status_temp,
```

```
                    read_additional_device_status_help_temp}
            ADD {178, "My Special Burst Command", "My Help"}
        }
    }

    VARIABLE burst_command_3
    {
        TYPE ENUMERATED
        {
            DELETE 48;
            ADD {33, Read_Device_Variables_temp,     Read_Device_Variables_help_temp }
            ADD {48, read_additional_device_status_temp,
                    read_additional_device_status_help_temp}
            ADD {178, "My Special Burst Command", "My Help"}
        }
    }
```

7.8.3.11 Burst and Event Notification Message Collections

When importing std_burst_message_collection, burst_message_1, burst_message_2, burst_message_3, event_control, and event_control_1 collections from the HART common practice Burst Reference Array DD, the collections need to be redefined to remove IO system specific members if the DD is not being developed for an IO System.

All devices that don't connect to subdevices (are not IO Systems) should have the following lines in the redefinition section of the Common Practice Burst Reference Array DD import (depending on how many burst messages and event notifications are supported):

```
REDEFINITIONS
{
    COLLECTION std_burst_message_collection
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }

    COLLECTION burst_message_1
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }

    COLLECTION burst_message_2
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }

    COLLECTION burst_message_3
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
```

```
        }
    }

    COLLECTION event_control
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }

    COLLECTION event_control_1
    {
        MEMBERS
        {
            DELETE SUB_DEVICE_MISSING;
            DELETE SUB_DEVICE_MAPPING;
        }
    }
}
```

7.8.3.12 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE update_period_1
    {
        SET_DEFAULT(1920000);
    }
}
```

7.8.3.13 Minimum and Maximum Values

It is best practice for all numeric values that are writeable to define an upper and lower limit. This helps to prevent an out of bounds entry from being attempted to be sent to a field device.

To define minimum and maximum values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE trigger_level
    {
        REDEFINE MIN_VALUE -200.0;
        REDEFINE MAX_VALUE 850.0;
    }
}
```

7.8.3.14 Display, Edit, and Time Formats

Depending on the measurement types and capabilities of the field device, the display format, edit format and/or time ormat of certain variables may need to be adjusted to match characteristics of the field device. Variables such as trigger_level, trend_value, and others may need to have their display, edit, and/or time formats adjusted.

To redefine the display, edit, or time format for a variable, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE trigger_level
    {
        TYPE FLOAT
        {
            REDEFINE DISPLAY_FORMAT ".2f";
            REDEFINE EDIT_FORMAT ".2f";
        }
    }

    VARIABLE trend_0_time_stamp
    {
        TYPE TIME_VALUE
        {
            REDEFINE TIME_FORMAT %H:%M";
        }
    }
}
```

7.8.3.15 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD readTrendFromDevice
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.8.3.16 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE burst_variable_code_1
    {
        REDEFINE HELP "Burst Variable Slot 1- Device variable code assigned to slot 1 to be
read in burst mode. For this device, variable code setting of 0 is for the main pressure
value, and variable code setting of 1 is for the external temperature value.";
    }
}
```

7.9 HART Common Practice IO System LIST DD

The Common Practice IO System LIST DD includes all items needed to implement IO system subdevice functionality. IO System subdevices can be modelled using either lists or reference arrays. This DD implements the list model, which is recommended because it allows any number of subdevices to be added or removed from the list automatically at run time. The number of elements in the subdevice list can change as the IO system detects changes in the live list of subdevices.

The Common Practice IO System List binary files are installed in \HCF\DDL\Library\000000\001e\0904.*, and 0b02.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Common9_4_IO_System_List.ddl, and Common11_2_IO_System_List.ddl.

NOTE: This DD can only be imported when modeling HART Universal Revision 7 or later. HART revisions 5 and 6 do not have support for IO System commands.

To import the common practice IO system list DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_IO_SYSTEM_LIST, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_IO_LIST_BASE_V9()
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_IO_SYSTEM_LIST, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_IO_LIST_BASE_V11()
}
```

NOTE: If this DD is imported, do NOT import the Common Practice IO System Reference Array DD.

NOTE: If this DD is imported, the Common Practice Burst Value Array DD must also be imported.

Import macros have been provided to make it easier to extract the correct items from the Common Practice IO System List DD. A description of each of these macros is described in 7.9.1.

7.9.1 IMPORTS

7.9.1.1 IMPORT_IO_LIST_ASSIGN()

The IMPORT_IO_LIST_ASSIGN() macro imports items required for IO system subdevice list assignment.

NOTE: This macro can only be used in DDs modeling HART Common Practice rev 10 or later.

To import these items, add the following line within the common practice IO system list import section:

```
IMPORT_IO_LIST_ASSIGN()
```

The list of items imported using the IMPORT_IO_LIST_ASSIGN() macro are described in Table 36.

Table 36: IMPORT_IO_LIST_ASSIGN() items

Item Type	Item Name	Description
LIST	*assignment_list*	Data referenced from commands 529 and 530.
VARIABLE	*max_subdevs_assigned*	Data referenced from command 528.
VARIABLE	*num_devices_transferred*	Data referenced from command 531.
COMMAND	*read_subdevice_assignment*	Command Definition for command 529.
COMMAND	*read_subdevice_assignment_list_info*	Command Definition for command 528.

Item Type	Item Name	Description
COLLECTION	*subdev_assignment*	Referenced from commands 529 and 530.
VARIABLE	*subdevs_assigned*	Data referenced from command 528.
COMMAND	*transfer_live_to_assignment_list*	Command Definition for command 531.
COMMAND	*write_subdevice_assignment*	Command Definition for command 530.

7.9.1.2 IMPORT_IO_LIST_BASE_V11()

The IMPORT_IO_LIST_BASE_V11() macro imports the items required to be included for implementation of IO systems with subdevices as was updated starting with Common Practice rev 11 DD.

NOTE: If the IMPORT_IO_LIST_BASE_V9() macro is used, do NOT use the IMPORT_IO_LIST_BASE_V11() macro.

NOTE: This macro can only be used in DDs modeling HART Universal rev 7 and Common Practice rev 10 or later.

To import these required tables, add the following line within the common practice IO system list import section:

```
IMPORT_IO_LIST_BASE_V11()
```

The list of items imported using the IMPORT_IO_LIST_BASE_V11() macro are described in Table 37.

Table 37: IMPORT_IO_LIST_BASE_V11() items

Item Type	Item Name	Description
VARIABLE	*ack_received_from_subdev*	Data referenced from command 86.
VARIABLE	*back_received_from_subdev*	Data referenced from command 86.
ARRAY	*card0_channels*	Data referenced from command 85.
VARIABLE	*devices_detected*	Data referenced from command 74.
ARRAY OF ARRAY	*io_cards*	Data referenced from command 85.
VARIABLE	*io_channel_ack_received*	Data referenced from command 85.
VARIABLE	*io_channel_back_received*	Data referenced from command 85.
VARIABLE	*IO_channel_number*	Data referenced from command 85.
VARIABLE	*io_channel_oack_received*	Data referenced from command 85.
VARIABLE	*io_channel_ostx_received*	Data referenced from command 85.
VARIABLE	*io_channel_stx_sent*	Data referenced from command 85.
COLLECTION	*io_stats*	Referenced from command 85.
VARIABLE	*max_channels_per_io_card*	Data referenced from command 74.
VARIABLE	*max_dr*	Data referenced from command 74.
VARIABLE	*max_io_cards*	Data referenced from command 74.
VARIABLE	*max_sub_dev_per_channel*	Data referenced from command 74.
COMMAND	*read_io_channel_stats*	Command Definition for command 85.
COMMAND	*read_io_system_capabilities*	Command Definition for command 74.

Item Type	Item Name	Description
COMMAND	*read_subdev_id_summary*	Command Definition for command 84.
COMMAND	*read_subdevice_statistics*	Command Definition for command 86.
VARIABLE	*retry_count*	Data referenced from commands 74 and 88.
VARIABLE	*stx_sent_to_subdev*	Data referenced from command 86.
COLLECTION	*sub_device*	Referenced from commands 84 and 86.
VARIABLE	*subdev_channel*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_dev_rev*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_device_id*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_identity*	Referenced from command 84.
VARIABLE	*subdev_IO_card*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_long_tag*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_stats*	Referenced from command 86.
VARIABLE	*subdev_universal_command_revision*	Data referenced from command 84.
LIST	*subdevice_list*	Data referenced from commands 84 and 86.
VARIABLE	*subdevice_number*	Data referenced from commands 84, 86, 529, 530.
COMMAND	*write_io_system_master_mode*	Command Definition for command 87.
COMMAND	*write_io_system_retry_count*	Command Definition for command 88.

7.9.1.3 IMPORT_IO_LIST_BASE_V9()

The IMPORT_IO_LIST_BASE_V9() macro imports the items required to be included for implementation of IO systems with subdevices as it was originally defined starting in Common Practice rev 9.

NOTE: If the IMPORT_IO_LIST_BASE_V11() macro is used, do NOT use the IMPORT_IO_LIST_BASE_V9() macro.

NOTE: This macro can only be used in DDs modeling HART rev 7 and prior to Common Practice rev 10.

To import these required tables, add the following line within the common practice IO system list import section:

```
IMPORT_IO_LIST_BASE_V9()
```

The list of items imported using the IMPORT_IO_LIST_BASE_V9() macro are described in Table 38.

Table 38: IMPORT_IO_LIST_BASE_V9() items

Item Type	Item Name	Description
VARIABLE	*ack_received_from_subdev*	Data referenced from command 86.
VARIABLE	*back_received_from_subdev*	Data referenced from command 86.
ARRAY	*card0_channels*	Data referenced from command 85.
VARIABLE	*devices_detected*	Data referenced from command 74.
ARRAY OF ARRAY	*io_cards*	Data referenced from command 85.

Item Type	Item Name	Description
VARIABLE	*io_channel_ack_received*	Data referenced from command 85.
VARIABLE	*io_channel_back_received*	Data referenced from command 85.
VARIABLE	*IO_channel_number*	Data referenced from command 85.
VARIABLE	*io_channel_oack_received*	Data referenced from command 85.
VARIABLE	*io_channel_ostx_received*	Data referenced from command 85.
VARIABLE	*io_channel_stx_sent*	Data referenced from command 85.
COLLECTION	*io_stats*	Referenced from command 85.
VARIABLE	*max_channels_per_io_card*	Data referenced from command 74.
VARIABLE	*max_dr*	Data referenced from command 74.
VARIABLE	*max_io_cards*	Data referenced from command 74.
VARIABLE	*max_sub_dev_per_channel*	Data referenced from command 74.
COMMAND	*read_io_channel_stats*	Command Definition for command 85.
COMMAND	*read_io_system_capabilities*	Command Definition for command 74.
COMMAND	*read_subdev_id_summary*	Command Definition for command 84.
COMMAND	*read_subdevice_statistics*	Command Definition for command 86.
VARIABLE	*retry_count*	Data referenced from commands 74 and 88.
VARIABLE	*stx_sent_to_subdev*	Data referenced from command 86.
COLLECTION	*sub_device*	Referenced from commands 84 and 86.
VARIABLE	*subdev_channel*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_device_id*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_identity*	Referenced from command 84.
VARIABLE	*subdev_IO_card*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_long_tag*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_stats*	Referenced from command 86.
VARIABLE	*subdev_universal_command_revision*	Data referenced from command 84.
LIST	*subdevice_list*	Data referenced from commands 84 and 86.
VARIABLE	*subdevice_number*	Data referenced from commands 84, 86, 529, 530.
COMMAND	*write_io_system_master_mode*	Command Definition for command 87.
COMMAND	*write_io_system_retry_count*	Command Definition for command 88.

7.9.1.4 IMPORT_IO_LIST_SYSTEM_COMM_STATS()

The IMPORT_IO_LIST_SYSTEM_COMM_STATS() macro imports items required to support system side communication statistics in IO systems.

To import these items, add the following line within the common practice IO system list import section:

```
IMPORT_IO_LIST_SYSTEM_COMM_STATS()
```

The list of items imported using the IMPORT_IO_LIST_SYSTEM_COMM_STATS() macro is described in Table 39.

Table 39: IMPORT_IO_LIST_SYSTEM_COMM_STATS() items

Item Type	Item Name	Description
VARIABLE	*messages_received*	Data referenced from command 94.
VARIABLE	*messages_returned*	Data referenced from command 94.
COMMAND	*read_io_client_side_communication_statistics*	Command Definition for command 94
VARIABLE	*requests_forwarded*	Data referenced from command 94.
VARIABLE	*responses_returned*	Data referenced from command 94.

7.9.2 DEPENDENCIES

When using the IMPORT_IO_LIST_BASE_V11() import, there is a dependency on standard tables and burst messages:

- IMPORT_TABLES_IO_ADAPTER_V11() Described in 7.13.1.12
- IMPORT_TABLES_DEVICE_TYPES_GROUP1() Described in 7.13.1.6
- IMPORT_TABLES_DEVICE_TYPES_GROUP2() Described in 7.13.1.7
- IMPORT_BURST_VAL_ARRAY_SUBDEVICE() Described in 7.7.1.4

When using the IMPORT_IO_LIST_BASE_V9() import, there is a dependency on standard tables and burst messages:

- IMPORT_TABLES_IO_ADAPTER_V9() Described in 7.13.1.13
- IMPORT_TABLES_DEVICE_TYPES_GROUP1() Described in 7.13.1.6
- IMPORT_TABLES_DEVICE_TYPES_GROUP2() Described in 7.13.1.7
- IMPORT_BURST_VAL_ARRAY_SUBDEVICE() Described in 7.7.1.4

When using the IMPORT_IO_LIST_ASSIGN() import, there is a dependency on standard tables:

- IMPORT_TABLES_IO_ASSIGN() Described in 7.13.1.14

7.9.3 REDEFINITIONS

7.9.3.1 REDEFINE_CARD_CHANNEL_VAL_ARRAY_SIZE(c)

The REDEFINE_CARD_CHANNEL_VAL_ARRAY_SIZE(c) macro is used in the REDEFINITIONS section of the Common Practice IO System List import. It is used to adjust the number of channels supported in the corresponding IO System. When using the REDEFINE_CARD_CHANNEL_VAL_ARRAY_SIZE(c) macro, the parameter ‘c’ is used to specify the number of card channels to be implemented. The same value of c has to be used in all macros that are used that call for a parameter ‘c’ to be used.

To redefine the number of channels to match what the device supports, add the following lines within the IO System List import redefinition section (Note – A device implementing 2 channels is shown in this example):

```
REDEFINITIONS
{
    ARRAY card0_channels
    {
        REDEFINE_CARD_CHANNEL_VAL_ARRAY_SIZE(2)
    }
}
```

7.9.3.2 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE max_io_cards
    {
        SET_DEFAULT(4);
    }
}
```

7.9.3.3 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE max_io_cards
    {
        REDEFINE HELP "Maximum Number of I/O Cards. This device only supports 1 IO card.";
    }
}
```

7.10 HART Common Practice IO System Reference Array DD

The HART Common Practice IO System Reference Array DD includes items needed to implement IO system subdevice functionality. IO System subdevices can be modelled using either lists or reference arrays. This DD implements the list model, which requires a method to manually read information about 1 subdevice at a time.

The Common Practice IO System Reference Array binary files are installed in \HCF\DDL\Library\000000\001f\0904.* and 0b02.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\ Common9_4_IO_System_Ref_Array.ddl, and Common11_2_IO_System_Ref_Array.ddl.

NOTE: This DD can only be imported when modeling HART Universal Revision 7 or later. HART revisions 5 and 6 do not have support for IO System commands.

