*International Journal of Innovative Computing, Information and Control*
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **549–560**

# DDL DEVELOPMENT FOR APPLICATIONS BASED ON HART PROTOCOL

Alexandre Baratella Lugli and Ana Carolina de Oliveira Patrício

Department of Industrial Automation
National Institute of Telecommunications (INATEL)
CEP 37.540-000, Santa Rita do Sapucaí, MG, Brazil
baratella@inatel.br

Received May 2017; revised September 2017

Abstract. *This article aims to describe a study about the HART (Highway Addressable Remote Transducer) protocol and the steps needed to develop a DDL (Device Description Language), showing its operation and its implementation for a temperature transmitter compatible with the HART communication protocol. The results are shown in the end of the article and describe at the DDL for the application shown in this article. The DDL was developed, tested and validated in a real temperature transmitter.*
**Keywords:** Application, DDL, Development, HART protocol, Software

## 1. Introduction.

The industrial networks emerged and were developed from a huge global discussion related to the issue by the end of the 1990s. The goal was to create a new standard to standardize all the industrial protocols [1,2].

Besides that, the industrial protocols have become more developed and have remained in wide application in industries to the current days [1,2]. Other articles already studied the DDL (*Device Description Language*) applied for HART (*Highway Addressable Remote Transducer*) network, such as [3,4], but they did not show the development and applications in detail.

The manuscripts [5,6] show an application with HART protocol and the use of the DDL file for process industry, evidencing the HART communication.

The objective of this research is to conduct a study about the industrial protocol HART focusing on the development of a DDL. Thus, it will accomplish a full development and presentation of a DDL file, and, later, laboratory tests of the communication between the HART network and the temperature transmitter, where the remote configuration of the commands implemented in the device description (DD) is set up. After some simulations, it is possible to see and understand how the DDL language is embedded in the *host* and how it is used to model the behavior of field devices applied in the HART network.

## 2. Concepts and Definitions.

This chapter shows the development of a DDL and its main functions.

### 2.1. HART protocol.

HART is a bidirectional communication protocol which allows the data access between smart field instruments and the *host* system (centralized) from any location in the process plant.

The *host* can be any computer application, portable device (laptop, handheld computers and even Smartphone) or any other systems that use some kind of control platform which enables a faster factory management, thus offering a more reliable and durable solution [3].

Introduced in the 90s, this technology emerged due to a greater need of interaction between the user and the instruments, either by configuring their functions, reading variables or diagnosing states. This demand grew as the process automation equipment's electronics evolved [7,8].

Its physical layer has a transmission rate of 1200bps if it is based on the Bell 202 standard of FSK (Frequency Shift Key), in which the frequency 1200Hz represents the bit "1" and the frequency 2200Hz represents the bit "0" [7,8].

The data are communicated simultaneously with the 4-20mA signal without interference and through the same wiring. As a signal modulated in FSK has null mean value, from the theory of communication, it does not interfere with the analog control system. Therefore, a signal modulated in current is more robust to electromagnetic interferences [1,8].

Figure 1 shows a communication signal superposed on the analog 4-20mA. The 4-20mA signal indicates the main process variable, and the communication signal indicates the device situation (status, configurations, diagnostics and others) [7,8].

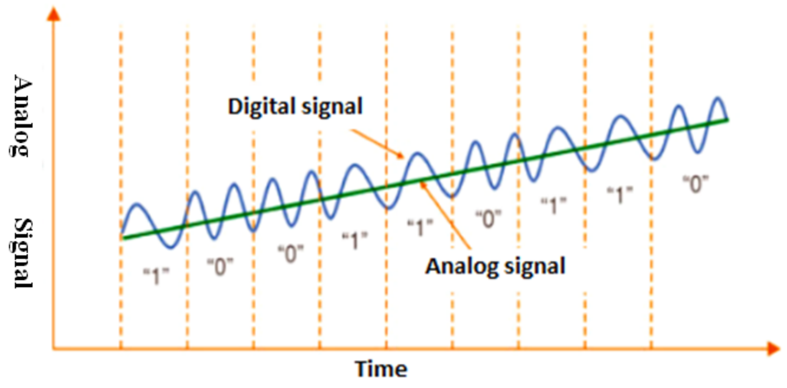


FIGURE 1. Frequency Shift Key (FSK) [9]

The HART protocol is based on the master/slave communication in which the slave only transmits if the message contains a request from the master. The communication with the equipment HART is generally done through an interface called "Modem HART", enabling real-time access between the *host* and the instrument [8,9].

