Title: **Requirement Specification Common Diagnosis Sub-System**

References:

[1]. Software Design Description – Common Field Device Framework
[2]. NE107 - Self-monitoring and diagnosis of field devices - Version: 01.03.2005
[3]. IEC 61508 Part 3 and Part 7 First edition 1998-12
[4]. Developer's Handbook - A Guideline for Developing Field-Devices in Business Unit Instrumentation
[5]. Common Diagnosis Subsystem Mapping Guideline, Rev. 0.4
[6]. ARM - Access Rights Management Requirement Specification (this document is in draft version, no subsystem has been developed yet)

Distribution Development, Product Management, Quality Management, Industrial Engineering, Service, Project Leader

Author Giovanni Invernizzi Date: 2007 March 19

Review Product Management

Quality Management

Approved R&D Management

Product Management

Industrial Engineering

Service

Supply Management

Project Leader

Remarks

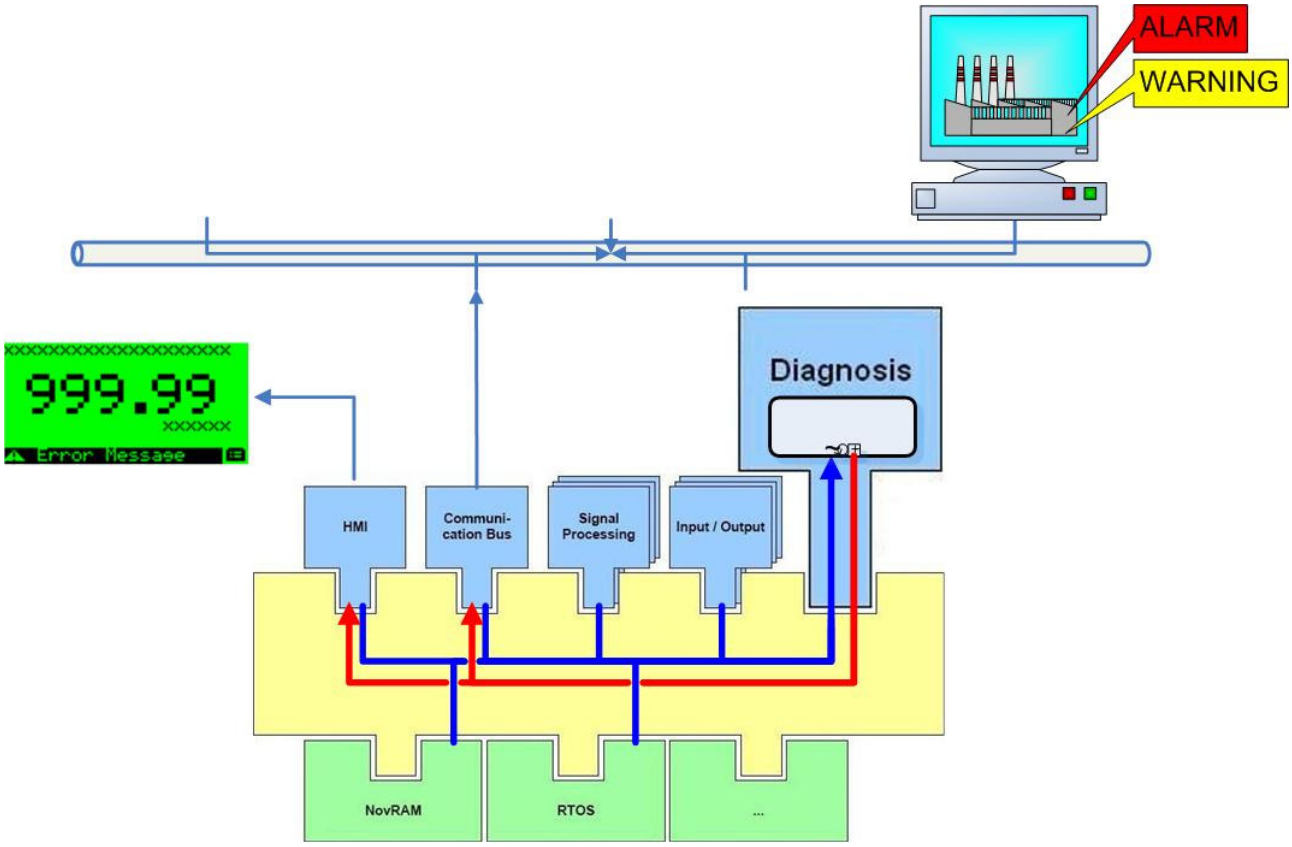
Contents

1 Introduction

1.1 Scope

Today BUI have a large variety of different diagnostic system developed for their products. The worst problem of this approch is that alarm presentation to the customer is different for each device type even if the alarm types are the same.