To import the common practice IO system reference array DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_IO_SYSTEM_REF_ARRAY, DEVICE_REVISION 9, DD_REVISION 4
{
    IMPORT_IO_REF_ARRAY_BASE_V9()
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_IO_SYSTEM_REF_ARRAY, DEVICE_REVISION 11, DD_REVISION 2
{
    IMPORT_IO_REF_ARRAY_BASE_V11()
}
```

NOTE: If this DD is imported, do NOT import the Common Practice IO System List DD.

NOTE: If this DD is imported, the Common Practice Burst Reference Array DD must also be imported.

Import macros have been provided to make it easier to extract the correct items from the Common Practice IO System Reference Array DD. A description of each of these macros is described in 7.10.1.

7.10.1 IMPORTS

7.10.1.1 IMPORT_IO_REF_ARRAY_ASSIGN()

The IMPORT_IO_REF_ARRAY_ASSIGN() macro imports items required for IO system subdevice list assignment.

NOTE: This macro can only be used in DDs modeling HART Common Practice rev 10 or later.

To import these items, add the following line within the common practice IO system reference array import section:

```
IMPORT_IO_REF_ARRAY_ASSIGN()
```

The list of items imported using the IMPORT_IO_REF_ARRAY_ASSIGN() macro is described in Table 40.

Table 40: IMPORT_IO_REF_ARRAY_ASSIGN() items

Item Type	Item Name	Description
LIST	*assignment_list*	Data referenced from commands 529 and 530.
VARIABLE	*max_subdevs_assigned*	Data referenced from command 528.
VARIABLE	*num_devices_transferred*	Data referenced from command 531.
COMMAND	*read_subdevice_assignment*	Command Definition for command 529.
COMMAND	*read_subdevice_assignment_list_info*	Command Definition for command 528.

Item Type	Item Name	Description
COLLECTION	*subdev_assignment*	Referenced from commands 529 and 530.
VARIABLE	*subdevice_number*	Index referenced from commands 529 and 530.
VARIABLE	*subdevs_assigned*	Data referenced from command 528.
COMMAND	*transfer_live_to_assignment_list*	Command Definition for command 531.
COMMAND	*write_subdevice_assignment*	Command Definition for command 530.

7.10.1.2 IMPORT_IO_REF_ARRAY_BASE_V11()

The IMPORT_IO_REF_ARRAY_BASE_V11() macro imports the items required to be included for implementation of IO systems with subdevices as was updated starting with Common Practice rev 11 DD.

NOTE: If the IMPORT_IO_REF_ARRAY_BASE_V9() macro is used, do NOT use the IMPORT_IO_REF_ARRAY_BASE_V11() macro.

NOTE: This macro can only be used in DDs modeling HART Universal rev 7 and Common Practice rev 10 or later.

To import these required tables, add the following line within the common practice IO system reference array import section:

```
IMPORT_IO_REF_ARRAY_BASE_V11()
```

The list of items imported using the IMPORT_IO_REF_ARRAY_BASE_V11() macro is described in Table 42.

Table 41: IMPORT_IO_REF_ARRAY_BASE_V11() items

Item Type	Item Name	Description
VARIABLE	*ack_received_from_subdev*	Data referenced from command 86.
VARIABLE	*back_received_from_subdev*	Data referenced from command 86.
VARIABLE	*devices_detected*	Data referenced from command 74.
ARRAY OF COLLECTION	*IO_card0*	Referenced from command 85.
VARIABLE	*io_channel_ack_received*	Data referenced from command 85.
VARIABLE	*io_channel_back_received*	Data referenced from command 85.
VARIABLE	*IO_channel_number*	Data referenced from command 85.
VARIABLE	*io_channel_oack_received*	Data referenced from command 85.
VARIABLE	*io_channel_ostx_received*	Data referenced from command 85.
VARIABLE	*io_channel_stx_sent*	Data referenced from command 85.
COLLECTION	*io_stats*	Referenced from command 85.
VARIABLE	*max_channels_per_io_card*	Data referenced from command 74.
VARIABLE	*max_dr*	Data referenced from command 74.
VARIABLE	*max_io_cards*	Data referenced from command 74.
VARIABLE	*max_sub_dev_per_channel*	Data referenced from command 74.
COMMAND	*read_io_channel_stats*	Command Definition for command 85.

Item Type	Item Name	Description
COMMAND	*read_io_system_capabilities*	Command Definition for command 74.
COMMAND	*read_subdev_id_summary*	Command Definition for command 84.
COMMAND	*read_subdevice_statistics*	Command Definition for command 86.
VARIABLE	*retry_count*	Data referenced from commands 74 and 88.
VARIABLE	*stx_sent_to_subdev*	Data referenced from command 86.
COLLECTION	*sub_device*	Referenced from commands 84 and 86.
VARIABLE	*subdev_channel*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_dev_rev*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_device_id*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_identity*	Referenced from command 84.
VARIABLE	*subdev_IO_card*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_long_tag*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_stats*	Referenced from command 86.
VARIABLE	*subdev_universal_command_revision*	Data referenced from command 84.
VARIABLE	*subdevice_number_int*	Data referenced from commands 84, 86, 529, 530.
COMMAND	*write_io_system_master_mode*	Command Definition for command 87.
COMMAND	*write_io_system_retry_count*	Command Definition for command 88.

7.10.1.3 IMPORT_IO_REF_ARRAY_BASE_V9()

The IMPORT_IO_REF_ARRAY_BASE_V9() macro imports the items required to be included for implementation of IO systems with subdevices as it was originally defined starting in Common Practice rev 9.

NOTE: If the IMPORT_IO_REF_ARRAY_BASE_V11() macro is used, do NOT use the IMPORT_IO_REF_ARRAY_BASE_V9() macro.

NOTE: This macro can only be used in DDs modeling HART rev 7 and prior to Common Practice rev 10.

To import these required tables, add the following line within the common practice IO system reference array import section:

```
IMPORT_IO_REF_ARRAY_BASE_V9()
```

The list of items imported using the IMPORT_IO_REF_ARRAY_BASE_V9() macro is described in Table 42.

Table 42: IMPORT_IO_REF_ARRAY_BASE_V9() items

Item Type	Item Name	Description
VARIABLE	*ack_received_from_subdev*	Data referenced from command 86.
VARIABLE	*back_received_from_subdev*	Data referenced from command 86.
VARIABLE	*devices_detected*	Data referenced from command 74.
ARRAY OF COLLECTION	*IO_card0*	Referenced from command 85.
VARIABLE	*io_channel_ack_received*	Data referenced from command 85.

Item Type	Item Name	Description
VARIABLE	*io_channel_back_received*	Data referenced from command 85.
VARIABLE	*IO_channel_number*	Data referenced from command 85.
VARIABLE	*io_channel_oack_received*	Data referenced from command 85.
VARIABLE	*io_channel_ostx_received*	Data referenced from command 85.
VARIABLE	*io_channel_stx_sent*	Data referenced from command 85.
COLLECTION	*io_stats*	Referenced from command 85.
VARIABLE	*max_channels_per_io_card*	Data referenced from command 74.
VARIABLE	*max_dr*	Data referenced from command 74.
VARIABLE	*max_io_cards*	Data referenced from command 74.
VARIABLE	*max_sub_dev_per_channel*	Data referenced from command 74.
COMMAND	*read_io_channel_stats*	Command Definition for command 85.
COMMAND	*read_io_system_capabilities*	Command Definition for command 74.
COMMAND	*read_subdev_id_summary*	Command Definition for command 84.
COMMAND	*read_subdevice_statistics*	Command Definition for command 86.
VARIABLE	*retry_count*	Data referenced from commands 74 and 88.
VARIABLE	*stx_sent_to_subdev*	Data referenced from command 86.
COLLECTION	*sub_device*	Referenced from commands 84 and 86.
VARIABLE	*subdev_channel*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_device_id*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_identity*	Referenced from command 84.
VARIABLE	*subdev_IO_card*	Data referenced from commands 84, 529, 530.
VARIABLE	*subdev_long_tag*	Data referenced from commands 84, 529, 530.
COLLECTION	*subdev_stats*	Referenced from command 86.
VARIABLE	*subdev_universal_command_revision*	Data referenced from command 84.
VARIABLE	*subdevice_number_int*	Data referenced from commands 84, 86, 529, 530.
COMMAND	*write_io_system_master_mode*	Command Definition for command 87.
COMMAND	*write_io_system_retry_count*	Command Definition for command 88.

7.10.1.4 IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS()

The IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS() macro imports items required to support system side communication statistics in IO systems.

To import these items, add the following line within the common practice IO system reference array import section:

```
IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS()
```

The list of items imported using the IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS() macro are described in Table 43.

Table 43: IMPORT_IO_REF_ARRAY_SYSTEM_COMM_STATS() items

Item Type	Item Name	Description
VARIABLE	*messages_received*	Data referenced from command 94.
VARIABLE	*messages_returned*	Data referenced from command 94.
COMMAND	*read_io_client_side_communication_statistics*	Command Definition for command 94.
VARIABLE	*requests_forwarded*	Data referenced from command 94.
VARIABLE	*responses_returned*	Data referenced from command 94.

7.10.2 DEPENDENCIES

When using the IMPORT_IO_REF_ARRAY_BASE_V11() import, there is a dependency on standard tables and burst messages:

- IMPORT_TABLES_IO_ADAPTER_V11() Described in 7.13.1.12
- IMPORT_TABLES_DEVICE_TYPES_GROUP1() Described in 7.13.1.6
- IMPORT_TABLES_DEVICE_TYPES_GROUP2() Described in 7.13.1.7
- IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(b) Described in 7.8.1.4
- IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) Described in 7.8.1.5

When using the IMPORT_IO_REF_ARRAY_BASE_V9() import, there is a dependency on tables and burst messages:

- TABLES_IO_ADAPTER_V9() Described in 7.13.1.13
- IMPORT_TABLES_DEVICE_TYPES_GROUP1() Described in 7.13.1.6
- IMPORT_TABLES_DEVICE_TYPES_GROUP2() Described in 7.13.1.7
- IMPORT_BURST_REF_ARRAY_SUBDEVICE_BURST(b) Described in 7.8.1.4
- IMPORT_BURST_REF_ARRAY_SUBDEVICE_EVENT(e) Described in 7.8.1.5

When using the IMPORT_IO_LIST_ASSIGN() import, there is a dependency on standard tables:

- IMPORT_TABLES_IO_ASSIGN() Described in 7.13.1.14

7.10.3 REDEFINITIONS

7.10.3.1 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter ‘x’ is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE max_io_cards
    {
        SET_DEFAULT(4);
    }
}
```

7.10.3.2 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE max_io_cards
    {
        REDEFINE HELP "Maximum Number of I/O Cards. This device only supports 1 IO card.";
    }
}
```

7.11 HART Common Practice Miscellaneous DD

The Common Practice Miscellaneous DD includes some additional common practice items that are not already categorized into the other Common Practice DD parts. The Common Practice Miscellaneous binary files are installed in \HCF\DDL\Library\000000\0020\0706.*, 0804.*, 0904.*, 0b02.*, and 0c02.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Common7_6_Misc.ddl, Common8_4_Misc.ddl, Common9_4_Misc.ddl, Common11_2_Misc.ddl, and Common12_2_Misc.ddl.

Only one device revision of the DD (either 7, 8, 9, 11, or 12) should be imported into a device DD depending on which HART universal revision is being modelled. To import the common practice miscellaneous DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 5 */
IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 7, DD_REVISION 6
{
    list of macros or individual items;
}

/* For HART Rev 6 */
IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 8, DD_REVISION 4
{
    list of macros or individual items;
}

/* For HART Rev 7, Common Practice 9 */
IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 9, DD_REVISION 4
{
    list of macros or individual items;
}

/* For HART Rev 7, Common Practice 10 or 11 */
IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 11, DD_REVISION 2
{
    list of macros or individual items;
}

/* For HART Rev 7, Common Practice 12 */
IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 12, DD_REVISION 2
{
    list of macros or individual items;
}
```

Import macros have been provided to make it easier to extract the correct items from the Common Practice Miscellaneous DD. A description of each of these macros is described in 7.11.1.

7.11.1 IMPORTS

7.11.1.1 IMPORT_MISC_COMM_STATS()

The IMPORT_MISC_COMM_STATS() macro imports items used to implement device communications statistics.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_COMM_STATS()
```

The list of items imported using the IMPORT_MISC_COMM_STATS() macro is described in Table 44.

Table 44: IMPORT_MISC_COMM_STATS() items

Item Type	Item Name	Description
VARIABLE	*ACK_count*	Data referenced from command 95.
VARIABLE	*BACK_count*	Data referenced from command 95.
COMMAND	*read_device_communication_statistics*	Command Definition for command 95.
VARIABLE	*STX_count*	Data referenced from command 95.

7.11.1.2 IMPORT_MISC_LOCATION()

The IMPORT_MISC_ LOCATION() macro imports items used to implement location coordinates.

NOTE: This macro can only be used in DDs modeling HART Universal Rev 7 and Common Practice Rev 10 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_LOCATION()
```

The list of items imported using the IMPORT_MISC_LOCATION() macro is described in Table 45.

Table 45: IMPORT_MISC_LOCATION() items

Item Type	Item Name	Description
VARIABLE	*device_altitude*	Data referenced from commands 516 and 517.
VARIABLE	*device_latitude*	Data referenced from commands 516 and 517.
VARIABLE	*device_location_description*	Data referenced from commands 516 and 517.
VARIABLE	*device_longitude*	Data referenced from commands 516 and 517.
COMMAND	*read_device_location*	Command Definition for command 516.
COMMAND	*read_location_description*	Command Definition for command 518.
COMMAND	*write_device_location*	Command Definition for command 517.
COMMAND	*write_location_description*	Command Definition for command 519.

7.11.1.3 IMPORT_MISC_LOCK()

The IMPORT_MISC_LOCK() macro imports items used to implement device locking.

NOTE: This macro can only be used in DDs modeling HART 6 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_LOCK()
```

The list of items imported using the IMPORT_MISC_LOCK() macro are described in Table 46.

Table 46: IMPORT_MISC_LOCK() items

Item Type	Item Name	Description
COMMAND	*lock_device*	Command Definition for command 71.

Item Type	Item Name	Description
COMMAND	*read_lock_device_state*	Command Definition for command 76.

7.11.1.4 IMPORT_MISC_PROCESS_UNIT_TAG()

The IMPORT_MISC_PROCESS_UNIT_TAG() macro imports items used to implement the configuration of process unit tagging.

NOTE: This macro can only be used in DDs modeling HART Universal Rev 7 and Common Practice Rev 10 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_PROCESS_UNIT_TAG()
```

The list of items imported using the IMPORT_MISC_PROCESS_UNIT_TAG() macro is described in Table 47.

Table 47: IMPORT_MISC_PROCESS_UNIT_TAG() items

Item Type	Item Name	Description
VARIABLE	*process_unit_tag*	Data referenced from commands 520 and 521.
COMMAND	*read_process_unit_tag*	Command Definition for command 520.
COMMAND	*write_process_unit_tag*	Command Definition for command 521.

7.11.1.5 IMPORT_MISC_READ_TIME()

The IMPORT_MISC_READ_TIME() macro imports items used to implement the reading of the device's real-time clock.

NOTE: If this macro is used, do NOT use the IMPORT_MISC_READ_WRITE_TIME() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_READ_TIME()
```

The list of items imported using the IMPORT_MISC_READ_TIME() macro is described in Table 48.

Table 48: IMPORT_MISC_READ_TIME() items

Item Type	Item Name	Description
VARIABLE	*current_date*	Data referenced from command 90.
VARIABLE	*current_time*	Data referenced from command 90.
VARIABLE	*last_clock_date*	Data referenced from commands 89 and 90.
VARIABLE	*last_clock_time*	Data referenced from commands 89 and 90.
COMMAND	*read_real_time_clock*	Command Definition for command 90.

7.11.1.6 IMPORT_MISC_READ_WRITE_TIME()

The IMPORT_MISC_READ_WRITE_TIME() macro imports items used to implement the reading and setting of the device's real-time clock.

NOTE: If this macro is used, do NOT use the IMPORT_MISC_READ_TIME() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_READ_WRITE_TIME()
```

The list of items imported using the IMPORT_MISC_READ_WRITE_TIME() macro are described in Table 49.

Table 49: IMPORT_MISC_READ_WRITE_TIME() items

Item Type	Item Name	Description
VARIABLE	*current_date*	Data referenced from command 90.
VARIABLE	*current_time*	Data referenced from command 90.
VARIABLE	*last_clock_date*	Data referenced from commands 89 and 90.
VARIABLE	*last_clock_time*	Data referenced from commands 89 and 90.
COMMAND	*read_real_time_clock*	Command Definition for command 90.
COMMAND	*set_real_time_clock*	Command Definition for command 89.

7.11.1.7 IMPORT_MISC_SI_UNIT()

The IMPORT_MISC_SI_UNIT() macro imports items used to implement SI Unit Control.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these items, add the following line within the common practice miscellaneous import section:

IMPORT_MISC_SI_UNIT()

The list of items imported using the IMPORT_MISC_SI_UNIT() macro is described in Table 50.

Table 50: IMPORT_MISC_SI_UNIT() items

Item Type	Item Name	Description
COMMAND	*read_country_code*	Command Definition for command 512.
COMMAND	*write_country_code*	Command Definition for command 513.

7.11.1.8 IMPORT_MISC_SQUAWK()

The IMPORT_MISC_SQUAWK() macro imports items used to implement locating a device by making it squawk.

NOTE: This macro can only be used in DDs modeling HART 6 or later.

To import these items, add the following line within the common practice miscellaneous import section:

```
IMPORT_MISC_SQUAWK()
```

The list of items imported using the IMPORT_MISC_SQUAWK() macro is described in Table 51.

Table 51: IMPORT_MISC_SQUAWK() items

Item Type	Item Name	Description
COMMAND	*squawk*	Command Definition for command 72.

7.11.1.9 Additional items

Some additional less often to used items are also included in the Common Practice Miscellaneous DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually.

To import an individual item, add the item within the common practice miscellaneous import section (example shown below):

```
COMMAND perform_device_reset;
```

The list of additional items (not covered in any of the previous macros) is described in Table 52.