2.2. **Temperature transmitter.** The temperature is one of the most measured physical variables in industrial processes, and usually ends up being a crucial factor in the industrial procedure. If the measurement is not reliable and accurate for some reason, it can have negative effects on the efficiency and quality of the products [10].

The temperature measurement in the industry field is frequently done using a sensor and a transmitter that converts the primary signal (sensor) in a current signal proportional to the signal generated at the input. Moreover, a transmitter is designed to measure temperatures using not only thermo-elements or thermo-resistances, but also other types of sensors with resistance and mili-voltage at their output. The technology used in the equipment allows the transmitter to have a simple interface between the field and the control room [10].

The temperature transmitter is often powered by a voltage of 12 to 45Vcc, and modulates the communication over an output current of 4-20mA. In this work, the instrument will use the HART communication protocol and, through a tool based on the DDL standards, the configuration, monitoring and diagnostic functions will be implemented [10].

The modularization of the temperature transmitter components, which uses the HART protocol, is described in the block diagram of Figure 2. The primary sensor's signal is converted from an analog value to a digital value, and then goes through a filter. This

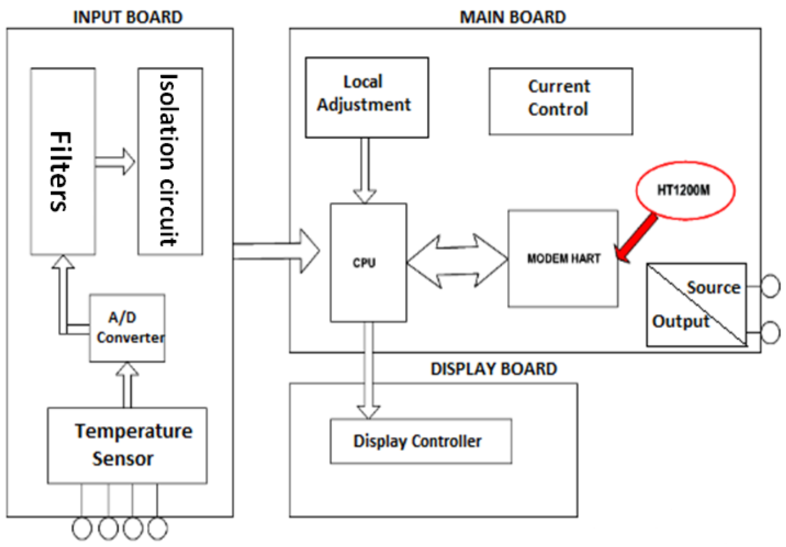


FIGURE 2. Block diagram of the temperature transmitter [10]

digital value is converted to temperature according to the sensor selected. Moreover, the temperature value is delivered to the output of the instrument as a current proportional to the calibrated range. It is also important to note that the sensor needs to be galvanically isolated from the output signal in order to avoid possible temperature measurement errors due to leakage currents to the ground [10].

2.3. **DDL.** *Device Description Language* (DDL) [10] has appeared from a necessity of having a language to model the behavior of field instruments and maintain a high compatibility between them regardless of the model or manufacturer. Due to the increasing number of equipment, it turned to be difficult and expensive to the manufacturers to keep their systems up to date as soon as many new types of equipment were emerging in the market.

The DDL enables detailed calibration, configuration, and diagnosis for a fast troubleshooting of the field equipment using applications (*host*). It also describes in a standardized way the characteristics of the equipment and informs to the system how to interface with the equipment, which commands should be sent, how to interpret the response, and how the data should be displayed [12].

