



**Inter-Processor Communication**

**Protocol**

Table of Contents

[1 Introduction 3](#_Toc325042945)

[2 General Architecture 4](#_Toc325042946)

[2.1 Physical Layer 4](#_Toc325042947)

[2.2 Link Layer 5](#_Toc325042948)

[2.3 Command format and definition. 6](#_Toc325042949)

[2.3.1 Dynamic Parameter Exchange Command (Command 177) 7](#_Toc325042950)

[2.3.2 Read Array Command (Command 178) 8](#_Toc325042951)

[2.3.3 Write Short Variables (Command 186) 9](#_Toc325042952)

[2.3.4 Write Array (Command 187) 10](#_Toc325042953)

[2.4 Variable and Action IDs 11](#_Toc325042954)

[2.5 Variable Status and error reporting 11](#_Toc325042955)

[2.6 Variable representation 13](#_Toc325042956)

[3 Loss of communication detection 15](#_Toc325042957)

[3.1 Detection of loss of communications on the APP board 15](#_Toc325042958)

[3.2 Detection of loss of communications on the FF board 15](#_Toc325042959)

[4 To be discussed and defined 16](#_Toc325042960)

[5 Revision History 17](#_Toc325042961)

# Introduction

The purpose of this document is to present the design and definition of the Inter-Processor Communication (IPC) protocol for FF Project.

The Role of IPC is to provide the information exchange between Field Bus Front End processor (FFP) and Application specific processor (APP).

# General Architecture

The IPC is a Master Slave protocol, with FFP being a Master and APP being a Slave.



Fig.1 Typical IPC message structure.

As shown on Fig 1 the Master (FFP) forms a command and sends the command via the Link and Physical layer to the Slave (APP). The Slave (APP) receives the command, parses the command parameters, executes the command and then sends the response back to the master within predefined time.

## Physical Layer

The Physical layer is implemented by directly connecting two UARTS of the Master and Slave. For simplicity, both APP and FPP microcontrollers utilize the same UART peripheral (on the chip) devices. Given the performance limitations the Physical Layer will utilize DMA.

Physical Layer configuration:

* 38.4kBaud
* 8-bit character
* No parity
* 1 stop-bit.

## Link Layer

Link Layer will be utilizing the following Standard HART Command format with modification: using short address.

This format is based on HART Command format and allows utilizing the existing HART command dispatcher.

The commands will be of a fixed size and fixed format for each command, to allow deterministic receiving and processing time. The commands’ format is described later in this document.

The responses are of a pre-defined format specific for each command to allow deterministic transmission and processing time. The response’s format will be defined later in this document.



Fig. 2 Structure of layers

## Command format and definition.

IPC Interfaces between FFP and APP will be implemented as a set of commands for moving variables and memory data between these two microcontrollers and for instructing APP to perform a certain activity.

The IPC protocol is a Master - Slave protocol with very strict role definitions, the FPP is a Master, and APP is a Slave.

A command for moving data from FPP to APP will be named “Writing”; moving data from APP to FPP will be named “Reading”. The instruction to perform some activity will be called “Action”. Every variable in read/write operation is identified by a unique ID (“Variable ID”); every action is identified by unique “Action ID”.

Variable ID = 0x0 indicates that the variable does not really exist and the data field shall be filled with 0-s for compatibility reasons. The requests for read and write a variable with ID=0 should be ignored.

Every command may be initiated only by FFP, which generates a request. Immediately after a command is received by APP, APP must respond in accordance with the type of the command and its parameters. The APP after completion of the command shall sent the response in an appropriate predefined format.

***Note: all commands are of the fixed size – 14 bytes.***

***The responses’ format and size are defined appropriately for each specific command and the data being transferred. The Master and the Slave shall build and parse the responses appropriately.***

***The variable format and representation shall be defined individually for ach variable. The FFP and APP shall use appropriate format for sending each variable. The Variable ID shall define the format of the variable.***

***Variable ID’s shall be unique: one ID for one variable of Action. Refer to the document describing the variables and their IDs.***

The number representation: HART commands shall utilize the standard number representation as defined by the HART specification:

HART Communication Foundation Document Number: HCF\_SPEC-99

HART®-SMART Communication Protocol: Universal Command Specification

### Dynamic Parameter Exchange Command (Command 177)

Command is intended to be sent from Master (FFP) to Slave (APP) to send the new value of the Set Point (SP) obtain the values of several variables, defined in the command by their indexes.