Table 52: Other Common Practice Misc items

Item Type	Item Name	Description
METHOD	*burn_eeprom*	Deprecated METHOD for committing RAM to EEPROM using command 39.
METHOD	*device_master_reset*	METHOD for resetting the device using commands 42 and 48.
METHOD	*device_self_test*	Method for testing the device using commands 41 and 48.
COMMAND	*eeprom_control*	Deprecated Command Definition for command 39.
VARIABLE	*eeprom_select*	Deprecated Data referenced from command 39.
COMMAND	*findDevice*	Command Definition for command 73. (Available with HART6 and later only)
COMMAND	*flush_delayed_responses*	Command Definition for command 106. (Available with HART6 and later only)
METHOD	*lock_unlock_device*	METHOD for managing device lock using commands 71 and 76. (Available with HART6 and later only)
METHOD	*makeItSquawk*	METHOD for finding the device using command 72. (Available with HART6 and later only)
COMMAND	*perform_device_reset*	Command Definition for command 42.
COMMAND	*perform_self_test*	Command Definition for command 41.
COMMAND	*read_unit_tag_descriptor_date*	Deprecated Command Definition for command 57.
METHOD	*Set_Real_Time_Clock*	METHOD for setting the real-time clock using commands 89 and 90. (Available with HART7 and later only)
VARIABLE	*unit_date*	Deprecated Data referenced from command 57 and 58.
VARIABLE	*unit_descriptor*	Deprecated Data referenced from command 57 and 58.
VARIABLE	*unit_tag*	Deprecated Data referenced from command 57 and 58.
COMMAND	*write_number_of_response_preambles*	Command Definition for command 59.

Item Type	Item Name	Description
COMMAND	*write_unit_tag_descriptor_date*	Deprecated Command Definition for command 58.

7.11.2 DEPENDENCIES

When using the IMPORT_MISC_LOCATION() import, there is a dependency on standard tables:

- IMPORT_TABLES_LOCATION() Described in 7.13.1.15

When using the IMPORT_MISC_LOCK() import, there is a dependency on standard tables:

- IMPORT_TABLES_LOCK() Described in 7.13.1.16

When using the IMPORT_MISC_READ_TIME() import, there is a dependency on standard tables:

- IMPORT_TABLES_READ_TIME() Described in 7.13.1.18

When using the IMPORT_MISC_READ_WRITE_TIME() import, there is a dependency on standard tables:

- IMPORT_TABLES_READ_WRITE_TIME() Described in 7.13.1.19

When using the IMPORT_MISC_SI_UNIT() import, there is a dependency on standard tables:

- IMPORT_TABLES_SI_UNIT() Described in 7.13.1.20

When using the IMPORT_MISC_SQUAWK() import, there is a dependency on standard tables:

- IMPORT_TABLES_SQUAWK() Described in 7.13.1.21

When manually importing COMMAND write_number_of_response_preambles in a DD using the HART rev 5 model, the variable response_preambles must be manually imported from the universal DD. HART rev 6 and 7 will already have this variable imported by using the IMPORT_UNIVERSAL_BASE(u) for values of u at 6 or greater.

7.11.3 REDEFINITIONS

7.11.3.1 Read only Real-time clock

For devices that only support reading of the real-time clock and not setting it (e.g. when using the IMPORT_MISC_READ_TIME() macro), the last_clock_date and last_clock_time variables need to have their handling redefined to be only READ. This is especially typical for WirelessHART devices.

To redefine the last_clock_date and last_clock_time macros, add the following line to the redefinitions section of the HART Common Practice Miscellaneous DD import.

```
REDEFINITIONS
{
    VARIABLE last_clock_date
    {
        REDEFINE HANDLING READ;
    }
    VARIABLE last_clock_time
    {
        REDEFINE HANDLING READ;
    }
}
```

7.11.3.2 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE device_altitude
    {
        SET_DEFAULT(0);
    }
}
```

7.11.3.3 Time Formats

Depending on the measurement types and capabilities of the field device, the time format of certain variables may need to be adjusted to match characteristics of the field device. Variables such as current_time, and others may need to have their time formats adjusted.

To redefine the time format for a variable, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE current_time
    {
        TYPE TIME_VALUE
        {
            REDEFINE TIME_FORMAT %H:%M";
        }
    }
}
```

7.11.3.4 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD device_self_test
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.11.3.5 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE current_time
    {
        REDEFINE HELP "Current Time based upon the real-time clock in the field device.
This device synchronizes its time from the wirelessHART network and cannot be adjusted
directly.";
    }
}
```

7.12 HART Wireless DD

The Wireless DD includes items used to model wirelessHART capabilities. The Wireless binary files are installed in \HCF\DDL\Library\000000\0018\0104.* and 0202.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Wireless1_4.ddl and Wireless2_2.ddl.

NOTE: This DD can only be imported in DDs modeling HART 7 or later.

Only one device revision of the DD (either 1 or 2) should be imported into a device DD depending on which HART Wireless Commands revision is being modelled. To import the wireless DD into a product DD, use one of the following IMPORT statements:

```
/* For HART Rev 7, Wireless Commands verison 1 */
IMPORT STANDARD _WIRELESS, DEVICE_REVISION 1, DD_REVISION 4
{
    IMPORT_WIRELESS_BASE()
}

/* For HART Rev 7, Wireless Commands version 2 */
IMPORT STANDARD _WIRELESS, DEVICE_REVISION 2, DD_REVISION 2
{
    IMPORT_WIRELESS_BASE_V2()
}
```

Import macros have been provided to make it easier to extract the correct items from the Wireless DD. A description of each of these macros is described in 7.12.1.

7.12.1 IMPORTS

7.12.1.1 IMPORT_WIRELESS_BASE()

The IMPORT_WIRELESS_BASE() macro imports mandatory items necessary to model WirelessHART capabilities for the first revision of the HART wireless command specification.

NOTE: If the IMPORT_WIRELESS_BASE_V2() macro is used, do NOT use the IMPORT_WIRELESS_BASE() macro.

To import these required items, add the following line within the wireless import section:

```
IMPORT_WIRELESS_BASE()
```

The list of items imported using the IMPORT_WIRELESS_BASE() macro are described in Table 53.

Table 53: IMPORT_WIRELESS_BASE() items

Item Type	Item Name	Description
VARIABLE	*advertisements*	Data referenced from command 769.
COMMAND	*force_join_mode*	Command Definition for command 771.
VARIABLE	*join_attempts*	Data referenced from command 769.
VARIABLE	*join_key_1*	Data referenced from command 768.
VARIABLE	*join_key_2*	Data referenced from command 768.
VARIABLE	*join_key_3*	Data referenced from command 768.
VARIABLE	*join_key_4*	Data referenced from command 768.
VARIABLE	*join_retry_timer*	Data referenced from command 769.
VARIABLE	*join_shed_time*	Data referenced from commands 771 and 772.

Item Type	Item Name	Description
VARIABLE	*neighbor_count*	Data referenced from command 769.
VARIABLE	*network_id*	Data referenced from commands 773 and 774.
VARIABLE	*network_search_timer*	Data referenced from command 769.
VARIABLE	radio_power_output	Data referenced from commands 797 and 798.
COMMAND	*read_join_mode_configuration*	Command Definition for command 772.
COMMAND	*read_join_status*	Command Definition for command 769.
COMMAND	*read_network_id*	Command Definition for command 774.
COMMAND	read_radio_output_power	Command Definition for command 798.
COMMAND	*read_wireless_module_revision*	Command Definition for command 64512.
VARIABLE	*reserved_bits*	Data referenced from command 769.
VARIABLE	*wireless_module_hardware_revision*	Data referenced from command 64512.
VARIABLE	*wireless_module_software_revision*	Data referenced from command 64512.
VARIABLE	*wireless_module_transmitter_revision*	Data referenced from command 64512.
COMMAND	*write_join_key*	Command Definition for command 768.
COMMAND	*write_network_id*	Command Definition for command 773.

7.12.1.2 IMPORT_WIRELESS_BASE_V2()

The IMPORT_WIRELESS_BASE_V2() macro imports mandatory items necessary to model WirelessHART capabilities for version v of the HART wireless command specification.

NOTE: If the IMPORT_WIRELESS_BASE() macro is used, do NOT use the IMPORT_WIRELESS_BASE_V2() macro.

To import these required items, add the following line within the wireless import section:

```
IMPORT_WIRELESS_BASE_V2()
```

The list of items imported using the IMPORT_WIRELESS_BASE_V2() macro are described in Table 54.

Table 54: IMPORT_WIRELESS_BASE_V2() items

Item Type	Item Name	Description
VARIABLE	*advertisements*	Data referenced from command 769.
VARIABLE	*current_network_id*	Data referenced from commands 773 and 774.
COMMAND	*force_join_mode*	Command Definition for command 771.
VARIABLE	*join_attempts*	Data referenced from command 769.
VARIABLE	*join_key_1*	Data referenced from command 768.
VARIABLE	*join_key_2*	Data referenced from command 768.
VARIABLE	*join_key_3*	Data referenced from command 768.
VARIABLE	*join_key_4*	Data referenced from command 768.
VARIABLE	*join_retry_timer*	Data referenced from command 769.

Item Type	Item Name	Description
VARIABLE	*join_shed_time*	Data referenced from commands 771 and 772.
VARIABLE	*max_join_retries*	Data referenced from commands 771 and 772.
VARIABLE	*neighbor_count*	Data referenced from command 769.
VARIABLE	*network_id*	Data referenced from commands 773 and 774.
VARIABLE	*network_search_timer*	Data referenced from command 769.
VARIABLE	radio_power_output	Data referenced from commands 797 and 798.
COMMAND	*read_join_mode_configuration*	Command Definition for command 772.
COMMAND	*read_join_status*	Command Definition for command 769.
COMMAND	*read_network_id*	Command Definition for command 774.
COMMAND	read_radio_output_power	Command Definition for command 798.
COMMAND	*read_wireless_module_revision*	Command Definition for command 64512.
VARIABLE	*reserved_bits*	Data referenced from command 769.
VARIABLE	*wireless_module_hardware_revision*	Data referenced from command 64512.
VARIABLE	*wireless_module_software_revision*	Data referenced from command 64512.
VARIABLE	*wireless_module_transmitter_revision*	Data referenced from command 64512.
COMMAND	*write_join_key*	Command Definition for command 768.
COMMAND	*write_network_id*	Command Definition for command 773.

7.12.1.3 IMPORT_WIRELESS_DEVICE_ADVERTISING()

The IMPORT_WIRELESS_DEVICE_ADVERTISING() macro imports items necessary to model the capability to enable active advertising in the wireless network. Typically this capability is accessed from the wirelessHART gateway, but the capability is also available to import into a wireless device's DD to provide access to the capability from the device level.

To import active advertising related items, add the following line within the wireless import section:

```
IMPORT_WIRELESS_DEVICE_ADVERTISING()
```

The list of items imported using the IMPORT_WIRELESS_DEVICE_ADVERTISING() macro are described in Table 55.

Table 55: IMPORT_WIRELESS_DEVICE_ADVERTISING() items

Item Type	Item Name	Description
VARIABLE	advertised_neighbors_number	Data referenced from command 770.
VARIABLE	advertising_period	Data referenced from command 770.
COMMAND	request_active_advertising	Command Definition for command 770.
VARIABLE	shed_time	Data referenced from command 770.

7.12.1.4 IMPORT_WIRELESS_DEVICE_CAPABILITIES()

The IMPORT_WIRELESS_DEVICE_CAPABILITIES() macro imports items necessary to read a wireless device's network routing capabilities. Typically, this information is only accessed by the wireless device network manager and does not need to be provided at the device DD level.

To import the capability to access device routing information into a DD, add the following line within the wireless import section:

```
IMPORT_WIRELESS_DEVICE_CAPABILITIES()
```

The list of items imported using the IMPORT_WIRELESS_DEVICE_CAPABILITIED() macro are described in Table 56.

Table 56: IMPORT_WIRELESS_DEVICE_CAPABILITIES() items

Item Type	Item Name	Description
VARIABLE	keep_alive_min_time	Data referenced from command 777.
VARIABLE	max_neighbors	Data referenced from command 777.
VARIABLE	max_packet_buffers	Data referenced from command 777.
VARIABLE	peak_packet_load	Data referenced from command 777.
VARIABLE	peak_packets_per_second	Data referenced from command 777.
COMMAND	read_wireless_device_capabilities	Command Definition for command 777.
VARIABLE	receive_signal_level	Data referenced from command 777.
VARIABLE	recover_time	Data referenced from command 777.

7.12.1.5 IMPORT_WIRELESS_NETWORK_ACCESS()

The IMPORT_WIRELESS_NETWORK_ACCESS() macro imports items necessary to model the capability to control how devices are allowed access to the wirelessHART network. Typically, this capability is implemented only in wirelessHART gateways.

To import this capability into the DD, add the following line within the wireless import section:

```
IMPORT_WIRELESS_NETWORK_ACCESS()
```

The list of items imported using the IMPORT_WIRELESS_NETWORK_ACCESS() macro are described in Table 57.

Table 57: IMPORT_WIRELESS_NETWORK_ACCESS() items

Item Type	Item Name	Description
COMMAND	read_network_access_mode	Command Definition for command 822.
COMMAND	write_network_access_mode	Command Definition for command 821.

7.12.1.6 IMPORT_WIRELESS_NETWORK_TAG()

The IMPORT_WIRELESS_NETWORK_TAG() macro imports items necessary to model the capability to assign and access a wireless network tag. WirelessHART gateways are required to implement this capability. WirelessHART field devices may implement this capability.

To import wireless network tag related items, add the following line within the wireless import section:

```
IMPORT_WIRELESS_NETWORK_TAG()
```

The list of items imported using the IMPORT_WIRELESS_NETWORK_TAG() macro are described in Table 58.

Table 58: IMPORT_WIRELESS_NETWORK_TAG() items

Item Type	Item Name	Description
VARIABLE	network_tag	Data referenced from commands 775 and 776.
COMMAND	read_network_tag	Command Definition for command 776.
COMMAND	write_network_tag	Command Definition for command 775.

7.12.1.7 IMPORT_WIRELESS_NICKNAME()

The IMPORT_WIRELESS_NICKNAME() macro imports items necessary to model the capability to read a device's wireless nickname address. Typically, the wirelessHART network manager is the only application that needs to read a device's nickname. Importing this capability into a DD for end user access is not recommended and causes an "Access Restricted" response if a host tries to access this information when the device is not currently joined to a wirelessHART network.

To import wireless nickname related items, add the following line within the wireless import section:

```
IMPORT_WIRELESS_NICKNAME()
```

The list of items imported using the IMPORT_WIRELESS_NICKNAME() macro are described in Table 59.

Table 59: IMPORT_WIRELESS_NICKNAME() items

Item Type	Item Name	Description
VARIABLE	device_nickname_address	Data referenced from command 781.
COMMAND	read_device_nickname_address	Command Definition for command 781.

7.12.1.8 Additional items

Some additional less often used items are also included in the Wireless DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually. To import an individual item, add the item within the wireless import section (example shown below):

```
METHOD active_advertising_standard;
```

The list of additional items (not covered in any of the previous macros) is described in Table 60.

Table 60: Other Wireless items

Item Type	Item Name	Description
METHOD	active_advertising_standard	METHOD to enable active advertising in the network using command 770.
VARIABLE	battery_life	Depfrecated Data referenced from command 778.
COMMAND	read_battery_life	Deprecated Command Definition for command 778.
COMMAND	write_radio_power_output	Command Definition for command 797.

7.12.2 DEPENDENCIES

When using the IMPORT_WIRELESS_BASE() import, there is a dependency on standard tables:

- IMPORT_TABLES_WIRELESS() Described in 7.13.1.25

When using the IMPORT_WIRELESS_DEVICE_CAPABILITIES() macro, there is a dependency on standard tables:

- IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES() Described in 7.13.1.26

When using the IMPORT_WIRELESS_NETWORK_ACCESS() macro, there is a dependency on standard tables:

- IMPORT_TABLES_WIRELESS_NETWORK_ACCESS() Described in 7.13.1.27

7.12.3 REDEFINITIONS

7.12.3.1 Writeable Radio Power Output

Wireless device DDs will typically only allow end users to read the device's radio transmit power setting. The ability to modify the device's radio transmit power is typically only recommended to be controlled at the gateway/network manager level. To import the capability of writing the radio transmit power, the radio_output_power variable needs to have it's handling redefined to READ & WRITE, and the write_radio_output_power COMMAND needs to be imported.

To redefine the transmit power level to be writeable, add lines similar to the following:

```
REDEFINITIONS
{
    VARIABLE radio_power_output
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.12.3.2 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE wireless_module_hardware_revision
    {
        SET_DEFAULT(1);
    }
}
```

7.12.3.3 Minimum and Maximum Values

It is best practice for all numeric values that are writeable to define an upper and lower limit. This helps to prevent an out of bounds entry from being attempted to be sent to a field device.

To define minimum and maximum values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE radio_power_output
```

```
    {
        REDEFINE MIN_VALUE 0;
        REDEFINE MAX_VALUE 10;
    }
}
```

7.12.3.4 METHODs

In many cases, when importing a METHOD from the HART standard DDs, the logic within the METHOD may need to be customized depending on certain operations of the device.