Through text files called *Device Description* (DD), the control systems, configurations, handheld and other software tools can communicate with each instrument evenly, which permits the description of the menu layout structure that the user will see when communicating with the equipment. It is important to note that the DDL is not an open technology. The DD text files are translated and supplied as standard binary files from the equipment manufacturer and they allow that menus, methods and parameters to be presented as desired.

The general structure of a DDL is formed from a group of objects. Each object has an important function in the organization, exchange and presentation of data to the user. The objects are summarized in [12,13]:

- **Data** (Parameters)
- **Communication** (Commands)
- **Graphics** (Menus)
- **Operation** (Methods)

To develop a DDL file, it is necessary to understand some essential standards for the study and implementation of a device language. This article will exemplify only some steps necessary to create it [12].

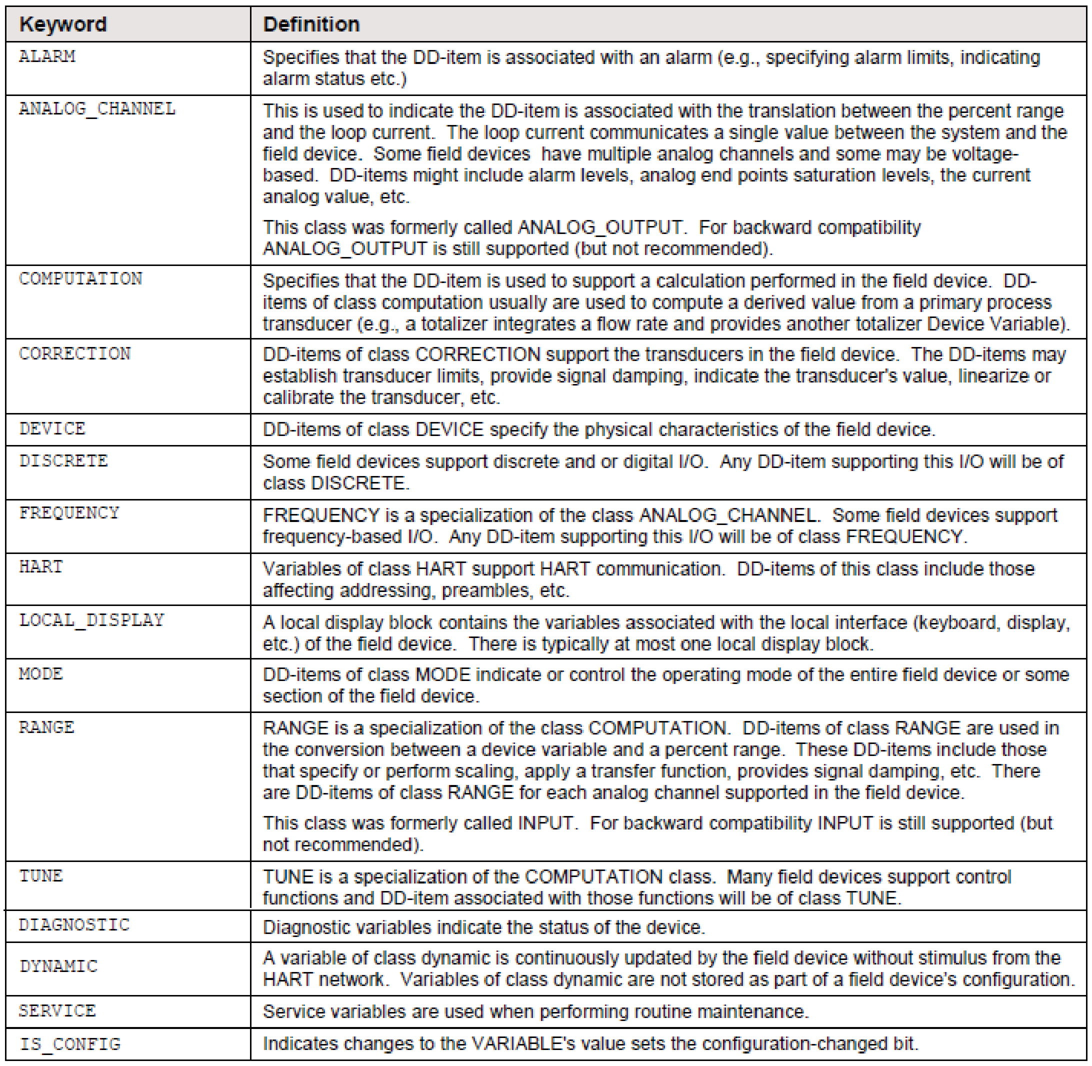
- **Creating Variables**

The variable is the main mechanism for modeling the instrument data, which could be any information contained therein. The *VARIABLE* object describes a large range of proprieties allowing the data to be completely described [11].