The introduction of the Process 5k philosophy in the software design brought to a component-oriented development method that would assure time reduction in device development through the reusability of field device software. Another important goal reached by the Process 5k methodology is the harmonization of the customer oriented part of each device, from housing to data presentation (via HMI or digital communication protocol). Even if the presentation of an alarm to the outer world is not within the scope of this project, we recognize that a Common Diagnosis SubSystem will help in the harmonization process though a common integrated way of managing alarms.



Many users consider the possibility to get diagnosis information from field instruments through communication protocols one of the most important aspects and advantages of FieldBus technology. Just for this reason, some organizations, representing the device end users (NAMUR, BSI, etc.), produced recommendations (NE107) in which they defined a common terminology and a common structured way for the treatment and interpretation of the diagnostic conditions. These recommendations in practice require that each diagnostic condition produced by the field instrument has associated additional attributes/information necessary for better understanding which actions are necessary to be taken and/or which part of the instrument has been affected by the malfunction and so on.

Furthermore it has been recognized as an added value by the user, the possibility to interact with each diagnosis condition masking or simulating them depending by the specific operations or conditions in which they are working.

The purpose of this requirement specification is to define the minimal set of requirements every BUI device diagnosis has to fulfill. Therefore requirements from the different R&D-Teams were collected and presented in this specification.

1.2 Definitions, acronyms, and abbreviations

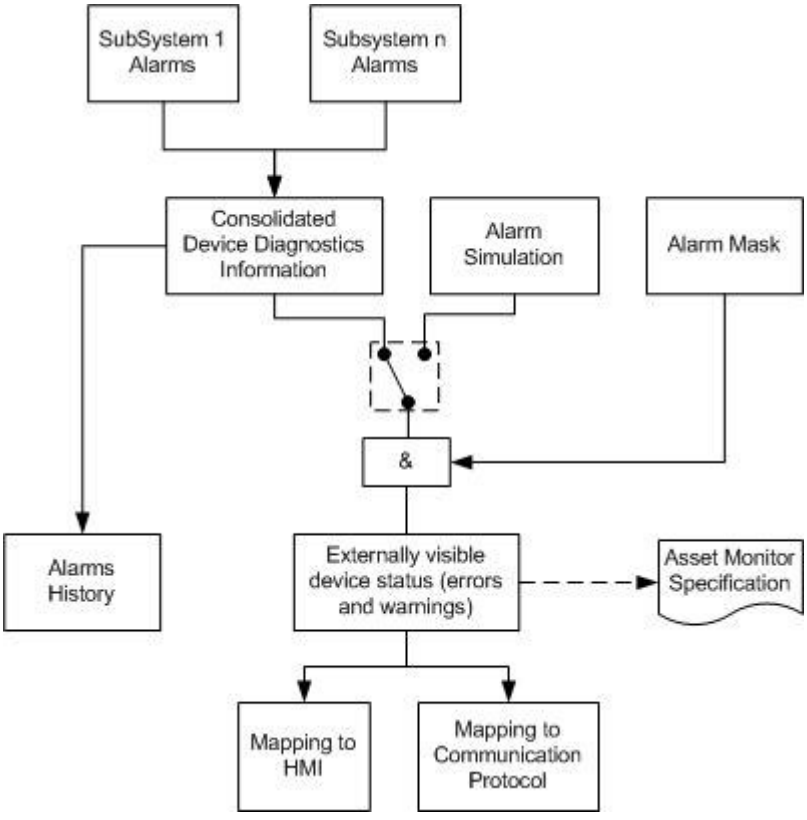
FF	FOUNDATION FieldBus
PA	PROFIBUS PA
UxTHE	Universal Extended Hart Transmitter Electronics
HART	HART Communication Protocol
HMI	Human Machine Interface (Local Indicator + Keys)
HW	Hardware
Alarm	Any abnormal condition occurred within a device
Error	Any abnormal condition that stops the device primary variable production
Warning	Any abnormal condition that could reduce the device accuracy
ARM	Access Rights Management common component

2 Overall description

With the help of self-monitoring, the status information from field instruments can be made available. It serves primarily to check the measurement values from the sensor or the functionality of the actuator for plausibility and to avoid negative influences on the product quality or even system shut downs due to failure of the field instrumentation.
Device specific advanced diagnostic features usually can't be used from the control system because it has no knowledge on how to use and interpret such diagnostic information. Therefore ABB has defined its own proprietary method, which is based on so called Asset Monitors. An asset monitor can read diagnostics information of a device and has all the knowledge on corrective measures, alarm texts to display etc. Device specific asset monitors can also contain specific know-how about non-standard diagnostics information so ABB devices can be more valuable in ABB systems than in other non-ABB systems.