To redefine the behavior of a METHOD, add lines similar to the following for all applicable methods:

```
REDEFINITIONS
{
    METHOD active_advertising_standard
    {
        REDEFINE DEFINITION
        {
            //include replacement method code here
        }
    }
}
```

7.12.3.5 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE active_advertising_standard
    {
        REDEFINE HELP "To allow wireless field devices to join more quickly, Active
Advertising can be enabled. Active Advertising causes the wireless network to send
advertisements more often. This device also allows active advertising to be enabled from the
local operater panel.";
    }
}
```

7.13 HART Standard Tables DD

The HART Tables DD includes all enumerated and bit-enumerated table definitions. Tables in this DD are often referenced from other standard DDs and will need to be imported into your product DD to satisfy any of the references from those other standard DDs. The HART standard table binary files are installed in \HCF\DDL\Library\000000\0001\1902.*. The corresponding DDL source code for this file is located in HCF\DDL\HCFinfo\Standards\000000\Tabl25_2.ddl. To import the tables DD into a product DD, use the following IMPORT statement:

```
IMPORT STANDARD _TABLES, DEVICE_REVISION 25, DD_REVISION 2
{
    IMPORT_TABLES_BASE()
}
```

Import macros have been provided to make it easier to extract the correct items from the HART standard Tables DD. A description of each of these macros is described in 7.13.1.

7.13.1 IMPORTS

7.13.1.1 IMPORT_TABLES_BASE()

The TABLES_BASE() macro imports a list of all tables required to be included in all DDs.

To import these required tables, add the following line within the common tables import section:

```
IMPORT_TABLES_BASE()
```

The list of items imported using the IMPORT_TABLES_BASE() macro is described in Table 61.

Table 61: IMPORT_TABLES_BASE() items

Item Type	Item Name	Description
VARIABLE	*device_flags*	Data item referenced from commands 0, 11, 21, and 73.
VARIABLE	*device_status*	Data item referenced from ALL commands.
VARIABLE	*loop_alarm_code*	Data item referenced from command 15.
VARIABLE	*physical_signaling_code*	Data item referenced from commands 0, 11, 21, and 73.
VARIABLE	*transfer_function*	Data item referenced from commands 15, and 47.
VARIABLE	*write_protect*	Data item referenced from command 15.

7.13.1.2 IMPORT_TABLES_BURST_REF_ARRAY(u, *b*)

The IMPORT_TABLES_BURST_REF_ARRAY(u, b) macro imports a list of tables used for burst mode operation. Burst Mode Operation in the standard DDs can be modelled using either value arrays, or reference arrays. Different sets of burst related items are necessary depending on the HART universal revision that the DD is modeling. The parameter ‘u’ is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used. This macro imports the reference array model, which is limited to a maximum of 4 burst messages. The parameter ‘b’ is used to specify how many burst messages are to be supported (allowable values are from 1 to 4). The same value of b has to be used in all macros that are used that call for a parameter ‘b’ to be used. The reference array model may be preferred for compatibility in older host applications.

NOTE: If this is macro used, then do NOT use the IMPORT_TABLES_BURST_VAL_ARRAY(u) macro.

NOTE: When Universal Revision (u) is less than 7, only 1 burst message (b) is allowed to be modeled.

To import these tables required for burst mode operation, add the following line within the common tables import section (Note – This example if for a device that implements 3 burst messages):

```
IMPORT_TABLES_BURST_REF_ARRAY(7,3)
```

The list of items imported using the IMPORT_TABLES_BURST_REF_ARRAY(u, b) macro is described in Table 62. The "Universal Revisions" column identifies for which value of u a variable will be imported or not. The "b Values" column identifies for which values of 'b' a variable will be imported or not.

Table 62: IMPORT_TABLES_BURST_REF_ARRAY(u, b) items

Item Type	Item Name	Universal Revisions	b Values	Description
VARIABLE	*burst_message_trigger_mode*	7	≥ 1	Data item referenced from commands 104 and 105. This variable is used for burst message index 0.
VARIABLE	*burst_mode_select_0*	5, 6, 7	≥ 1	Data item referenced from commands 105 and 109. This variable is used for burst message index 0.
VARIABLE	*burst_mode_select_1*	7	≥ 2	Data item referenced from commands 105 and 109. This variable is used for burst message index 1.
VARIABLE	*burst_mode_select_2*	7	≥ 3	Data item referenced from commands 105 and 109. This variable is used for burst message index 2.
VARIABLE	*burst_mode_select_3*	7	≥ 4	Data item referenced from commands 105 and 109. This variable is used for burst message index 3.
VARIABLE	*burst_trigger_class_1*	7	≥ 2	Data item referenced from commands 104 and 105. This variable is used for burst message index 1.
VARIABLE	*burst_trigger_class_2*	7	≥ 3	Data item referenced from commands 104 and 105. This variable is used for burst message index 2.
VARIABLE	*burst_trigger_class_3*	7	≥ 4	Data item referenced from commands 104 and 105. This variable is used for burst message index 3.
VARIABLE	*burst_trigger_classification*	7	≥ 1	Data item referenced from commands 104 and 105. This variable is used for burst message index 0.
VARIABLE	*burst_trigger_mode_1*	7	≥ 2	Data item referenced from commands 104 and 105. This variable is used for burst message index 1.
VARIABLE	*burst_trigger_mode_2*	7	≥ 3	Data item referenced from commands 104 and 105. This variable is used for burst message index 2.

Item Type	Item Name	Universal Revisions	b Values	Description
VARIABLE	*burst_trigger_mode_3*	7	≥ 4	Data item referenced from commands 104 and 105. This variable is used for burst message index 3.
VARIABLE	*burst_trigger_units*	7	≥ 1	Data item referenced from commands 104 and 105. This variable is used for burst message index 0.
VARIABLE	*burst_trigger_units_1*	7	≥ 2	Data item referenced from commands 104 and 105. This variable is used for burst message index 1.
VARIABLE	*burst_trigger_units_2*	7	≥ 3	Data item referenced from commands 104 and 105. This variable is used for burst message index 2.
VARIABLE	*burst_trigger_units_3*	7	≥ 4	Data item referenced from commands 104 and 105. This variable is used for burst message index 3.

7.13.1.3 IMPORT_TABLES_BURST_VAL_ARRAY(u)

The IMPORT_TABLES_BURST_VAL_ARRAY(u) macro imports a list of tables used for burst mode operation. Burst Mode Operation in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the value array model, which is recommended because it allows any number of burst messages to be modelled without creating unique variable copies for each supported burst message. Simply changing the number of elements in the value array can change how many unique burst messages are being modelled in the DD.

Different sets of burst related items are necessary depending on the HART universal revision that the DD is modeling. The parameter 'u' is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter 'u' to be used.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_BURST_REF_ARRAY(u, b) macro.

To import these tables required for burst mode operation, add one of the following lines within the common tables import section:

```
IMPORT_TABLES_BURST_VAL_ARRAY(5)
IMPORT_TABLES_BURST_VAL_ARRAY(6)
IMPORT_TABLES_BURST_VAL_ARRAY(7)
```

The list of items imported using the IMPORT_TABLES_BURST_VAL_ARRAY(u) macro is described in Table 63. The "Universal Revisions" column identifies for which value of u a variable will be imported or not.

Table 63: IMPORT_TABLES_BURST_VAL_ARRAY(u) items

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*burst_message_trigger_mode*	7	Data item referenced from commands 104 and 105.
VARIABLE	*burst_mode_select*	5, 6, 7	Data item referenced from commands 105 and 109.
VARIABLE	*burst_trigger_classification*	7	Data item referenced from commands 104 and 105.

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*burst_trigger_units*	7	Data item referenced from commands 104 and 105.

7.13.1.4 IMPORT_TABLES_CONDENSED_STATUS(n)

The IMPORT_TABLES_CONDENSED_STATUS(n) macro imports a list of tables used for condensed status capability. Different sets of tables are necessary depending on how many bytes are being modelled in the command 48 response. When using the IMPORT_TABLES_CONDENSED_STATUS(n) macro, the parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the 2 bytes for response code and device status). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used. The minimum allowed value for n is 10 for devices that implement condensed status.

NOTE: This macro can only be used in DDs modeling HART 7 and Common Practice 10 or later.

To import these tables required for devices that implement condensed status, add the following line within the common tables import section (Note – A device implementing 10 additional status bytes in command 48 is shown in this example):

```
IMPORT_TABLES_CONDENSED_STATUS(10)
```

The list of items imported using the IMPORT_TABLES_CONDENSED_STATUS(n) macro is described in Table 64. The "n Values" column identifies for which values of 'n' a variable will be imported or not.

Table 64: IMPORT_TABLES_CONDENSED_STATUS(n) items

Item Type	Item Name	n Values	Description
VARIABLE	*simulate_bit_value*	≥ 10	Local variable referenced from command 527.
VARIABLE	*status_simulate_mode*	≥ 10	Local variable referenced from command 526.

Item Type	Item Name	n Values	Description
VARIABLE	*statusmap_variable_xxx* xxx values 000 – 007 correspond with the 8 bits in device_status xxx values 008 - 015 correspond with the 8 bits in device_specific_status_0 xxx values 016 – 023 correspond with the 8 bits in device_specific_status_1 xxx values 024 – 031 correspond with the 8 bits in device_specific_status_2 xxx values 032 – 039 correspond with the 8 bits in device_specific_status_3 xxx values 040 – 047 correspond with the 8 bits in device_specific_status_4 xxx values 048 – 055 correspond with the 8 bits in device_specific_status_5 xxx values 056 – 063 correspond with the 8 bits in extended_fld_device_status xxx values 064 – 071 correspond with the 8 bits in operatingMode xxx values 072 – 079 correspond with the 8 bits in standardized_status_0 xxx values 080 – 087 correspond with the 8 bits in standardized_status_1	≥ 10	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.

Item Type	Item Name	n Values	Description
VARIABLE	*statusmap_variable_xxx* xxx values 088 – 095 correspond with the 8 bits in analog_channel_saturated1	≥ 11	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 096 – 103 correspond with the 8 bits in standardized_status_2	≥ 12	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 104 – 111 correspond with the 8 bits in standardized_status_3	≥ 13	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 112 – 119 correspond with the 8 bits in analog_channel_fixed1	≥ 14	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 120 – 127 correspond with the 8 bits in device_specific_status_14	≥ 15	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 128 – 135 correspond with the 8 bits in device_specific_status_15	≥ 16	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 136 – 143 correspond with the 8 bits in device_specific_status_16	≥ 17	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 144 – 151 correspond with the 8 bits in device_specific_status_17	≥ 18	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 152 – 159 correspond with the 8 bits in device_specific_status_18	≥ 19	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.

Item Type	Item Name	n Values	Description
VARIABLE	*statusmap_variable_xxx* xxx values 160 – 167 correspond with the 8 bits in device_specific_status_19	≥ 20	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 168 – 175 correspond with the 8 bits in device_specific_status_20	≥ 21	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 176 – 183 correspond with the 8 bits in device_specific_status_21	≥ 22	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 184 – 191 correspond with the 8 bits in device_specific_status_22	≥ 23	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 192 – 199 correspond with the 8 bits in device_specific_status_23	≥ 24	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.
VARIABLE	*statusmap_variable_xxx* xxx values 200 – 207 correspond with the 8 bits in device_specific_status_24	25	Data item referenced from commands 523 and 524. There is one statusmap variable per bit in the status bytes returned from command 48. xxx in the variable name starts at 000 for bit 0 of device_status and ends at 207 for bit 7 of device_specific_status_24.

7.13.1.5 IMPORT_TABLES_DEVELOPER()

The IMPORT_TABLES_DEVELOPER() macro imports a list of all common tables that the DD developer will need to model their own device specific variables after.

To import these required tables, add the following line within the common tables import section:

```
IMPORT_TABLES_DEVELOPER()
```

The list of items imported using the IMPORT_TABLES_DEVELOPER() macro are described in Table 65.

Table 65: IMPORT_TABLES_DEVELOPER() items

Item Type	Item Name	Description
VARIABLE	capture_mode_codes	Local variable defines the list of enumerations used for device variable capture modes.

Item Type	Item Name	Description
VARIABLE	dev_var_property_codes	Local variable defines the list of enumerations used for device variable properties.
VARIABLE	device_family_status	Local variable defines the list of enumerations used for device family status bits.
VARIABLE	device_variable_classification_codes	Local variable defines the list of enumerations used for device variable classifications.
VARIABLE	*device_variable_code_codes*	Local variable defines the list of enumerations used device variable codes (i.e. the HART standard variables like Percent of Range, Loop Current, and others).
VARIABLE	device_variable_family_codes	Local variable defines the list of enumerations used for device variable families.
VARIABLE	limit_status	Local variable defines the list of enumerations used for limit status bits.
VARIABLE	material_code	Local variable defines the list of enumerations used for materials of construction.
VARIABLE	process_data_status	Local variable defines the list of enumerations used for process data status bits.
VARIABLE	trim_point_codes	Local variable defines the list of enumerations used for device variable trim options.
VARIABLE	write_device_variable_codes	Local variable defines the list of enumerations used for writing device variable values.

7.13.1.6 IMPORT_TABLES_DEVICE_TYPES_GROUP1()

The IMPORT_TABLES_DEVICE_TYPES_GROUP1() macro imports the first half of the list of all manufacturer's device types that are in the HART common Tables DD. The list is broken into 2 halves due to a limitation in the length a MACRO can be. Most common field devices due not need to import the list of device types. IO systems that can connect to subdevices will should import these in order to be able to identify device types by name.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables for IO systems, add the following line within the common tables import section:

```
IMPORT_TABLES_DEVICE_TYPES_GROUP1()
```

Due to the large number of manufacturer's device types, the list of items is not included in this document.

7.13.1.7 IMPORT_TABLES_DEVICE_TYPES_GROUP2()

The IMPORT_TABLES_DEVICE_TYPES_GROUP2() macro imports the second half of the list of all manufacturer's device types that are in the HART common Tables DD. The list is broken into 2 halves due to a limitation in the length a MACRO can be. Most common field devices due not need to import the list of device types. IO systems that can connect to subdevices will should import these in order to be able to identify device types by name.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables for IO systems, add the following line within the common tables import section:

```
IMPORT_TABLES_DEVICE_TYPES_GROUP2()
```

Due to the large number of manufacturer's device types, the list of items is not included in this document.

7.13.1.8 IMPORT_TABLES_EVENT_MANAGER()

The IMPORT_TABLES_EVENT_MANAGER() macro imports a list of tables used for event notification managers.

NOTE: This macro can only be used in DDs modeling HART rev 7 and Common Practice rev 10 or later.

To import these tables required for event managers, add the following line within the common tables import section:

```
IMPORT_TABLES_EVENT_MANAGER()
```

The list of items imported using the IMPORT_TABLES_EVENT_MANAGER() macro is described in Table 66.

Table 66: IMPORT_TABLES_EVENT_MANAGER() items

Item Type	Item Name	Description
VARIABLE	*event_manager_registration_control*	Data item referenced from command 514.
VARIABLE	*event_manager_registration_status*	Data item referenced from command 515.

7.13.1.9 IMPORT_TABLES_EVENT_REF_ARRAY(*e*)

The IMPORT_TABLES_EVENT_REF_ARRAY(e) macro imports a list of tables used for event notification operation. Event notification in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the reference array model, which is limited to a maximum of 2 event notifications. The parameter 'e' is used to specify how many event notifications are to be supported (allowable values are from 1 to 2). The same value of e has to be used in all macros that are used that call for a parameter 'e' to be used. The reference array model may be preferred for compatibility in older host applications.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_EVENT_VAL_ARRAY() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for event notification, add the following line within the common tables import section (Note – This example if for a device that implements 1 event notification):

```
IMPORT_TABLES_EVENT_REF_ARRAY(1)
```

The list of items imported using the IMPORT_TABLES_EVENT_REF_ARRAY(e) macro is described in Table 67. The "e Values" column identifies for which values of 'e' a variable will be imported or not.

Table 67: IMPORT_TABLES_EVENT_REF_ARRAY(e) items

Item Type	Item Name	e Values	Description
VARIABLE	*analog_channel_fixed1_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*analog_channel_fixed1_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*analog_channel_fixed1_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*analog_channel_fixed1_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.

Item Type	Item Name	e Values	Description
VARIABLE	*analog_channel_saturated1_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*analog_channel_saturated1_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*analog_channel_saturated1_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*analog_channel_saturated1_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*std_operating_mode_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*dev_operating_mode_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*dev_operating_mode_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*dev_operating_mode_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*device_status_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*device_status_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*device_status_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*device_status_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*event_notification_control_n*	≥ 1	Data item referenced from commands 115 and 118. This variable is used for event notification index 0.
VARIABLE	*event_notification_control_n_1*	≥ 2	Data item referenced from commands 115 and 118. This variable is used for event notification index 1.
VARIABLE	*event_status*	≥ 1	Data item referenced from command 115. This variable is used for event notification index 0.

Item Type	Item Name	e Values	Description
VARIABLE	*event_status_1*	≥ 2	Data item referenced from command 115. This variable is used for event notification index 1.
VARIABLE	*extended_fld_device_status_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*extended_fld_device_status_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*extended_fld_device_status_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*extended_fld_device_status_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*standardized_status_0_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*standardized_status_0_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*standardized_status_0_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*standardized_status_0_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*standardized_status_1_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*standardized_status_1_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*standardized_status_1_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*standardized_status_1_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*standardized_status_2_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*standardized_status_2_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.

Item Type	Item Name	e Values	Description
VARIABLE	*standardized_status_2_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*standardized_status_2_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.
VARIABLE	*standardized_status_3_latched_value*	≥ 1	Data item referenced from command 119. This variable is used for event notification index 0.
VARIABLE	*standardized_status_3_latched_value_1*	≥ 2	Data item referenced from command 119. This variable is used for event notification index 1.
VARIABLE	*standardized_status_3_mask*	≥ 1	Data item referenced from commands 115 and 116. This variable is used for event notification index 0.
VARIABLE	*standardized_status_3_mask_1*	≥ 2	Data item referenced from commands 115 and 116. This variable is used for event notification index 1.

7.13.1.10 IMPORT_TABLES_EVENT_VAL_ARRAY()

The IMPORT_TABLES_EVENT_VAL_ARRAY() macro imports a list of tables used for event notification operation. Event notification in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the value array model, which is recommended because it allows any number of event notification messages to be modelled without creating unique variable copies for each supported event notification. Simply changing the number of elements in the value array can change how many unique event notifications are being modelled in the DD.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_EVENT_REF_ARRAY(e) macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for event notification, add the following line within the common tables import section:

```
IMPORT_TABLES_EVENT_VAL_ARRAY()
```

The list of items imported using the IMPORT_TABLES_EVENT_VAL_ARRAY() macro is described in Table 68Table 68: IMPORT_TABLES_EVENT_VAL_ARRAY() items.

Table 68: IMPORT_TABLES_EVENT_VAL_ARRAY() items

Item Type	Item Name	Description
VARIABLE	*analog_channel_fixed1_latched_value*	Data item referenced from command 119.
VARIABLE	*analog_channel_fixed1_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*analog_channel_saturated1_latched_value*	Data item referenced from command 119.
VARIABLE	*analog_channel_saturated1_mask*	Data item referenced from commands 115 and 116.

Item Type	Item Name	Description
VARIABLE	*std_operating_mode_latched_value*	Data item referenced from command 119.
VARIABLE	*dev_operating_mode_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*device_status_latched_value*	Data item referenced from command 119, uses device_status local variable for the enumeration list.
VARIABLE	*device_status_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*event_notification_control_n*	Data item referenced from commands 115 and 118.
VARIABLE	*event_status*	Data item referenced from command 115.
VARIABLE	*extended_fld_device_status_latched_value*	Data item referenced from command 119.
VARIABLE	*extended_fld_device_status_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*standardized_status_0_latched_value*	Data item referenced from command 119.
VARIABLE	*standardized_status_0_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*standardized_status_1_latched_value*	Data item referenced from command 119.
VARIABLE	*standardized_status_1_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*standardized_status_2_latched_value*	Data item referenced from command 119.
VARIABLE	*standardized_status_2_mask*	Data item referenced from commands 115 and 116.
VARIABLE	*standardized_status_3_latched_value*	Data item referenced from command 119.
VARIABLE	*standardized_status_3_mask*	Data item referenced from commands 115 and 116.

7.13.1.11 IMPORT_TABLES_HART(u)

The IMPORT_TABLES_HART(u) macros imports a list of all tables that the DD developer will need to model certain device specific variables after. Different sets of tables are necessary depending on the HART universal revision that the DD is modeling. The parameter ‘u’ is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used.

To import these required tables, add one of the following lines within the common tables import section:

```
IMPORT_TABLES_HART(7)
IMPORT_TABLES_HART(6)
IMPORT_TABLES_HART(5)
```

The list of items imported using the IMPORT_TABLES_HART(u) macros are described in Table 69. The “Universal Revisions” column indicates which variables are imported for each of the HART universal revisions.

Table 69: IMPORT_TABLES_HART(u) items

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*device_profile*	7	Data item referenced from commands 0, 11, 21, and 73.
VARIABLE	*device_type*	7	Data item referenced from commands 0, 11, 21, and 73. Note – for HART 5 & 6, this variable is imported from the Universal DD See 7.1
VARIABLE	*loop_current_mode*	6, 7	Data item referenced from commands 6, and 7.
VARIABLE	*loop_flags*	6, 7	Data item referenced from command 15.
VARIABLE	*manufacturer_id*	7	Data item referenced from commands 0, 11, 21, and 73. Note – for HART 5 & 6, this variable is imported from the Universal DD See 7.1
VARIABLE	*private_label_distributor*	7	Data item referenced from commands 0, 11, 21, and 73. Note – for HART 5 & 6, this variable is imported from the Universal DD See 7.1.

7.13.1.12 IMPORT_TABLES_IO_ADAPTER_V11()

The IMPORT_TABLES_IO_ADAPTER_V11() macro imports a list of tables used for IO Systems capability as was updated starting with Common Practice rev 11.

NOTE: If the IMPORT_TABLES_IO_ADAPTER_V9() macro is used, do NOT use the IMPORT_TABLES_IO_ADAPTER_V11() macro.

NOTE: This macro can only be used in DDs modeling HART Universal rev 7 and Common Practice rev 10 or later.

To import these tables required for IO Adapters, add the following line within the common tables import section:

```
IMPORT_TABLES_IO_ADAPTER_V11()
```

The list of items imported using the IMPORT_TABLES_IO_ADAPTER_V11() macro is described in Table 70.

Table 70: IMPORT_TABLES_IO_ADAPTER_V11() items

Item Type	Item Name	Description
VARIABLE	*master_mode*	Data item referenced from commands 74 and 87.
VARIABLE	*subdev_dev_profile*	Data item referenced from command 84.
VARIABLE	*subdev_distributor*	Data item referenced from command 84.
VARIABLE	*subdev_expanded_device_type*	Data item referenced from commands 84, 529, and 530.
VARIABLE	*subdev_manufacturer_id*	Data item referenced from commands 84, 529, and 530.

7.13.1.13 IMPORT_TABLES_IO_ADAPTER_V9()

The IMPORT_TABLES_IO_ADAPTER_V9() macro imports a list of tables used for IO Systems capability as it was originally defined starting in Common Practice rev 9.

NOTE: If the IMPORT_TABLES_IO_ADAPTER_V11() macro is used, do NOT use the IMPORT_TABLES_IO_ADAPTER_V9() macro.

NOTE: This macro can only be used in DDs modeling HART rev 7 and prior to Common Practice rev 10.

To import these tables required for IO Adapters, add the following line within the common tables import section:

```
IMPORT_TABLES_IO_ADAPTER_V9()
```

The list of items imported using the IMPORT_TABLES_IO_ADAPTER_V9() macro is described in Table 71.

Table 71: IMPORT_TABLES_IO_ADAPTER_V9() items

Item Type	Item Name	Description
VARIABLE	*master_mode*	Data item referenced from commands 74 and 87.
VARIABLE	*subdev_expanded_device_type*	Data item referenced from commands 84, 529, and 530.
VARIABLE	*subdev_manufacturer_id*	Data item referenced from commands 84, 529, and 530.

7.13.1.14 IMPORT_TABLES_IO_ASSIGN()

The IMPORT_TABLES_IO_ASSIGN() macro imports a list of tables used for IO Adapter List assignment capability.

NOTE: This macro can only be used in DDs modeling HART rev 7 and Common Practice rev 10 or later.

To import these tables required for IO Adapter devices with list assignment capability, add the following line within the common tables import section:

IMPORT_TABLES_IO_ASSIGN()

The list of items imported using the IMPORT_TABLES_IO_ASSIGN() macro is described in Table 72.

Table 72: IMPORT_TABLES_IO_ASSIGN() items

Item Type	Item Name	Description
VARIABLE	*subdev_assign_status*	Data item referenced from commands 528 and 529.
VARIABLE	*subdevice_transfer_code*	Local variable referenced from command 531.

7.13.1.15 IMPORT_TABLES_LOCATION()

The IMPORT_TABLES_LOCATION() macro imports a list of tables used for location coordinate functionality.

NOTE: This macro can only be used in DDs modeling HART 7 or later with Common Practice 10 or later.

To import these tables required for location coordinates, add the following line within the common tables import section:

```
IMPORT_TABLES_LOCATION()
```

The list of items imported using the IMPORT_TABLES_LOCATION() macro is described in Table 73.

Table 73: IMPORT_TABLES_LOCATION() items

Item Type	Item Name	Description
VARIABLE	*location_mechanism*	Data item referenced from commands 516 and 517.

7.13.1.16 IMPORT_TABLES_LOCK()

The IMPORT_TABLES_LOCK() macro imports a list of tables used for the device locking mechanism.

NOTE: This macro can only be used in DDs modeling HART 6 or later.

To import these tables required for device lock feature, add the following line within the common tables import section:

```
IMPORT_TABLES_LOCK()
```

The list of items imported using the IMPORT_TABLES_LOCK() macro is described in Table 74.

Table 74: IMPORT_TABLES_LOCK() items

Item Type	Item Name	Description
VARIABLE	*lock_device_code*	Data item referenced from command 71.
VARIABLE	*lock_device_status_code*	Data item referenced from command 76.

7.13.1.17 IMPORT_TABLES_MULTI_AO(u)

The IMPORT_TABLES_MULTI_AO(u) macro imports a list of tables used for devices that support more than 1 analog output. Different sets of tables are necessary depending on the HART universal revision that the DD is modeling. The parameter ‘u’ is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter ‘u’ to be used.

To import these tables required for multi-AO devices, add one of the following lines within the common tables import section:

```
IMPORT_TABLES_MULTI_AO(7)
IMPORT_TABLES_MULTI_AO(6)
IMPORT_TABLES_MULTI_AO(5)
```

The list of items imported using the IMPORT_TABLES_MULTI_AO(u) macro is described in Table 75. The “Universal Revisions” column indicates which variables are imported for each of the HART universal revisions.

Table 75: IMPORT_TABLES_MULTI_AO(u) items

Item Type	Item Name	Universal Revisions	Description
VARIABLE	*QVanalogChannelAlarmCode*	5, 6, 7	Data item referenced from commands 63 and 100.
VARIABLE	*QVanalogChannelFlags*	6, 7	Data item referenced from command 63.
VARIABLE	*QVtransferFunction*	5, 6, 7	Data item referenced from commands 63 and 69.
VARIABLE	*SVanalogChannelAlarmCode*	5, 6, 7	Data item referenced from commands 63 and 100.
VARIABLE	*SVanalogChannelFlags*	6, 7	Data item referenced from command 63.
VARIABLE	*SVtransferFunction*	5, 6, 7	Data item referenced from commands 63 and 69.
VARIABLE	*TVanalogChannelAlarmCode*	5, 6, 7	Data item referenced from commands 63 and 100.
VARIABLE	*TVanalogChannelFlags*	6, 7	Data item referenced from command 63.
VARIABLE	*TVtransferFunction*	5, 6, 7	Data item referenced from commands 63 and 69.

7.13.1.18 IMPORT_TABLES_READ_TIME()

The IMPORT_TABLES_READ_TIME() macro imports a list of tables used for the current time mechanism. This macro is used for implementations that support only reading but not setting the time in the field device.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_READ_WRITE_TIME() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for time functionality, add the following line within the common tables import section:

```
IMPORT_TABLES_READ_TIME()
```

The list of items imported using the IMPORT_TABLES_READ_TIME() macro is described in Table 76.

Table 76: IMPORT_TABLES_READ_TIME() items

Item Type	Item Name	Description
VARIABLE	*real_time_clock_flag*	Data item referenced from command 90.

7.13.1.19 IMPORT_TABLES_READ_WRITE_TIME()

The IMPORT_TABLES_READ_WRITE_TIME() macro imports a list of tables used for the current time mechanism. This macro is used for implementations that support reading and setting the time in the field device.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_READ_TIME() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for time functionality, add the following line within the common tables import section:

```
IMPORT_TABLES_READ_WRITE_TIME()
```

The list of items imported using the IMPORT_TABLES_READ_WRITE_TIME() macro is described in Table 77.

Table 77: IMPORT_TABLES_READ_WRITE_TIME() items

Item Type	Item Name	Description
VARIABLE	*real_time_clock_flag*	Data item referenced from command 90.
VARIABLE	*time_set*	Data item referenced from command 89.

7.13.1.20 IMPORT_TABLES_SI_UNIT()

The IMPORT_TABLES_SI_UNIT() macro imports a list of tables used for the SI Unit mechanism.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for SI Unit feature, add the following line within the common tables import section:

IMPORT_TABLES_SI_UNIT()

The list of items imported using the IMPORT_TABLES_SI_UNIT() macro is described in Table 78.

Table 78: IMPORT_TABLES_SI_UNIT() items

Item Type	Item Name	Description
VARIABLE	*country_code*	Data item referenced from commands 512 and 513.
VARIABLE	*si_control*	Data item referenced from commands 512 and 513.

7.13.1.21 IMPORT_TABLES_SQUAWK()

The IMPORT_TABLES_SQUAWK() macro imports a list of tables used for the Squawk mechanism. These tables are only necessary for devices that have implemented the updated capability of the squawk mechanism introduced starting in Common Practice rev 11 DD.

NOTE: This macro can only be used in DDs modeling HART Universal Rev 7 and Common Practice Rev 10 or later.

To import these tables required for Squawk feature, add the following line within the common tables import section:

IMPORT_TABLES_SQUAWK()

The list of items imported using the IMPORT_TABLES_SQUAWK() macro is described in Table 79.

Table 79: IMPORT_TABLES_SQUAWK() items

Item Type	Item Name	Description
VARIABLE	*squawk_control*	Data item referenced from command 72.

7.13.1.22 IMPORT_TABLES_STANDARD_STATUS(u, *n*)

The IMPORT_TABLES_STANDARD_STATUS(u, *n)* macro imports a list of tables used for additional device status. Different sets of tables are necessary depending on which universal revision is used, and how many bytes are being modelled in the command 48 response. When using the IMPORT_TABLES_STANDARD_STATUS(u, n) macro, The parameter 'u' is used to specify which universal HART revision should be modelled. The same value of u has to be used in all macros that are used that call for a parameter 'u' to be used. The parameter 'n' is used to specify the number of command 48 bytes to be implemented (not including the device status and response code). The same value of n has to be used in all macros that are used that call for a parameter 'n' to be used.

To import these required tables, add the following line within the common tables import section (Note – A device implementing HART7 with 9 "additional status" bytes in command 48 is shown in this example):

```
IMPORT_TABLES_STATUS(7, 9)
```

The list of items imported using the IMPORT_TABLES_STATUS(u, n) macro is described in Table 80. The "Universal Revisions" column identifies which universal revisions a variable will be imported or not. The "n Values" column identifies for which values of 'n' a variable will be imported or not.

Table 80: IMPORT_TABLES_STATUS(u, n) items

Item Type	Item Name	Universal Revisions	n Values	Description
VARIABLE	*analog_channel_fixed1*	5, 6, 7	≥ 14	Data item referenced from command 48.
VARIABLE	*analog_channel_saturated1*	5, 6, 7	≥ 11	Data item referenced from command 48.

Item Type	Item Name	Universal Revisions	n Values	Description
VARIABLE	*extended_fld_device_status*	6, 7	≥ 7	Data item referenced from commands 0, 9, 11, 21, 48, and 73.
VARIABLE	*standardized_status_0*	7	≥ 9	Data item referenced from command 48.
VARIABLE	*standardized_status_1*	7	≥ 10	Data item referenced from command 48.
VARIABLE	*standardized_status_2*	7	≥ 12	Data item referenced from command 48.
VARIABLE	*standardized_status_3*	7	≥ 13	Data item referenced from command 48.

7.13.1.23 IMPORT_TABLES_TREND_REF_ARRAY()

The IMPORT_TABLES_TREND_REF_ARRAY() macro imports a list of tables used for the variable trending mechanism. Variable trending in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the reference array model. The reference array model may be preferred for compatibility in older host applications.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_TREND_VAL_ARRAY() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for variable trending, add the following line within the common tables import section:

```
IMPORT_TABLES_TREND_REF_ARRAY()
```

The list of items imported using the IMPORT_TABLES_TREND_REF_ARRAY() macro is described in Table 81.

Table 81: IMPORT_TABLES_TREND_REF_ARRAY() items

Item Type	Item Name	Description
VARIABLE	*trend_classification*	Data item referenced from commands 93.
VARIABLE	*trend_control*	Data item referenced from commands 91 and 92.
VARIABLE	*trend_digital_units*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_1*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_2*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_3*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_4*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_5*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_6*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_7*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_8*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_9*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_10*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_data_quality_11*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_1*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_2*	Data item referenced from command 93.

Item Type	Item Name	Description
VARIABLE	*std_trend_value_0_device_family_status_3*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_4*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_5*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_6*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_7*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_8*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_9*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_10*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_device_family_status_11*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_1*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_2*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_3*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_4*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_5*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_6*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_7*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_8*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_9*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_10*	Data item referenced from command 93.
VARIABLE	*std_trend_value_0_limit_status_11*	Data item referenced from command 93.
VARIABLE	*trend_value_data_quality*	Data item referenced from command 93.
VARIABLE	*std_trend_value_device_family_status*	Data item referenced from command 93.
VARIABLE	*trend_value_limit_status*	Data item referenced from command 93.

7.13.1.24 IMPORT_TABLES_TREND_VAL_ARRAY()

The IMPORT_TABLES_TREND_VAL_ARRAY() macro imports a list of tables used for the variable trending mechanism. Variable trending in the standard DDs can be modelled using either value arrays, or reference arrays. This macro imports the value array model, which is recommended because it allows any number of trend values to be modelled without creating unique variable copies for each supported event notification.

NOTE: If this macro is used, then do NOT use the IMPORT_TABLES_TREND_REF_ARRAY() macro.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for variable trending, add the following line within the common tables import section:

```
IMPORT_TABLES_TREND_VAL_ARRAY()
```

The list of items imported using the IMPORT_TABLES_TREND_VAL_ARRAY() macro is described in Table 82.

Table 82: IMPORT_TABLES_TREND_VAL_ARRAY() items

Item Type	Item Name	Description
VARIABLE	*trend_classification*	Data item referenced from commands 93.
VARIABLE	*trend_control*	Data item referenced from commands 91 and 92.
VARIABLE	*trend_digital_units*	Data item referenced from command 93.
VARIABLE	*trend_value_data_quality*	Data item referenced from command 93.
VARIABLE	*std_trend_value_device_family_status*	Data item referenced from command 93.
VARIABLE	*trend_value_limit_status*	Data item referenced from command 93.

7.13.1.25 IMPORT_TABLES_WIRELESS()

The IMPORT_TABLES_WIRELESS() macro imports a list of tables used for wirelessHART capability.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables required for wireless devices, add the following line within the common tables import section:

```
IMPORT_TABLES_WIRELESS()
```

The list of items imported using the IMPORT_TABLES_WIRELESS() macro is described in Table 83.

Table 83: IMPORT_TABLES_WIRELESS() items

Item Type	Item Name	Description
VARIABLE	*join_mode*	Data item referenced from commands 771 and 772.
VARIABLE	*join_status*	Data item referenced from command 769.
VARIABLE	*wireless_mode*	Data item referenced from command 769.
VARIABLE	*wireless_module_device_type*	Data item referenced from command 64512.
VARIABLE	*wireless_module_manufacturer_id*	Data item referenced from command 64512.

7.13.1.26 IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES()

The IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES() macro imports a list of tables used for reading the routing capabilities from a wirelessHART device.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables, add the following line within the common tables import section:

```
IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES()
```

The list of items imported using the IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES() macro is described in Table 84.

Table 84: IMPORT_TABLES_WIRELESS_DEVICE_CAPABILITIES() items

Item Type	Item Name	Description
VARIABLE	*power_source*	Data item referenced from command 777.
VARIABLE	*wireless_capability*	Data item referenced from command 777.

7.13.1.27 IMPORT_TABLES_WIRELESS_NETWORK_ACCESS()

The IMPORT_TABLES_WIRELESS_NETWORK_ACCESS() macro imports tables used for accessing the wireless device's network access mode.

NOTE: This macro can only be used in DDs modeling HART 7 or later.

To import these tables, add the following line within the common tables import section:

```
IMPORT_TABLES_WIRELESS_NETWORK_ACCESS()
```

The list of items imported using the IMPORT_TABLES_WIRELESS_NETWORK_ACCESS() macro is described in Table 85.

Table 85: IMPORT_TABLES_WIRELESS_NETWORK_ACCESS() items

Item Type	Item Name	Description
VARIABLE	*network_access_mode*	Data item referenced from commands 821 and 822.

7.13.1.28 Additional items

Some additional less often to be used tables are also included in the standard tables DD. While these items have not been integrated into a MACRO, they can still be imported into a DD individually.

To import an individual item, add the item within the common tables import section (example shown below):

```
VARIABLE power_source;
```

The list of additional items (not covered in any of the previous macros) is described in Table 86.

Table 86: Other Tables items

Item Type	Item Name	Description
VARIABLE	*alarm_selection_code*	Local variable defines the list of enumerations used for alarm selections.
VARIABLE	*analog_output_numbers_code*	Local variable defines the list of enumerations used for analog output indexes.
VARIABLE	*burst_message_trigger_mode_codes*	Local variable defines the list of enumerations used for analog burst trigger options.
VARIABLE	*burst_mode_control_code*	Local variable defines the list of enumerations used for analog burst mode settings.
VARIABLE	*company_identification_code*	Local variable defines the list of enumerations used for manufacturer codes.

Item Type	Item Name	Description
VARIABLE	*country_code_codes*	Local variable defines the list of enumerations used for country codes. These are 16-bit integers derived from 2-letter ascii representation of the official 2 character country code designation.
VARIABLE	*device_power_status*	Data item to hold the power status of a device. This item is not referenced from anywhere but saved for future use.
VARIABLE	*device_power_status_codes*	Local variable defines the list of enumerations used for power status bits.
VARIABLE	*device_profile_codes*	Local variable defines the list of enumerations used for device profiles.
VARIABLE	*device_type_code*	Local variable defines the list of enumerations used for device model names. Uses numerous xxx_model_code local variables to build master list.
ARRAY OF VARIABLE	*device_types*	Data array of all device type codes.
VARIABLE	*event_manager_registration_control_codes*	Local variable defines the list of enumerations used for event manager control.
VARIABLE	*event_manager_registration_status_codes*	Local variable defines the list of enumerations used for event manager status bits.
VARIABLE	*event_nofication_control_codes*	Local variable defines the list of enumerations used for event notification modes.
VARIABLE	*join_mode_codes*	Local variable defines the list of enumerations used for join modes.
VARIABLE	*location_method_codes*	Local variable defines the list of enumerations used for location positioning methods.
VARIABLE	*lock_device_codes*	Local variable defines the list of enumerations used for device lock choices.
VARIABLE	*loop_current_mode_codes*	Local variable defines the list of enumerations used for loop current modes.
VARIABLE	*master_mode_code*	Local variable defines the list of enumerations used for IO system master modes.
VARIABLE	*network_access_mode_codes*	Local variable defines the list of enumerations used for network access modes.
VARIABLE	*operating_mode_code*	Local variable defines the list of enumerations used for dev_operating_mode_mask and std_operating_mode_latched_value.
VARIABLE	*physical_layer_type_codes*	Local variable defines the list of enumerations used for physical layer types.
VARIABLE	*physical_signaling_codes*	Local variable defines the list of enumerations used for physical signaling types.
VARIABLE	*power_source_codes*	Local variable defines the list of enumerations used for power sources.
VARIABLE	*si_control_codes*	Local variable defines the list of enumerations used for si unit control choices.

Item Type	Item Name	Description
VARIABLE	*squawk_control_codes*	Local variable defines the list of enumerations used for squawk options.
VARIABLE	*statusmap_codes*	Local variable defines the list of enumerations used for status mapping options.
VARIABLE	*status_simulate_mode_codes*	Local variable defines the list of enumerations used for status simulation modes.
VARIABLE	*status_simulate_value_codes*	Local variable defines the list of enumerations used for status simulation values.
VARIABLE	*std_analog_channel_fixed_codes*	Local variable defines the list of enumerations used for analog channel fixed status bits.
VARIABLE	*std_analog_channel_flags*	Local variable defines the list of enumerations used for analog channel flags.
VARIABLE	*std_analog_channel_saturated_codes*	Local variable defines the list of enumerations used for analog channel saturated status bits.
VARIABLE	*std_event_status_codes*	Local variable defines the list of enumerations used for event notifications status bits.
VARIABLE	*std_extended_device_status_codes*	Local variable defines the list of enumerations used for extended device status bits.
VARIABLE	*std_flag_assignment*	Local variable defines the list of enumerations used for flag assignments.
VARIABLE	*std_join_process_status_codes*	Local variable defines the list of enumerations used for join process status bits.
VARIABLE	*std_lock_device_status_codes*	Local variable defines the list of enumerations used for device lock status bits.
VARIABLE	*std_real_time_clock_flag_codes*	Local variable defines the list of enumerations used for real-time clock states.
VARIABLE	*std_standardized_status_0_codes*	Local variable defines the list of enumerations used for standardized status 0 bits.
VARIABLE	*std_standardized_status_1_codes*	Local variable defines the list of enumerations used for standardized status 1 bits.
VARIABLE	*std_standardized_status_2_codes*	Local variable defines the list of enumerations used for standardized status 2 bits.
VARIABLE	*std_standardized_status_3_codes*	Local variable defines the list of enumerations used for standardized status 3 bits.
VARIABLE	*subdevice_assignment_status_codes*	Local variable defines the list of enumerations used for subdevice assignment status bits.
VARIABLE	*subdevice_assignment_transfer_codes*	Local variable defines the list of enumerations used for subdevice assignment transfers.
VARIABLE	*synchronous_action_control_flag_codes*	Local variable defines the list of enumerations used for synchronous action controls.
VARIABLE	*time_set_codes*	Local variable defines the list of enumerations used for setting the real-time clock.
VARIABLE	*transfer_function_code*	Local variable defines the list of enumerations used for transfer functions.

Item Type	Item Name	Description
VARIABLE	*trend_control_codes*	Local variable defines the list of enumerations used for variable trending options.
VARIABLE	*units_code*	Local variable defines the list of enumerations used for engineering units.
VARIABLE	*units_code_acceleration*	Local variable defines the list of enumerations used for all engineering units codes in the acceleration class.
VARIABLE	*units_code_analytical*	Local variable defines the list of enumerations used for all engineering units codes in the analytical class.
VARIABLE	*units_code_angle*	Local variable defines the list of enumerations used for all engineering units codes in the angle class.
VARIABLE	*units_code_angular_velocity*	Local variable defines the list of enumerations used for all engineering units codes in the angular velocity class.
VARIABLE	*units_code_capacitance*	Local variable defines the list of enumerations used for all engineering units codes in the capacitance class.
VARIABLE	*units_code_concentration*	Local variable defines the list of enumerations used for all engineering units codes in the concentration class.
VARIABLE	*units_code_conductance*	Local variable defines the list of enumerations used for all engineering units codes in the conductance class.
VARIABLE	*units_code_current*	Local variable defines the list of enumerations used for all engineering units codes in the current class.
VARIABLE	*units_code_emf*	Local variable defines the list of enumerations used for all engineering units codes in the emf class.
VARIABLE	*units_code_energy*	Local variable defines the list of enumerations used for all engineering units codes in the energy class.
ARRAY OF VARIABLE	*units_code_expanded*	Data array of all engineering unit codes.
VARIABLE	*units_code_force*	Local variable defines the list of enumerations used for all engineering units codes in the force class.
VARIABLE	*units_code_frequency*	Local variable defines the list of enumerations used for all engineering units codes in the frequency class.
VARIABLE	*units_code_length*	Local variable defines the list of enumerations used for all engineering units codes in the length class.
VARIABLE	*units_code_mass*	Local variable defines the list of enumerations used for all engineering units codes in the mass class.
VARIABLE	*units_code_mass_energy_density*	Local variable defines the list of enumerations used for all engineering units codes in the mass energy density class.
VARIABLE	*units_code_mass_flow*	Local variable defines the list of enumerations used for all engineering units codes in the mass flow class.
VARIABLE	*units_code_mass_per_volume*	Local variable defines the list of enumerations used for all engineering units codes in the mass per volume class.
VARIABLE	*units_code_miscellaneous*	Local variable defines the list of enumerations used for all engineering units codes in the misc class.

Item Type	Item Name	Description
VARIABLE	*units_code_power*	Local variable defines the list of enumerations used for all engineering units codes in the power class.
VARIABLE	*units_code_pressure*	Local variable defines the list of enumerations used for all engineering units codes in the pressure class.
VARIABLE	*units_code_resistance*	Local variable defines the list of enumerations used for all engineering units codes in the resistance class.
VARIABLE	*units_code_temperature*	Local variable defines the list of enumerations used for all engineering units codes in the temperature class.
VARIABLE	*units_code_thermal_expansion*	Local variable defines the list of enumerations used for all engineering units codes in the thermal expansion class.
VARIABLE	*units_code_time*	Local variable defines the list of enumerations used for all engineering units codes in the time class.
VARIABLE	*units_code_torque*	Local variable defines the list of enumerations used for all engineering units codes in the torque class.
VARIABLE	*units_code_turbidity*	Local variable defines the list of enumerations used for all engineering units codes in the turbidity class.
VARIABLE	*units_code_velocity*	Local variable defines the list of enumerations used for all engineering units codes in the velocity class.
VARIABLE	*units_code_viscosity*	Local variable defines the list of enumerations used for all engineering units codes in the viscosity class.
VARIABLE	*units_code_volume*	Local variable defines the list of enumerations used for all engineering units codes in the volume class.
VARIABLE	*units_code_volume_per_mass*	Local variable defines the list of enumerations used for all engineering units codes in the volume per mass class.
VARIABLE	*units_code_volume_per_volume*	Local variable defines the list of enumerations used for all engineering units codes in the volume per volume class.
VARIABLE	*units_code_volumetric_energy_density*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric energy density class.
VARIABLE	*units_code_volumetric_flow*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric flow class.
VARIABLE	*units_code_volumetric_gas_per_day*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric gas per day class.
VARIABLE	*units_code_volumetric_gas_per_hour*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric gas per hour class.
VARIABLE	*units_code_volumetric_gas_per_minute*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric gas per minute class.
VARIABLE	*units_code_volumetric_gas_per_second*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric gas per second class.

Item Type	Item Name	Description
VARIABLE	*units_code_volumetric_liquid_flow_per_day*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric liquid flow per day class.
VARIABLE	*units_code_volumetric_liquid_flow_per_hour*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric liquid flow per hour class.
VARIABLE	*units_code_volumetric_liquid_flow_per_minute*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric liquid flow per minute class.
VARIABLE	*units_code_volumetric_liquid_flow_per_second*	Local variable defines the list of enumerations used for all engineering units codes in the volumetric liquid flow per second class.
VARIABLE	*wireless_capability_flag_codes*	Local variable defines the list of enumerations used for wireless device capability.
VARIABLE	*wireless_operation_mode_codes*	Local variable defines the list of enumerations used for wireless operational states.
VARIABLE	*write_protect_code*	Local variable defines the list of enumerations used for write protection selections.
VARIABLE	*xxx_model_code*	Numerous local variables to define the list of manufacturer device types.

7.13.2 DEPENDENCIES

The Common Tables DD does not have any dependencies.

7.13.3 REDEFINITIONS

7.13.3.1 VARIABLE device_type

When importing VARIABLE device_type from the HART common tables DD, the enumerated type may need to be redefined to include only the corresponding expanded device type and appropriate label to the enumeration.

To redefine the device_type, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE device_type
    {
        REDEFINE TYPE ENUMERATED (2)
        {
            // Replace 0xf9f9 with Expanded Device Type code and appropriate label
            { 0xf9f9,  "New Example Test" }
        }
    }
}
```

7.13.3.2 VARIABLE manufacturer_id

When importing VARIABLE manufacturer_id from the HART common tables DD, the enumerated type may need to be redefined to include only the corresponding manufacturer code and appropriate label to the enumeration. If the manufacturer ID is already included in the standard dictionary, then use the corresponding dictionary reference “[handle]”. Otherwise, use the manufacturer label directly as a string “Company Name”.

To redefine the manufacturer_id, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE device_type
    {
        REDEFINE TYPE ENUMERATED (2)
        {
            // Replace 0x00f9 with your Manufacturer code and appropriate label
            { 0x00f9,  [FieldComm_Group] }
        }
    }
}
```

7.13.3.3 VARIABLE private_label_distributor

When importing VARIABLE private_label_distributor from the HART common tables DD, the enumerated type may need to be redefined to include only the corresponding manufacturer code of the distributor and appropriate label to the enumeration. If the manufacturer ID is already included in the standard dictionary, then use the corresponding dictionary reference “[handle]”. Otherwise, use the manufacturer label directly as a string “Company Name”.

To redefine the private_label_distributor, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE private_label_distributor
    {
        REDEFINE TYPE ENUMERATED (2)
        {
            // Replace 0x00f9 with your Manufacturer code and appropriate label
            { 0x00f9,  "FieldComm Group" }
        }
    }
}
```

7.13.3.4 VARIABLE wireless_module_device_type

When importing VARIABLE wireless_module_device_type from the HART common tables DD, the enumerated type may need to be redefined to include only the corresponding expanded device type and appropriate label to the enumeration.