Every variable must have a name to be referenced anywhere in the DDL. As it can see below, there are ten attributes that the variable can support, and the only mandatory field is the *TYPE* [11].

- *LABEL*: It specifies the text that the host applications must exhibit when a variable is referenced.
- *HELP*: It specifies details about a variable when requested by the user.
- *CLASS*: It is used to model a behavior of a variable. There are many types of class, as shown in Table 1.
- *VALIDITY*: Some equipment can operate in different modes. As an example, a temperature transmitter can be used either as a temperature measurer or as a signal

Table 1. Variable classes [12]

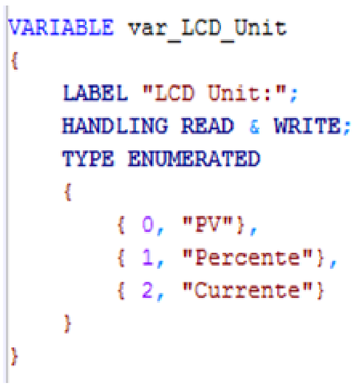


| Keyword | Definition |
|---|---|
| ALARM | Specifies that the DD-item is associated with an alarm (e.g., specifying alarm limits, indicating alarm status etc.) |
| ANALOG_CHANNEL | This is used to indicate the DD-item is associated with the translation between the percent range and the loop current. The loop current communicates a single value between the system and the field device. Some field devices have multiple analog channels and some may be voltage-based. DD-items might include alarm levels, analog end points saturation levels, the current analog value, etc. This class was formerly called ANALOG_OUTPUT. For backward compatibility ANALOG_OUTPUT is still supported (but not recommended). |
| COMPUTATION | Specifies that the DD-item is used to support a calculation performed in the field device. DD-items of class computation usually are used to compute a derived value from a primary process transducer (e.g., a totalizer integrates a flow rate and provides another totalizer Device Variable). |
| CORRECTION | DD-items of class CORRECTION support the transducers in the field device. The DD-items may establish transducer limits, provide signal damping, indicate the transducer's value, linearize or calibrate the transducer, etc. |
| DEVICE | DD-items of class DEVICE specify the physical characteristics of the field device. |
| DISCRETE | Some field devices support discrete and or digital I/O. Any DD-item supporting this I/O will be of class DISCRETE. |
| FREQUENCY | FREQUENCY is a specialization of the class ANALOG_CHANNEL. Some field devices support frequency-based I/O. Any DD-item supporting this I/O will be of class FREQUENCY. |
| HART | Variables of class HART support HART communication. DD-items of this class include those affecting addressing, preambles, etc. |
| LOCAL_DISPLAY | A local display block contains the variables associated with the local interface (keyboard, display, etc.) of the field device. There is typically at most one local display block. |
| MODE | DD-items of class MODE indicate or control the operating mode of the entire field device or some section of the field device. |
| RANGE | RANGE is a specialization of the class COMPUTATION. DD-items of class RANGE are used in the conversion between a device variable and a percent range. These DD-items include those that specify or perform scaling, apply a transfer function, provides signal damping, etc. There are DD-items of class RANGE for each analog channel supported in the field device. This class was formerly called INPUT. For backward compatibility INPUT is still supported (but not recommended). |
| TUNE | TUNE is a specialization of the COMPUTATION class. Many field devices support control functions and DD-item associated with those functions will be of class TUNE. |
| DIAGNOSTIC | Diagnostic variables indicate the status of the device. |
| DYNAMIC | A variable of class dynamic is continuously updated by the field device without stimulus from the HART network. Variables of class dynamic are not stored as part of a field device's configuration. |
| SERVICE | Service variables are used when performing routine maintenance. |
| IS_CONFIG | Indicates changes to the VARIABLE's value sets the configuration-changed bit. |

controller. In this way, some transmitter functions will not be available when operating in controller mode. The attribute *VALIDITY* verifies the appropriate mode in order to enable or not the variable to the user.