Devices implementing a similar diagnostic architecture will be Host independent because all the most relevant information is directly produced by the field device.

The following data flow diagram shows data flow of diagnostics information from their origins, which are the subsystems of a device to the consumers of this information. From the subsystems the diagnostics information is collected and consolidated for generation of a device status. This status is made available for further use. Users can both simulate the device status and mask-out specific status bits. The so processed status is then mapped to different users such as HMI, communication interfaces etc.



Field devices producing common diagnostic conditions can be integrated in the overall systems/tools in an easier and better-harmonized way only if these diagnostic conditions have been mapped in the same way within the field devices. Additional device specific diagnostic conditions can be individually identified for each device but their handling and mapping must be specified and implemented following a common rule.

Whenever all the devices will satisfy the above requirements, it will be possible to treat part of their diagnosis capability with:

- Same reading mechanism/logic
- Same specification for the Asset Monitor (Optimize IT)
- Same representation in the DTM
- Same description in the DD
- Same naming and meaning

The above requirement enforces treatment of each diagnosis condition not as a simple bit but as a combination of information covering several aspects that were traditionally implemented/assembled within the maintenance stations (ASSET MONITORs). The advantage is that it should be no longer necessary to produce specific device integrations for different Hosts because all the relevant information is directly produced by the field devices.

The goal of a Common Diagnosis Sub System, under the process 5k methodology, is to define a common way of dealing with alarms (either warnings or errors) and develop a subsystem that is able to manage any kind of alarms from a common set of diagnosis alarms to device specific alarms. The result of this project will be:

1. A software component based on the common framework methodology that has to be used in all future field device developed by ABB BUI. Thus it is ensured that any kind of subsystem can be easily reused greatly modifying the alarm handling of the subsystem
2. A guideline explaining how to manage the alarms. A common presentation (Fieldbus telegram mapping, HMI string, etc.) of the same error condition valid for all the BUI devices ensures a common look & feel which should be recognized by the users as a real benefit.
3. A code generator to help the final user to easily configure the Common Diagnosis subsystem.

3 Assumptions and dependencies

The NAMUR Recommendation NE107 shall be considered as a part of this document.

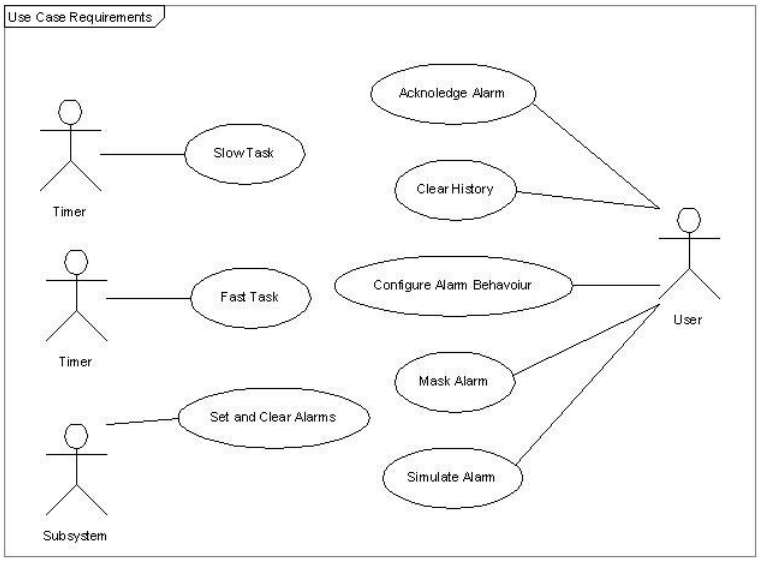
Whereas, in this document is mentioned the: user interface, the presentation of parameters, alarms and interface mechanism to the Host, these shall be discussed in separate projects like:

- Common Function Block for FF and PA.
- UxTHE for HART protocol and commands definition.
- In separate working groups of these projects involving also FieldBus protocols experts and system people.

4 Functional requirements

This chapter gives an overview about the functional requirements the transmitter has to fulfill.