To redefine the wireless_module_device_type, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE wireless_module_device_type
    {
        REDEFINE TYPE ENUMERATED (2)
        {
            // Replace 0xf9f9 with Expanded Device Type code and appropriate label
            { 0xf9f9,  "New Example Radio" }
        }
    }
}
```

7.13.3.5 VARIABLE wireless_module_manufacturer_id

When importing VARIABLE wireless_module_manufacturer_id from the HART common tables DD, the enumerated type may need to be redefined to include only the corresponding manufacturer code and appropriate label to the

enumeration. If the manufacturer ID is already included in the standard dictionary, then use the corresponding dictionary reference “[handle]”. Otherwise, use the manufacturer label directly as a string “Company Name”.

To redefine the wireless_module_manufacturer_id, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE wireless_module_manufacturer_id
    {
        REDEFINE TYPE ENUMERATED (2)
        {
            // Replace 0x00f9 with radio module Manufacturer code and label
            { 0x00f9,  [FieldComm_Group] }
        }
    }
}
```

7.13.3.6 VARIABLE write_protect

The write_protect VARIABLE is typically defined to be read-only, whose state is often determined by a hardware write protect jumper or switch. However, some devices may allow for software control of the write protect state by including a device-specific HART command to write the value. In cases where the device has software control of the write protect, the handling of the write_protect variable needs to be redefined.

To redefine the write_protect variable, add the following lines to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE write_protect
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.13.3.7 Write Only Burst Mode

For HART 5 devices, the variable burst_mode_select_0 can only be written to the device (there is no read command to read the value until it was added in HART6).

For HART5 devices, to redefine `burst_mode_select_0 to be write-only, add the following lines the redefinition section of the Common Tables DD import:`

```
REDEFINITIONS
{
    VARIABLE burst_mode_select_0
    {
        REDEFINE HANDLING WRITE;
    }
}
```

7.13.3.8 Device Specific Status Mapping

The condensed status mapping variables are intended to allow each of the Additional status (Command 48) status bits to be mapped to a condensed status category. It is expected that each of the status map variables have their labels redefined to match the label as defined in command 48 (see 7.1.3.8). To accomplish this, the status mapping needs to be redefined.

To redefine the status mapping variables, add lines similar to the following for each of the supported device-specific status bits:

```
REDEFINITIONS
{
    VARIABLE statusmap_variable_xxx
    {
        REDEFINE LABEL "My Device Failure Mapping";
        REDEFINE HELP "My Device Failure Mapping – used to map a 'My Device Failure'
condition to a severity level. It allows the status bits returned by the Field Device
in the Command 48 response to be mapped to the Condensed Status bits in the Extended
Field Device Status Byte."
    }
    VARIABLE statusmap_variable_xxx
    {
        REDEFINE LABEL "My Configuration Failure Mapping";
        REDEFINE HELP "My Configuration Failure Mapping – used to map a 'My
Configuratoin Failure' condition to a severity level. It allows the status bits
returned by the Field Device in the Command 48 response to be mapped to the Condensed
Status bits in the Extended Field Device Status Byte."
    }
    … etc.
}
```

7.13.3.9 Writable Transfer function, Alarm Direction, and Write Protect

By default, the HART standard DD defines transfer functions, loop alarm directions, and write protect to be read-only variables. If the deice implements HART commands (e.g. commands 47, 69, 100) to allow the values to be written to the field device, then the variables need to be redefined to have handling READ & WRITE.

To redefine variables to be writeable, add lines similar to the following to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE loop_alarm_code
    {
        REDEFINE HANDLING READ & WRITE;
    }

    VARIABLE transfer_function
    {
        REDEFINE HANDLING READ & WRITE;
    }
}
```

7.13.3.10 Fixed Condensed Status Mapping

By default, the HART standard DD defines all condensed status mapping variables to be writeable variables. If the deice implements some of the mapping to be non-changeable, then the variables need to be redefined to have only handling READ.

To redefine mapping variables to be fixed (read only), add lines similar to the following to the redefinition section of the Common Tables DD import:

```
REDEFINITIONS
{
    VARIABLE statusmap_variable_xxx
    {
        REDEFINE HANDLING READ;
```

```
        }
    }
```

7.13.3.11 SET_DEFAULT(x)

It is best practice to define appropriate default values for the variables in the DD. The default values are used in offline configuration sessions to present a reasonable set of device values to use when a live device is not available. The parameter 'x' is used to specify the setting to be used for the default value.

To define default values for imported variables, add lines similar to the following for all applicable variables:

```
REDEFINITIONS
{
    VARIABLE burst_mode_select_0
    {
        SET_DEFAULT(0);
    }
}
```

7.13.3.12 Device Specific Enumerations

In some cases, the HART Protocol specification allows for custom vendor-defined enumerations to be used. One example of this is the vendor-specific units code enumerations numbered between 240 and 249.

To redefine an enumerated value to include additional enumerations, add lines similar to the following for the affected variables that are imported from HART standard Tables:

```
REDEFINITIONS
{
    VARIABLE transfer_function
    {
        TYPE ENUMERATED
        {
            ADD {234, "Custom"}
        }
    }
}
```

7.13.3.13 Unsupported Enumerations

When importing an enumerated or bit-enumerated variable from the common tables DD, redefining the list of enumerations may be necessary. For variables that are writeable, reducing the list of enumerated choices in a drop-down menu (for example) will benefit users of the DD. This allows the user to understand which choices of values are implemented for a given device. Additionally, the bit-enumerations for standardized status variables that are not supported should also be removed. Note that in cases of large sets of standard enumerations where the device only supports a small subset (e.g. engineering units codes), redefinition of the type with including of only the few supported items may be used rather than deleting a large number of enumerations.

To remove enumerations or bit enumerations from imported variables, add lines similar to the following wherever applicable:

```
REDEFINITIONS
{
    VARIABLE write_protect
    {
        TYPE ENUMERATED
        {
            DELETE 250;
            DELETE 251;
            DELETE 252;
            DELETE 253;
        }
    }

    VARIABLE standardized_status_0
    {
        TYPE BIT_ENUMERATED
        {
            DELETE 0x08;
        }
    }

    VARIABLE burst_trigger_units
    {
        REDEFINE TYPE ENUMERATED
        {
            UNITS_CODE(1),
            /* Add each unit code that is to be supported */
             UNITS_CODE(250)
        }
    }
}
```

7.13.3.14 Burst Option of Enumerations

The HART protocol allows Burst mode operation to be enabled on multiple different communication channels (i.e. token-passing, wireless, HART-IP). For devices that support bursting on multiple interfaces, the labels of the burst mode enumerations need to be updated to make it clear for end users to identify which interface the burst option choices relate to. In this case, the label of the supported options may be redefined, but ONLY using an appropriate label from the HART standard dictionary.

To redefine the burst mode enumerations, add lines similar to below. Additionally, some options may need to be deleted as specified in 7.13.3.13. The example below illustrates all possible options that may need label redefinitions so that the standard dictionary references are captured. Actual DD implementations should only redefine the enumerations that pertain to their device.

```
REDEFINITIONS
{
    VARIABLE burst_mode_control_code
    {
        TYPE ENUMERATED
        {
            REDEFINE {1, on_token_temp, burst_on_token_help_temp },
            REDEFINE {2, on_tdma_temp, burst_on_tdma_help_temp },
            REDEFINE {4, on_hart_ip_temp, on_hart_ip_help_temp }
        }
}
```

7.13.3.15 Help

In order to maintain consistency of standardized items, the LABEL and HELP of HART Standard DD item should not normally be redefined. The exception to this is that vendor specific helpful information may sometimes be useful to add to the help string. For example, adding the location of where to find switches or other accessible items on the field device might be helpful. When redefining the help of an item, the original help text should be used, along with the additional vendor specific help added.

To redefine the help of an item, add lines similar to below. Note that the example only shows the redefined help in English, but the strings for all languages to be supported by your product should also be included in the redefined help string.

```
REDEFINITIONS
{
    VARIABLE write_protect
    {
        REDEFINE HELP "Write Protect- Indicates whether changes to the device's
                       configuration is allowed. This mode can be enabled using hardware
                       jumpers located behind the LCD cover.";
    }
}
```

Annex A: Frequently Asked Questions

A.1 How do I migrate an existing DD to use the new HART Standard DDs?

In general, existing DDs designed to use the previous revision of HART standard DDs will still be largely compatible with this new revision of HART standard DDs. The majority of changes necessary to use this latest release of HART Standard DDs will be take place within the import sections themselves. DD developer specific source can remain largely unmodified.

The fundamental changes with this release that need to be taken into consideration during migration of a DD to the latest revision are described below.

- Import and redefinition macros have been created to allow a simpler way to keep imported items consistent with each other.
- The common practice DD was broken up into smaller individual functional DDs. DDs only need to import the smaller common practice DDs that apply to their device functionality, in some cases they will import more than 1 common practice part.
- The collection used for the deviceVariables array has a new member CONFIG_UNITS that needs to be supplied.
- Burst variable slots index into a new imported burstDeviceVariables[] array instead of the common deviceVariables[] array.

Environment Changes

In order to have access to all the new import macros, the DD must include the “Macros2.h” file rather than the previous “Macros.h”. The older Macros.h” file is still included in the DD development directory to support tokenizing existing DDs. The newer “Macros2.h” file should be used for all updated DDs going forward.

IMPORT Changes

With the new release of HART Standard DDs, most of the items will now be imported by use of common macros. Instead of importing items individually, groups of items can be imported with the use of a single MACRO. When migrating an existing DD to the new revision, it is recommended to look through the list of items being imported, and cross reference them from the tables in section 7 to determine which import macro will pull them in from the various DDs. If an item is included in a new macro, replace the individual import of the item with the new macro definition. Some items will remain individually imported as described in section 7. Note that in some cases some items may have moved to a different standard DD and will need to be imported from a different DD compared to the previous release.

Redefining items that have been imported from the HART standard DDs has caused some registration challenges in the past. Starting with this new release, redefinitions that are expected to be made are documented in specific REDEFINITIONS clauses of section 7. In many cases, macros are included to allow necessary redefinitions to follow HART DD registration policies. When migrating a DD from previous release, look through all the redefinitions used in the existing DD and align them to the redefinitions described in section 7. If the existing DD has a redefinition that is not described in section 7, then either that redefinition is no longer necessary, or must be discussed with FCG to ensure the redefinition is appropriate. If additional necessary redefinitions are found, these will be added to this document in future revisions.

When importing the _PV standard DD, PV, SV, TV, and QV are now referenced by dynamic variable codes by default. Previously they were fixed to map to device variable indexes 0, 1, 2, and 3 respectively. If the existing DD does not support command 50, then the PV1 import will need to include redefinitions to fix to static mapping. See 7.2.3.2 for details.

The HART Standard DD revisions have been updated with this new release. Refer to Table 2 and make sure the DD is updated to import the new standard DD revisions.

Figure 1 shows an example of what the common practice import could look like after migrating to the new HART Standard DDs. Figure 2 shows an example of what the redefinitions of common practice items could look like after migrating to the new HART Standard DDs.

Figure 1 Example of migration of Common Practice Imports

```
IMPORT STANDARD _COMMON_PRACTICE, DEVICE_REVISION 9, DD_REVISION 2
{
    COMMAND read_device_variables;
    COMMAND perform_device_reset;
    COMMAND read_device_variable_assignments;
    COMMAND write_device_variable_units;
    COMMAND read_device_variable_information;
    COMMAND write_number_of_response_preambles;
    COMMAND write_device_variable;
    COMMAND read_real_time_clock;
    COMMAND write_burst_period;
    COMMAND write_burst_trigger;
    COMMAND read_burst_mode_configuration;
    COMMAND flush_delayed_responses;
    COMMAND write_burst_device_variables;
    COMMAND write_burst_command_number;
    COMMAND burst_mode_control;
    COMMAND read_event_notification_summary;
    COMMAND write_event_notification_bit_mask;
    COMMAND write_event_notification_timing;
    COMMAND event_notification_control;
    COMMAND acknowledge_event_notification;
    COMMAND read_country_code;
    COMMAND write_country_code;

    VARIABLE primary_variable_code;
    VARIABLE secondary_variable_code;
    VARIABLE tertiary_variable_code;
    VARIABLE quaternary_variable_code;
    VARIABLE device_variable_code;
    VARIABLE current_date;
    VARIABLE current_time;
    VARIABLE last_clock_time;
    VARIABLE last_clock_date;
    VARIABLE burst_message_number;
    VARIABLE eventNumber;
    VARIABLE update_period;
    VARIABLE max_update_period;
    VARIABLE burst_variable_code;
    VARIABLE time_of_first_unack_event;
    VARIABLE event_notification_retry_time_event;
    VARIABLE max_update_time_event;
    VARIABLE event_debounce_interval_event;
    VARIABLE total_number_burst_messages;
    VARIABLE burst_command_number;
    VARIABLE trigger_level;
    VARIABLE number_events_supported;
    VARIABLE device_specific_status_0_mask;
    VARIABLE device_specific_status_1_mask;
    VARIABLE device_specific_status_2_mask;
    VARIABLE device_specific_status_3_mask;
    VARIABLE device_specific_status_4_mask;
    VARIABLE device_specific_status_5_mask;
    VARIABLE device_specific_status_0_latched_value;
    VARIABLE device_specific_status_1_latched_value;
    VARIABLE device_specific_status_2_latched_value;
    VARIABLE device_specific_status_3_latched_value;
    VARIABLE device_specific_status_4_latched_value;
    VARIABLE device_specific_status_5_latched_value;
    VARIABLE config_change_counter_latched_value;

    COLLECTION event_control;
    COLLECTION event_mask;
    COLLECTION event_report;
    COLLECTION event;

    METHOD return_to_normal;
    METHOD warning_message;
    METHOD device_self_test;
    METHOD device_master_reset;
```

```
IMPORT STANDARD _COMMON_PRACTICE_DEVICE_VARIABLES, DEVICE_REVISION 9, DD_REVISION 3
{
    IMPORT_DEV_VARS_BASE()
    IMPORT_DEV_VARS_MAPPING_READ()

    VARIABLE device_variable_code;

    COMMAND write_device_variable;
    COMMAND write_device_variable_units;

    METHOD return_to_normal;
}

IMPORT STANDARD _COMMON_PRACTICE_ANALOG_CHANNELS, DEVICE_REVISION 9, DD_REVISION 3
{
    COMMAND write_pv_range_values;
}

IMPORT STANDARD _COMMON_PRACTICE_MISC, DEVICE_REVISION 9, DD_REVISION 3
{
    IMPORT_MISC_READ_TIME()
    IMPORT_MISC_SI_UNIT()

    COMMAND perform_device_reset;
    COMMAND write_number_of_response_preambles;
    COMMAND flush_delayed_responses;

    METHOD device_self_test;
    METHOD device_master_reset;