The Master sends **Dynamic Parameter Exchange Command** withas a Command **(),** SP and 3 variable indexes to the Slave. The Slave obtains the requested values, converts them to FFP required scale, Engineering Units and representation, and then sends them to the Master in the defined format.

**Request: FFP --> APP (14 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Target**  **Mode** | **SP** | | | | | **I/P Operation** | | **Variable 1**  **ID** | | **Variable 2**  **ID** | | **Variable 3**  **ID** | |
| **Status** | **Value** | | | | **Status** | **I/P Action** |
|  |  |  | | | |  |  |  | |  | |  | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |

**Response: APP --> FFP (21 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Actual**  **Mode** | **Position** | | | | | **Variable 1** | | | | | **Variable 2** | | | | | **Variable 3** | | | | |
|  | **Status** | **Value** | | | | **Status** | **Value** | | | | **Status** | **Value** | | | | **Status** | **Value** | | | |
|  |  | | | |  |  | | | |  |  | | | |  |  | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |

The Target mode will contain the requested mode for the APP board. The mode may be:

0x00 – No change

0x04 – Normal Mode

0x80 – Out of Service

The Actual mode will contain the requested mode for the APP board. The mode may be:

0x02 – Manual Mode

0x04 – Normal Mode

0x80 – Out of Service

Status is defined in the section 2.5.

SP will contain the Set Point position for the APP board. SP is presented as a Floating Point number.

Position is also presented as a Floating Point number.

I/P Action is defined as follows:

0x00 – No action

0x01 – Full Open

0x02 – Tight Shutoff

Application note: The status of the SP shall be BAD if the status of the I/P Operation is GOOD. Vice versa, the status of I/P Operation shall be BAD if the status of SP is GOOD. If both statuses defined as GOOD then the command should be treated as a bad message that can trigger a Fault State transition. Defining both statuses as BAD is acceptable behavior.

Variable X is the ID for the variable that should be read back. Note, that the ID should identify variables of the following type:

Boolean ( it will be sent as Unsigned Int8)

Int8

Int16

Int32

UInt8

UInt16

UInt32

Float

Bit String (1, 2 and 4 bytes). The Bit strings shall be sent as follows:

Bit string of 8 bits (1 byte) shall be sent as Unsigned Int8

Bit string of 16 bits (2 byte) shall be sent as Unsigned Int16

Bit string of 32 bits (4 byte) shall be sent as Unsigned Int32

The following Data types should NOT be read with this command:

Visible String

Octet String

Date

Time of Day

Time Difference

Time Value

### Read Array Command (Command 178)

The other variation of Read command is intended for reading byte arrays. The Master sends Command **Read Array** withArray VariableID. The Slave obtains the requested values from the array variable (up to 32 bytes), and sends them to the FFP. **Note:** the FFP and the APP must know the proper array size for the specified Variable ID.

**Request: FFP --> APP (14 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array**  **Variable**  **ID** | | **Data Block**  **Number** | **Reserved** | | | | | | | | | | |
|  | |  | **Fill with 0.** | | | | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | |

**Response: APP --> FFP (33 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable**  **Status** | **Array data (up to 32 bytes)** | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **…** | **29** | **30** | **31** | **32** |

Data Block Number – Identifies the block number being read. It is applicable only for reading variables longer than 32 bytes.

Data block number should be equal to 0 for all variables, 32 byte or less.

Data block number should indicate the block number used to calculate the offset from the beginning of the block according to the following formula:

Offset = Data Block Number \* 32

For Variables, that are more than 32 bytes, data consistency has to be maintained during the whole variable data transfer. The client should ask for the blocks starting with number 0 and following the natural sequence of data block numbers – 1, 2, etc.

Reading the block numbers in wrong sequence will trigger an error response.

The server shall maintain the consistency of data until the last data block number is read. Triggering another command before the last data block number is read shall break the consistency of the data and will require to read the data block numbers from beginning – 0, 1, 2, etc.

The following Data types are expected to be read with this command:

Visible String

Octet String

Date

Time of Day

Time Difference

Time Value

When the reply is smaller than 32 bytes, the data will be filled from the beginning of the array.

### Write Short Variables (Command 186)

Command is intended to be sent from Master (FFP) to Slave (APP) to write the values of several variables, defined in the command by their indexes. Master sends Command **Write variables** with SP and 3 variable indexes and values to the Slave. The Slave receives the command, re-maps the variables and stores the values. Slave also forms the response and sends it to the Master.

As it was noted above, Variable ID = 0x0 indicates that the variable does not really exist and its value shall be ignored. The unused field shall be filled with 0-s for compatibility reasons.