- *HANDLING*: It specifies the operation that the *host* must execute. The options are *READ* and *WRITE. READ* indicates that the variable must be read by the equipment and *WRITE* indicates that the value must be written. Most equipment has READ & WRITE as standard.
- *CONSTANT_UNIT*: If the variable has a unitary code that does not need to be changed, it can be specified by this attribute (e.g., current is Always in Ampere).
- *DEFAULT_VALUE*: It allows the DDL to change values through the *host* when the equipment is not physically connected (operating on off-line mode).
- *TYPE*: This is a required item because it informs the host which data format will be used and how many bytes. In addition, the data types of the variable for DDL are *Integer, Unsigned Integer, Float, Double, Enumerated, Index, ASCII, Password, Date* and *Time_Value*. Some of these types (*Integer, Unsigned Integer, Float* and *Double*) have additional information of formation, scale and boundary.
- *Pre/Post actions*: An action must be executed before or after reading/writing or editing a value of a variable in the equipment. The methods do not have preferences that make them run in the order in which they were defined.
- *REFRESH_ACTIONS*: The methods defined in this attribute are executed before displaying or updating the variable on the screen.

Figure 3 is an example of a *VARIABLE*, titled LCD Unit, of operations read and write of enumerated type.



```
VARIABLE var_LCD_Unit
{
    LABEL "LCD Unit:";
    HANDLING READ & WRITE;
    TYPE ENUMERATED
    {
        { 0, "PV"},
        { 1, "Percente"},
        { 2, "Currente"}
    }
}
```

FIGURE 3. Example of variables in the equipment

- **Creating Command**

The command is a data packet used to establish an exchange of information between the *host* and the HART device. Each command needs to have a name and can be referenced in other parts of the DDL. There are three mandatory types of command operations.

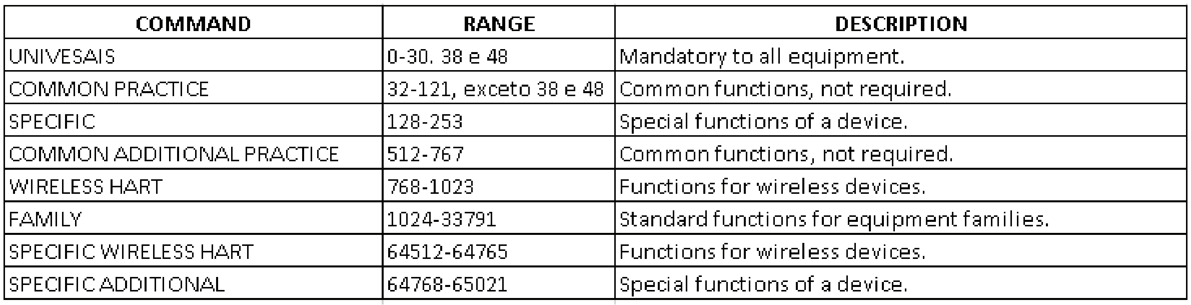
*NUMBER*: It specifies the number of HART commands on the equipment. Table 2 shows how the protocol standard divides the command interval.

*OPERATION*: There are three operation options that need to be done after receiving the command [11]:

*READ*: When an equipment receives a read command, it must return the updated values referenced to the variables (indicated in *REPLY* as shown in Figure 10).