4.1 Use Cases



No	Use Case[1]	Actors[2]	Description
1	Set and Clear Alarms	Subsystems	Storing and removing alarms subsystem alarm into diagnosis subsystem
2	Fast Task	Timer	Diagnosis Subsystem computes actions to be taken during fast task
3	Slow Task	Timer	Diagnosis Subsystem updates: 1. The status of the device based on set alarms. 2. Internal information about diagnosis condition.
4	Configure Alarm Behaviour	User	Set up the actions to be taken during fast task
5	Simulate Alarms	User	Simulate an alarm condition to test device functionality
6	Mask Alarms	User	Mask an alarm condition in order to remove it from Slow Task update
7	Clear History	User	Clear the alarm history for a selected alarm condition
8	Acknoledge Alarm	User	Acknoledge an alarm that needs to be ackonledged to be removed.

[1] Use Cases specify what happens when actors interact with the system. They specify the intent, not the action detail.
[2] Actors in this context are external entities (people or other systems) who interact with the system to achieve a desired goal.

4.2 General Requirements[3]

4.2.1 Asset Monitors Relationship

Definition / Motivation	**Prio** 1	**Source**
The common diagnosis component shall be able to make decisions about the device status. The above condition is the basic prerequisite in order to have devices not requiring specific integration for specific Asset Monitors, so the device's diagnosis must be HOST Independent. This means that the device itself must produce diagnosis conditions organized in a structured way including additional info giving complete details of which actions the user has to take.	**Stability** C	Development

4.2.2 Common Framework

Definition / Motivation	**Prio** 1	**Source**
It is a common software component to be developed in line with the Framework concept with the task to manage the device's diagnosis conditions and their additional info before being presented to the external world via Communication Protocol, Local HMI, and Reaction of the device (4-20 mA or Safe state).	**Stability** C	Development

4.2.3 Process 5k

Definition / Motivation	**Prio** 1	**Source**
The common diagnosis component shall be developed according to Process 5k directives.	**Stability** C	Development

4.2.4 Architecture and Coding Rule

Definition / Motivation	**Prio** 1	**Source**
The Common Diagnostics Component should be HW, Compiler, and field bus independent.	**Stability** C	Development

[3] Priority: 1 = mandatory, 2 = desirable, 3 = future
Stability: **C**ommited, **N**ot yet agreed, **L**ikely to change

4.3 Diagnosis Conditions Representation Requirements

4.3.1 Classification

Definition / Motivation	**Prio** 1	**Source**
Each diagnosis condition should be identified with one of the following classifications: **Status signal** / **Examples of detailed information** Check function: Change of configuration local operation, Substitute value entered Maintenance required: Maintenance needed short-term; Maintenance needed mid-term Failure: Cause of failure device-internal; Cause of failure process-related Off-specification: Device being operated off-spec; Uncertain value due to process influence	**Stability** C	NAMUR NE 107

4.3.2 Grouping

Definition / Motivation	**Prio** 1	**Source**

In the devices can be identified more sources of diagnosis conditions. NAMUR NE107 lists more error sources but for the major part of our instruments an error can be grouped as:
• Hardware Status
  • Electronics
  • Sensor
  • Actuator
• Operating Conditions
  • Process
  • Power Supply
• Configuration Status
  • Setup
 	
This is only a raw estimation of the groups needed by the different devices; any modification to this table shall be discussed.
	Stability C	NAMUR NE 107

4.3.3 Priority

Definition / Motivation	**Prio** 1	**Source**
Each diagnosis condition produced by a device should be identified with an internal priority level. NAMUR NE107 identifies several fault conditions and their priority for each device type. The priorities should be kept inside the device and will not be communicated via any interfaces. They shall be used to sort alarms in order to set the worst diagnosis condition in subsystems that can communicate single information (i.e. CurrentOut).	**Stability** C	NAMUR NE 107

4.3.4 Numbering

Definition / Motivation	**Prio** 1	**Source**
Each diagnosis condition shall be identified uniquely thought a numbering mechanism.	**Stability** C	Development

4.3.5 Additional Information

Definition / Motivation	**Prio** 1	**Source**
Every diagnosis condition shall be enriched by a description, a possible cause and a suggested action. Those information shall be added to make diagnosis condition independent from every Asset Monitor and they are protocol dependant: • For the HART, FF and PA that information shall be enumerated variables (the explanation text is stored into the eDD and in the DTM). • For the HMI the strings are inside the HMI.	**Stability** C	Development

4.4 Diagnostic Sub-System Functional Requirements

4.4.1 Summary Output

Definition / Motivation	**Prio** 1	**Source**
The diagnosis subsystem shall be able to provide a summary output of the device status, it shall be composed by: • The worst classification of all the active diagnosis condition. • All the groups in which an active diagnosis condition is set.	**Stability** C	Development

4.4.2 Detailed Output

Definition / Motivation	**Prio** 1	**Source**
The diagnosis subsystem shall be able to provide one by one each active diagnosis condition with all the additional NAMUR information linked with it.	**Stability** C	Development