**Request: FFP --> APP (14 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable 1**  **ID** | | **Variable 1** | | | | | **Variable 1**  **ID** | | **Variable 2** | | | | |
|  | | **Status** | **Value** | | | |  | | **Status** | **Value** | | | |
|  | |  |  | | | |  | |  |  | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |

**Response: APP --> FFP (2 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable 1**  **ID** | | **Variable 1** | | | | | **Variable 2**  **ID** | | **Variable 2** | | | | |
|  |  | **Status** | **Value** | | | |  |  | **Status** | **Value** | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |

Variable X is the ID for the variable that should be written. Note, that the ID should identify variables of the following type:

Boolean (it will be sent as Unsigned Int8)

Int8

Int16

Int32

UInt8

UInt16

UInt32

Float

Bit String (1, 2 and 4 bytes). The Bit strings shall be sent as follows:

Bit string of 8 bits (1 byte) shall be sent as Unsigned Int8

Bit string of 16 bits (2 byte) shall be sent as Unsigned Int16

Bit string of 32 bits (4 byte) shall be sent as Unsigned Int32

### Write Array (Command 187)

The other variation of Write command is intended for writing byte arrays. The Master sends Command **Write** **Array** withvariable ID, Data Block Number within the array to write from and up to 11 bytes of data. The Slave stores the received data and forms the response and sends it to the Master. **Note:** the Master does not send the offset with the array; instead it sends the Data Block Number – the number of the 11-byte Data Block within the array. The **Write Array** command always shall start from the Data Block 0 and continue incrementing it for further data.

For example: writing 32-byte array will require 3 commands:

* with Data Block Number 0 and 11 bytes of data (array bytes 0..10);
* with Data Block Number 1 and 11 bytes of data (array bytes 11..21);
* with Data Block Number 2 and 10 bytes of data (array bytes 22..31).

The unused bytes at the end of the buffer shall be filled with 0s. The server must ignore the unused bytes and assure that no buffer overflow will be created when the data are copied. The server also will maintain the data consistency.

**Request: FFP --> APP (14 bytes)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable**  **ID** | | **Data Block**  **Number** | **Data Block (up to 11 bytes)**  **Fill unused bytes with 0.** | | | | | | | | | | |
|  | |  |  | | | | | | | | | | |
|  | |  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |

**Response: APP --> FFP (1 byte)**

|  |
| --- |
| **Variable**  **Status** |
| **Status** |
|  |
| **0** |

Data Block Number – Identifies the block number being read. It is applicable only for reading variables longer than 11 bytes.

Data block number should be equal to 0 for all variables, 11 byte or less.

Data block number should indicate the block number used to calculate the offset from the beginning of the block according to the following formula:

Offset = Data Block Number \* 11

For Variables, that are more than 11 bytes, data consistency has to be maintained during the whole variable data transfer. The client should ask for the blocks starting with number 0 and following the natural sequence of data block numbers – 1, 2, etc.

Reading the block numbers in wrong sequence will trigger an error response.

The server shall maintain the consistency of data until the last data block number is read. Triggering another command before the last data block number is read shall break the consistency of the data and will require to read the data block numbers from beginning – 0, 1, 2, etc.

## Variable and Action IDs

The following table summarizes the variables used by the IPC protocol with Write and Read commands.

It also lists Action IDs, for actions that may be requested from APP module by FFP.

***“Action” commands will be described later, they are not critical to the IPC implementation.***

All variables may be read in any mode.

Some variables may be written only in certain APP modes. These modes are also indicated in the table along with variables.

No variable associated with **ID = 0**. When it is specified in the Write or Read command, it means no variable to be written or read.

## Variable Status and error reporting

Below is the format of the Status Byte:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit number** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Function** | **Variable Quality**  0 - Bad  2 - Good  3 - Good Cascaded  1 - Uncertain | | **Error code** | | | | **Limits** | |
| See table of Error codes below.  Error codes will be applicable only if the variable quality is marked Bad. Error code shall be OK (0x00) if the variable quality is Good | | | | **Upper** | **Lower** |

Data, written with Bad or Uncertain status should be ignored.