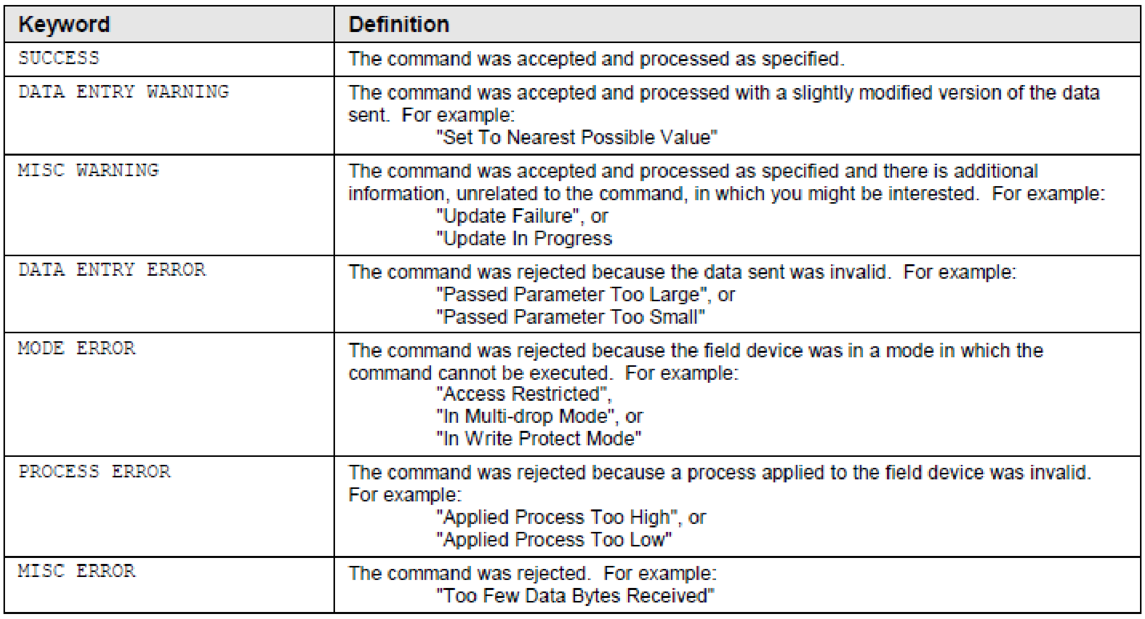
*WRITE*: When an equipment receives a write command, it assigns values received from the *host*.

Table 2. HART command ranges [7,14,15]



| COMMAND | RANGE | DESCRIPTION |
|---|---|---|
| UNIVESAIS | 0-30. 38 e 48 | Mandatory to all equipment. |
| COMMON PRACTICE | 32-121, exceto 38 e 48 | Common functions, not required. |
| SPECIFIC | 128-253 | Special functions of a device. |
| COMMON ADDITIONAL PRACTICE | 512-767 | Common functions, not required. |
| WIRELESS HART | 768-1023 | Functions for wireless devices. |
| FAMILY | 1024-33791 | Standard functions for equipment families. |
| SPECIFIC WIRELESS HART | 64512-64765 | Functions for wireless devices. |
| SPECIFIC ADDITIONAL | 64768-65021 | Special functions of a device. |

Table 3. *RESPONSE_CODES* types [12]



| Keyword | Definition |
|---|---|
| SUCCESS | The command was accepted and processed as specified. |
| DATA ENTRY WARNING | The command was accepted and processed with a slightly modified version of the data sent. For example: "Set To Nearest Possible Value" |
| MISC WARNING | The command was accepted and processed as specified and there is additional information, unrelated to the command, in which you might be interested. For example: "Update Failure", or "Update In Progress |
| DATA ENTRY ERROR | The command was rejected because the data sent was invalid. For example: "Passed Parameter Too Large", or "Passed Parameter Too Small" |
| MODE ERROR | The command was rejected because the field device was in a mode in which the command cannot be executed. For example: "Access Restricted", "In Multi-drop Mode", or "In Write Protect Mode" |
| PROCESS ERROR | The command was rejected because a process applied to the field device was invalid. For example: "Applied Process Too High", or "Applied Process Too Low" |
| MISC ERROR | The command was rejected. For example: "Too Few Data Bytes Received" |

*COMMAND*: As soon as this command is received, the equipment executes a specific action.

*TRANSACTION*: This command is not mandatory. Its function is to make a choice when the same command has many interpretations [11].