4.4.3 Simulation

Definition / Motivation	**Prio** 1	**Source**
It shall be possible to simulate the diagnosis conditions of a device. During simulating an error, the whole device should behave as if the error has occurred in reality.	**Stability** C	Development

4.4.4 Simulation Storage Type

Definition / Motivation	**Prio** 1	**Source**
Simulation shall be a volatile configuration.	**Stability** C	Development

4.4.5 Simulation Rules

Definition / Motivation	**Prio** 1	**Source**
For Users not having Service Access Rights [5], it will be possible to simulate only one diagnosis condition at a time. However, for Users having Service Access Rights [5], it will be possible to simulate several diagnosis conditions simultaneously.	**Stability** C	Development

4.4.6 Simulation Enable/Disable Command

Definition / Motivation	**Prio** 1	**Source**
An overall Diagnostic Simulation Enable/Disable command should be provided in order to prevent that in normal operation the user forgets that diagnostic conditions remain simulated. When Diagnostic Simulation is Enabled, the diagnostic status of the device shall go in check function to remind the user that simulation is active. When the disable command is performed the Common Diagnosis Subsystem shall reset any pending simulation flag.	**Stability** C	Development

4.4.7 Masking

Definition / Motivation	**Prio** 1	**Source**
It shall be possible to mask each non-fatal diagnosis condition individually. When a diagnosis condition is masked, it means that although it has been detected as TRUE, it is treated as FALSE concerning the actions to be taken and its presentation to the external world. It shall be possible to set the masking condition to a default value (every condition shall be unset).	**Stability** C	Development

4.4.8 Masking Storage Type

Definition / Motivation	**Prio** 1	**Source**
Masking is a non-volatile configuration.	**Stability** C	Development

4.4.9 Simulation and Masking Access Rights

Definition / Motivation	**Prio** 1	**Source**
The diagnosis simulation and masking conditions shall be accessible by the user.	**Stability** C	Development

4.4.10 Alarm and Event Counter

Definition / Motivation	**Prio** 1	**Source**
Some diagnostic information, typically the process related ones, shall be counted through dedicated event counters with the purpose to be used for history tracking (warranty purpose) and predictive maintenance. The counters shall be accessible and reset-able by the user. A copy of these counters must be kept in Nov-Ram and be clearable only by factory operations.	**Stability** C	Development

4.4.11 Alarm and Event History

Definition / Motivation	**Prio** 1	**Source**
A simple history mechanism indicating whether a diagnostics condition appeared or not, is needed for all diagnostic conditions. This can be reset by the user. Alarm history information should be stored into diagnosis subsystem. At least a run time hour counter shall be implemented; when possible a real time clock should provide a time stamp for the alarm.	**Stability** C	Development

4.4.12 Detection Speed

Definition / Motivation	**Prio** 1	**Source**
The output of the status signal (e.g. current out) has to change before the current process value is given to the user interfaces.	**Stability** C	Development

4.4.13 Diagnostic Condition Acknowledge Mechanism

Definition / Motivation	**Prio** 1	**Source**
It must be possible that active diagnostics conditions can either be self-cleared or cleared by the user. This should be configurable by the developer.	**Stability** C	Development

4.4.14 Behavior for Alarm Detection

Definition / Motivation	**Prio** 1	**Source**
It must be possible that the developer can provide rules for defining what happens on the different outputs, if a diagnostic condition or combination of them occurs.	**Stability** C	Development

5 Performance/accuracy requirements

6 Design constraints
None

7 Safety Requirements/Error-detection and error-handling
The device, using the common diagnosis subsystem, shall be at least to fulfill the SIL2 safety requirements; refer to [2] for details. The process 5k software design and development process is enough to achieve the SIL2 certification, refers to Developer's Handbook - A Guideline for Developing Field-Devices in Business Unit Instrumentation for details.

8 Manufacturing
None

9 Other requirements
None

10 Risks
None

11 Plan by stages
The common diagnosis subsystem will be used for the first time on the EMF project, so it shall follow the EMF project plan.

12 Testability Issues
None

13 Revision Chart

Rev.	Description of Version/Changes	Primary Author(s)	Date
0.1	First Distribution for team internal comments	Giovanni Invernizzi, Joachim Lenk , Nick Malcolm, Florian Reuss, Jan Suchotzki	13/09/2005
0.2	First Draft for BUI distribution	Giovanni Invernizzi	17/11/2005
0.3	Second Draft after Diagnosis Subsystem Prototyping	Giovanni Invernizzi	23/01/2006
0.4	Four draft after Lenno meeting on 2006 June 07	Giovanni Invernizzi	27/03/2007