Error codes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error code** | **Read command** | | **Write command** | |
| **Function** | **Description** | **Function** | **Description** |
| **0** | **OK** | Read Command executed successfully | **OK** | Write Command executed successfully |
| **1** | **Variable ID Not Applicable** | Variable ID = 0, indicates that the Variable is not actually used and the Read field shall be filled with 0’s or set as NaN for Floating Point variable. | **Variable ID Not Applicable** | Variable ID = 0, indicates that the Variable is not actually used and the part of the command shall be ignored. The Variable field shall be filled with 0’s or set as NaN for Floating Point variable. |
| **2** | **Bad Index** | Variable ID specified for the particular variable is undefined. | **Bad Index** | Variable ID specified for the particular variable is undefined. |
| **3** | **Bad Data Block Number** | The Data Block Number (32-byte data block) specified in the command is not valid. For example, when reading 32-byte array the Host specifies Data Block Number 5. | **Bad Data Block Number** | The Data Block Number (11-byte data block) specified in the command is not valid. For example, when writing 32-byte array the Host specifies Data Block Number 5. |
| **4** | **Value is not available** | The Variable ID specified in the command is valid but the variable is not actually available. For example, the pressures may or may not be available depending on the configuration, but the Variable IDs are defined. | **Variable is not available** | The Variable ID specified in the command is valid but the variable is not actually available and cannot be written. |
| **5** | **Type conflict** | Type conflict: Read Array issued for the Variable ID that is not an array, or Read Variable command is issued for the Variable ID that specifies an array variable. | **Type conflict** | Type conflict: Write Array issued for the Variable ID that is not an array, or Write Variable command is issued for the Variable ID that specifies an array variable. |
| **6** | **Inconsistent data** | Data cannot be read due to device condition that prevents consistent data access (memory error, etc.) | **Inconsistent data** | Data cannot be written due to device condition that prevents consistent data access (memory error, etc.) |
| **7** | ***N/A*** | *Error code undefined* | **Data not writeable** | The data cannot be written as they are write protected. |
| **8** | ***N/A*** | *Error code undefined* | **Out of Range** | The data cannot be written – data out of specified range. |
| **9** | ***N/A*** | *Error code undefined* | **Out of Enumeration** | The data cannot be written –data out of specified enumeration. |
| **10** | ***N/A*** | *Error code undefined* | **Validation failed** | The data cannot be written – data validation failed. |
| **11** | ***N/A*** | *Error code undefined* | **State Conflict** | The data cannot be written due to the device state conflict (incorrect mode, etc.) |
| **12** | **Reserved** | Reserved | **Reserved** | Reserved |
| **13** | **Reserved** | Reserved | **Reserved** | Reserved |
| **14** | **Reserved** | Reserved | **Reserved** | Reserved |
| **15** | **Extended Error Code** | This error code indicates that the Function / Variable / Device Specific Error had occurred, the extended Error Code is placed into the Variable field. | **Extended Error Code** | This error code indicates that the Function / Variable / Device Specific Error had occurred, the extended Error Code is placed into the Variable field. |

## Variable representation

Variable representation is defined by a separate document, providing mapping of the TB and RB variables and parameters.

For each of the variables defined in the APP board, the following information shall be provided:

* Variable ID
* Variable type – e.g. integer, unsigned integer, float, etc.
* Variable length in bytes – e.g. an integer may be 1, 2, or 4 bytes. Or a string may be 32 bytes.
* State, required to change the variable – e.g. the variable can be changed in Normal state or the variable can be changed only when the APP board is in OOS state.

All data will be transferred to and from the APP board in the native APP board data type and format. Transformations, if necessary shall be done in the FF board.

Only derivatives of psi units must be used in APP and for passing between FFP and APP. Should other pressure units have been required the translation must be performed in FPP.

The following table summarizes the units for integer representation of intrinsically decimal variables.

# Loss of communication detection

The procedures for detection of loss of communications and for the APP and FFP microcontroller behavior shall be described in the separate document. Below are exerts describing the proposed detection of loss of communications.

## Detection of loss of communications on the APP board

* Good message is not received in 10\*15ms (150 mS)
* Brocken message received 3 times in a row (incomplete message received, character frame error (parity), checksum incorrect)
* Two Requests received before the previous one is completed

## Detection of loss of communications on the FF board

* 3 broken responses received in a row (incomplete message received, character frame error (parity), checksum incorrect)
* Two responses did not come in a row
* More than one reply comes after the request

# To be discussed and defined

Document List

| **Reference** | **Document** | **Revision** | **Description** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 

# Revision History

The table below describes the revision history of this document.

|  |  |  |
| --- | --- | --- |
| Rev. | Changed figure, table, chapter | Title or brief description |
| A | 11/02/2011 | Initial revision. |
| B | 04/24/2012 | Update |
| C | 05/10/2012 | Update – extended error code; Write Short command format change. |
| D | 05/17/2012 | Update: Changed IPC commands numbers. |
|  |  |  |
|  |  |  |
|  |  |  |