}


IMPORT STANDARD _COMMON_PRACTICE_BURST_REF_ARRAY, DEVICE_REVISION 9, DD_REVISION 3
{
    IMPORT_BURST_REF_ARRAY_BASE(3)
    IMPORT_BURST_REF_ARRAY_EVENT(1)
```

Figure 2 Example of migration of Common Practice Redefinitions

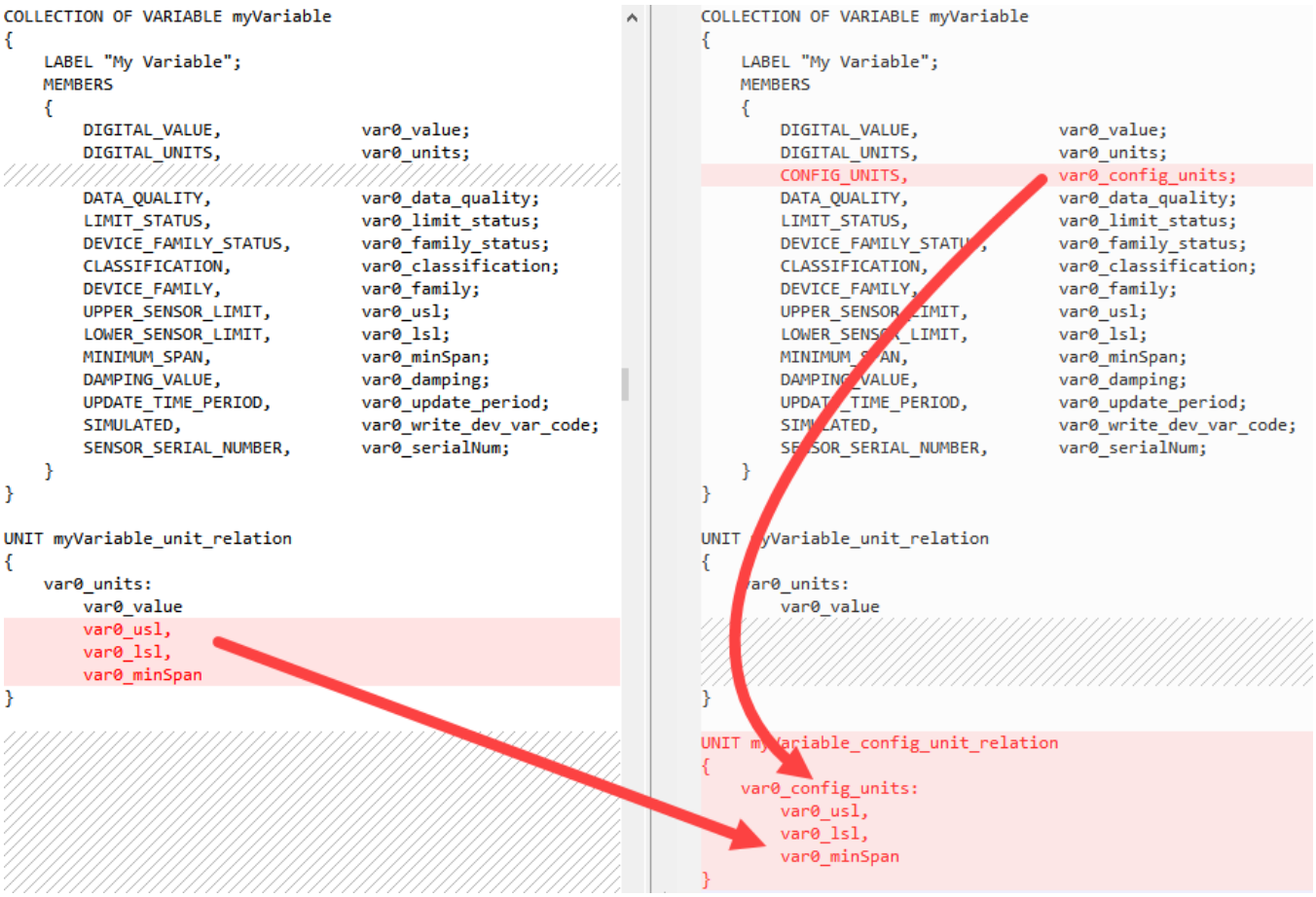
```
REDEFINITIONS
{
    COLLECTION OF VARIABLE event_mask
    {
        MEMBERS
        {
            DELETE ANALOG_CHANNEL_FIXED1_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_14_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_15_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_16_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_17_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_18_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_19_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_20_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_21_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_22_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_23_MASK;
            DELETE DEVICE_SPECIFIC_STATUS_24_MASK;
        }
    }
    COLLECTION OF VARIABLE event_report
    {
        MEMBERS
        {
            DELETE ANALOG_CHANNEL_FIXED1_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_14_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_15_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_16_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_17_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_18_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_19_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_20_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_21_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_22_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_23_LATCHED_VALUE;
            DELETE DEVICE_SPECIFIC_STATUS_24_LATCHED_VALUE;
        }
    }
    COMMAND read_event_notification_summary
    {
        REDEFINE TRANSACTION
        {
            REQUEST
            {
                eventNumber (INFO, INDEX)
            }
            REPLY
            {
                response_code, device_status,
                eventNumber (INFO, INDEX),
                number_events_supported,
                compat_events[eventNumber].EVENT_CONTROL.EVENT_STATUS<0xf0>,
                compat_events[eventNumber].EVENT_CONTROL.EVENT_NOTIFICATION_CONTROL<0x0f>,
                compat_events[eventNumber].EVENT_CONTROL.TIME_OF_FIRST_UNACK_EVENT,
                compat_events[eventNumber].EVENT_CONTROL.EVENT_NOTIFICATION_RETRY_TIME,
                compat_events[eventNumber].EVENT_CONTROL.MAX_UPDATE_TIME,
                compat_events[eventNumber].EVENT_CONTROL.EVENT_DEBOUNCE_INTERVAL,
                compat_events[eventNumber].EVENT_MASK.DEVICE_STATUS_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_0_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_1_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_2_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_3_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_4_MASK,
                compat_events[eventNumber].EVENT_MASK.DEVICE_SPECIFIC_STATUS_5_MASK,
                compat_events[eventNumber].EVENT_MASK.EXTENDED_FLD_DEVICE_STATUS_MASK,
                compat_events[eventNumber].EVENT_MASK.DEV_OPERATING_MODE_MASK//,
                compat_events[eventNumber].EVENT_MASK.STANDARDIZED_STATUS_0_MASK,
                compat_events[eventNumber].EVENT_MASK.STANDARDIZED_STATUS_1_MASK,
                compat_events[eventNumber].EVENT_MASK.ANALOG_CHANNEL_SATURATED1_MASK,
                compat_events[eventNumber].EVENT_MASK.STANDARDIZED_STATUS_2_MASK,
                compat_events[eventNumber].EVENT_MASK.STANDARDIZED_STATUS_3_MASK
            }
        }
    }
```

```
REDEFINITIONS
{
    COLLECTION OF VARIABLE event_mask
    {
        MEMBERS
        {
            REDEFINE_EVENT_MASK_SIZE(13)
        }
    }
    COLLECTION OF VARIABLE event_report
    {
        MEMBERS
        {
            REDEFINE_EVENT_REPORT_SIZE(13)
        }
    }
    COMMAND read_event_notification_summary
    {
        REDEFINE_COMMAND_115_REF_ARRAY_TRANSACTIONS(13)
    }
```

DD Developer additions

There is one additional attribute that DD developers will have to add to existing DDs in order to support the updated revision of HART Standard DDs. The CONFIG_UNITS attribute of each device variable collection needs to be added. Refer to 6.9 for details.

Figure 3 Example of migrating Device Variable Collection to updated HART Standard DD



A.2 What are solutions to common issues encountered during DD development?

This section collects some of the common issues that DD developers may experience during the DD development process. If you experience issues, look to the list below to see if the issue you are having has a common solution. Additionally, it is recommended to check for latest updates and seek support from the FieldComm Group by visiting https://support.fieldcommgroup.org/en/support/home.

Issue 1. When trying to tokenize my DD source, I am getting fatal error: Cannot open include file “macros2.h”.

Verify that the updated “macros2.h” file is included in your DD source directory. This updated include file can be found in the HCF\DDL\Dev directory.

Issue 2. When tokenizing my DD source, I am getting Warning: Wrong number of macro arguments, followed by a syntax error.

Many of the new import macros do not require any input parameters, but some do. Verify that the correct macro arguments are passed into the macro. Refer to the description of the macro in this document to determine what arguments are needed, and what value to use for the arguments. Any macro that requires an argument to be passed when called will have a letter designator between the parenthesis. For example, the IMPORT_TABLES_STATUS(n) macro needs

to be called with a number passed in for n representing the number of additional device status bytes are implemented in the field device.

Issue 3. When tokenizing my DD source, I am getting ERROR 309: syntax error after trying to use the new import macros as described in this document. How do I resolve this issue?

Verify that your DD source is including “macros2.h” instead of the older “macros.h”. Also, verify that there are a pair of parentheses (), and NOT a semicolon after the end of the MACRO line. Follow the syntax as shown within the IMPORTS sections found in section 7.

Issue 4. When tokenizing my DD source, I am getting ERROR 400 or WARNING 400: Could not find item to import.

Verify that the correct import macros are being used within the correct IMPORT DD section, and that the macros are spelled the same as described in this document. The names of the import macros use a common naming convention of IMPORT_xxx_yyy(). The xxx should have a name that is similar to the name used on the IMPORT line. For example, the macro named IMPORT_TABLES_BASE() is to be used within the IMPORT STANDARDS TABLES {} section.

For individually imported items, verify that the item being imported is referenced within the table of section 7 that corresponds to the standard DD it is being imported within.

This error may also be generated if 2 import macros that are intended to be mutually exclusive are both used at the same time. In the description of each import macro in section 7, there are some macros that cannot be used at the same time. These are noted with a highlighted NOTE: statement indicating which macros are mutually exclusive to each other. As an example, the IMPORT_DEV_VARS_MAPPING_READ() and IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro cannot be used at the same time. Verify that only the macros applicable to the device implementation are used.

Issue 5. When tokenizing my DD source, I am getting ERROR 411, ERROR 603, and/or ERROR 664: variable is wrong type of item to be used in a transaction, redefinition error of a variable that is not imported, or a variable is used but is undefined.

Verify that all dependencies of the imported items from the HART standard DD are being satisfied. Section 7 has a subsection for each HART Standard DD title ‘Dependencies’ to illustrate any additional items that are necessary to have in place in order to use each of the imports.

Issue 6. When tokenizing my DD source, I am getting WARNING 418, WARNING 503, or ERROR 608: Item defined was previously defined or imported multiple times.

Verify that the same macro, or same item is not being referenced multiple times. Each item or import macro should only be used once in each DD.

This error may also be generated if 2 import macros that are intended to be mutually exclusive are both used at the same time. In the description of each import macro in section 7, there are some macros that cannot be used at the same time. These are noted with a highlighted NOTE: statement indicating which macros are mutually exclusive to each other. As an example, the IMPORT_DEV_VARS_MAPPING_READ() macro (7.4.1.4) and IMPORT_DEV_VARS_MAPPING_READ_WRITE() macro (7.4.1.5) cannot be used at the same time. Verify that only the macros applicable to the device implementation are used.

Issue 7. When tokenizing my DD source, I am getting ERROR 424: Redefined item is not defined in the imported DD.

Verify that that the correct redefinition macros are being used within the correct IMPORT DD section, and that the macros are spelled the same as described in this document. The redefinition macros are defined in section 7, and are contained within a separate subsection for

each HART Standard DD. The redefinition macros can only be used within the DD import that they are intended for. As an example, the REDEFINE_COMMAND_3_TRANSACTIONS(d) redefinition macro (see 7.1.3.1) can only be used within the HART Universal DD import (see 7.1).

For individually redefined items, verify that the item being redefined is contained within the DD section that it is being imported from.

This error may also be generated if 2 DDs that are intended to be mutually exclusive are being imported at the same time. These are noted with a highlighted NOTE: statement indicating which DDs are mutually exclusive to each other. As an example, the HART Common Practice Burst Value Array DD (7.7) and the HART Common Practice Burst Reference Array DD (7.8) cannot be used at the same time. Verify that only the HART Standard DDs applicable to the device implementation are being imported.

Issue 8. When tokenizing my DD source, I am getting ERROR 425: Response codes have been changed.

This is an erroneous error that is encountered when using the REDEFINE_CONDENSED_STATUS_TRANSACTIONS() macro. You will need to add a command line option "-k 425" to the tokenizer command line to avoid this issue.

Issue 9. When tokenizing my DD source, I am getting WARNING 426: Redefinition of INDEXs is only allowed in a few VARIABLEs.

This warning is intended to make sure that you review any cases where a variable of type INDEX is being redefined in the standard DDs. In general, it is not advised to redefine these variables. Exceptions to this are explicitly described in the redefinitions in section 7 of this document. For example, redefining the device_variable_trim_code to allow trimming only a subset of all the device variables (see 7.4.3.3), or redefining primary_variable_code, secondary_variable_code, tertiary_variable_code, and/or quaternary_variable_code to allow only a subset of device variables to be mapped (see 7.4.3.1). When redefining an index using one of the redefinitions explicitly described in section 7, then it is expected to see this warning, and OK to ignore it. In all other cases, verify that a variable of type INDEX is not being redefined.

Issue 10. When tokenizing my DD source, I am getting ERROR 509 and/or ERROR 612 that items are used but not defined.

Verify that all necessary HART Standard items are being imported properly into your DD. The tokenizer should provide a list of the items that it could not find. Search for those item names within the tables of section 7 to find out which IMPORT macro pulls in those items. Also verify that the parameters passed into the macro calls have the right values, and are consistent with each other as follows:

Macros that have a 'b' attribute should all have the same value for b, that matches the number of burst messages being modeled.

Macros that have a 'd' attribute should all have the same value for d, that matches the number of dynamic variables being modeled.

Macros that have a 'e' attribute should all have the same value for e, that matches the number of event notifications being modeled.

Macros that have a 'n' attribute should all have the same value for n, that matches the number of additional status bytes being modeled.

Macros that have a 'u' attribute should all have the same value for u, that matches the universal HART revision being modelled.

If the item mentioned in the error message is “loop_flags”, and you are working with a HART 5 DD, make sure to include the redefinition as described in 7.2.3.4 so that loop_flags is not referenced incorrectly.

Issue 11. When tokenizing my DD source, I am getting FATAL ERROR 517: Item imported but is not in a symbol file.

Verify that your DD source directory has the new updated “std.sy8” file copied into it. The symbols files have been reorganized into new folders based on breaking apart the common practice DD into multiple smaller DDs. The tokenizer needs the updated std.sy8 file to know where to find all the common practice symbols.txt files. The updated std.sy8 file can be found in the HCF\DDL\Dev directory.

Issue 12. When tokenizing my DD source, I am getting WARNING 602: Same Description appears more than once in a collection.

Verify that the description attributes of each collection member are as intended. Generally, it is not advised to use the same description to describe multiple different items. This may make it difficult for users to determine the difference between multiple items in the same collection. There are some cases where it may be desired to use the same description for multiple members, and that is OK. This warning is meant to remind developers when this situation is encountered so that they can make sure it is as intended. One such case is with the new CONFIG_UNITS member added to the device variable collections. In this case, the CONFIG_UNITS and DIGITAL_UNITS are commonly labeled with the same description because to the end user they are the same thing.

Issue 13. When tokenizing my DD source, I am getting ERROR 608: item imported multiple times.

This will often happen when an item is being implicitly imported, that is already being imported in one of the import MACROs. Check each explicit item being imported to make sure it is not already being imported somewhere else (including from within a MACRO). Sections 7.x.1 in this document list tables of all items that will be imported by using the import MACROs. Also note, that a change between the file macros2.h between the version released in FDI IDE v 1.5 and 1.5.1 underwent a change that now automatically imports VARIABLE device_variable_code within the IMPORT_DEV_VARS_BASE() macro. Any DD source code that was developed using FDI IDE v 1.5 may need to remove a line of source code that was explicitly importing device_variable_code.

Issue 14. When tokenizing my DD source, I am getting WARNING 669: Variable Has CONSTANT_UNITS deleted on import which may affect interoperability.

This warning message may appear when making the redefinition discussed in clause 7.7.3.6. As long as the unit relation is added to replace the CONSTANT_UNIT relation, the value should still be associated with the correct units as reported by the field device. Verify that the proper units codes are being displayed with the corresponding value. If correct units code is displayed for all cases, this warning can be ignored.

Issue 15. When tokenizing my DD source, I am getting an ERROR 677: COMAND has item reference in the REQUEST packet that is missing in the REPLY packet.

When importing the IO System DD into your DD, you will need to add a command line option “-k 677” to the tokenizer command line. This is due to the way that subdevice mapping breaks apart the most significant bit to differentiate between event notification mapping and burst mapping.

Issue 16. When tokenizing my DD source, I am getting an ERROR 681: Item contains an invalid reference using element ‘x’.

This can occur when importing a command from the HART standard DD that is dependent on certain collection members that have not been provided in the device-specific part of the DD implementation. Refer to Table 3 and make sure that all device variable collection members are provided for the device variables in your DD.

Issue 17. My DD tokenized successfully, but when I try to use it with my live device, I see some communication errors or warnings. These errors or warnings can include messages about some commands did not return enough data bytes, or error response code 5 “Too Few Bytes Received”, or that a command could not be found to read or write a variable.

These types of issues that occur when communicating with a live device are typically caused by the DD model not matching the device implementation. Verify that the set of COMMANDs being imported are implemented in the field device, and that the commands implemented in the device are all being imported into your DD. All items defined in the HART Standard DDs are listed in the tables within section 7. Look specifically at the items that are listed as item type COMMAND. The commands used in the DD must match the commands implemented in the field device.

Some commands have varying lengths and must be redefined appropriately to match the implementation of the field device. The Redefinition subsections of section 7 describe the cases when these redefinitions are applicable. In many cases a redefinition MACRO is available, and a parameter must be passed into the macro to indicate what the field device supports. Verify that the values passed into parameterized macros are set to the right values according to the field device implementation.

Issue 18. My DD tokenized successfully, but when I try to use it with my live device, I see some parameter errors or warnings. This may be an indication such as an unknown or invalid enumeration when a value is read from the device. Or it could be a response code message such as invalid selection when attempting to write a value to the device.

These types of issues that occur when communicating with a live device are typically caused by the DD model not matching the device implementation. Verify that the set of enumerated variables being imported (usually from the tables DD) have the same set of enumeration that the field device implements, and that the enumerations implemented in the device are all included in the imported variables. All items defined in the HART Standard DDs are listed in the tables within section 7. Look specifically at the items that are listed as item type VARIABLE. The allowed variable values used in the DD must match the allowed variable values implemented in the field device.

Some enumerated variables need to be redefined to have optional values deleted, or device specific values added to enumerations to match the implementation of the field device. The Redefinition subsections of section 7 describe the cases when these redefinitions are applicable. Redefinitions of enumerated values are described in 7.13.3.11 and 7.13.3.12. Verify that the appropriate set of redefinitions are being used to match the implementation of the field device.

Issue 19. My DD tokenized successfully, but when I try to use it in a DD application, some of the strings for certain items are blank / empty.

This issue can occur specifically when using the version 8 tokenized binary file (.fm8). This occurs in certain older versions of DD applications that have not been updated to use the latest version of the HART standard dictionary (standard.dc8). To address this issue, the HART standard library has been tokenized in a way that the older version 8 file format references static strings rather than referencing to the external dictionary file. Make sure that your DD is importing the latest versions of the HART standard DDs (refer to Table 2). Additionally, if you are making any references to standard dictionary handles outside of the HART standard imported items (any

strings that use the square bracket dictionary references such as *[xyz]*), replace these with a direct string instead. See 6.10 for more details.