*RESPONSE_CODES*: This attribute is used in the line of code to, depending on the value returning from the equipment, having an answer regarding its status. A *RESPONSE_CODE* is composed of an integer value, a type (as shown in Table 3), and a description which specifies what the *host* must show when this command is returned from the equipment.

Figure 4 exemplifies a creation of a specific command, which was named 152, used to execute a specific action of reading or writing.

## 3. Practical Application.

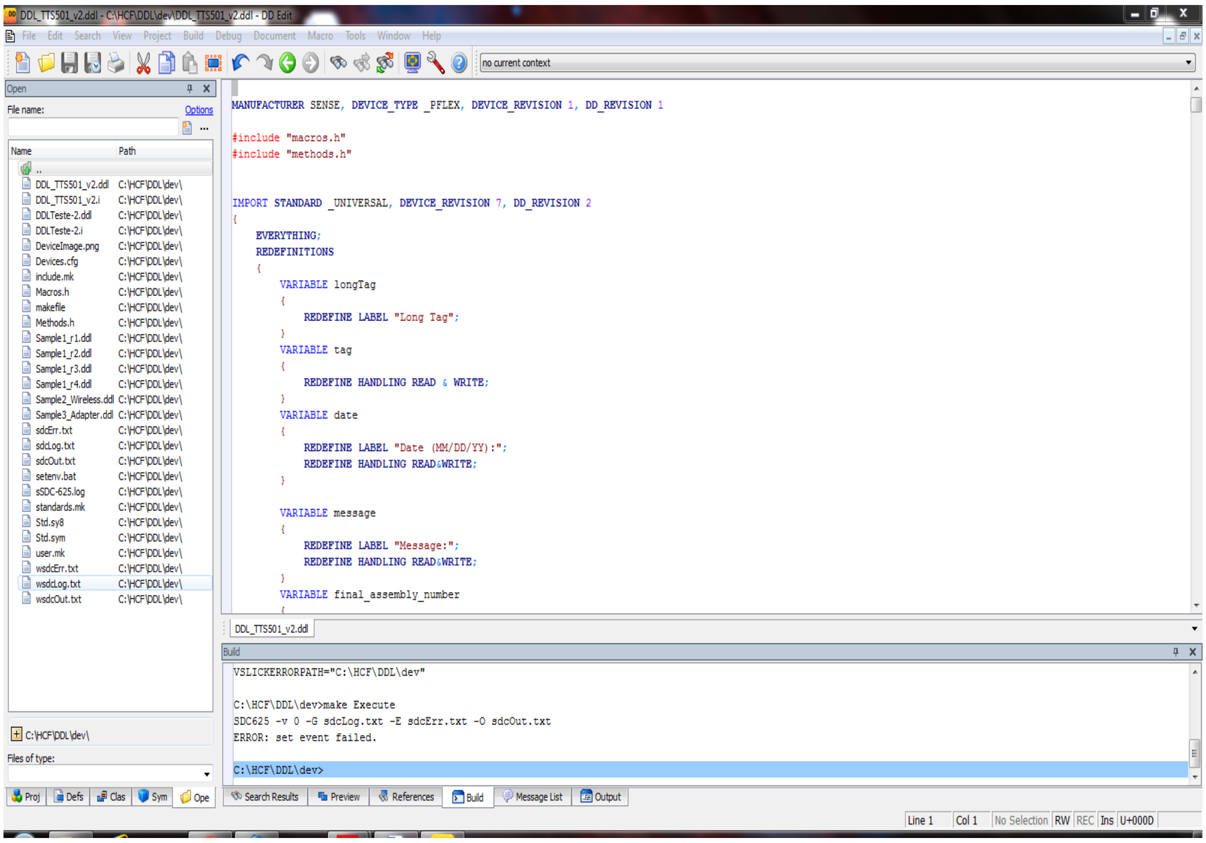
The goal of the practical part of this work is to develop a DDL, using a *DD Edit* computer application, implementing the *Loop Test* function and demonstrating its functionality through a temperature transmitter compatible with the HART communication protocol.

### 3.1. Development environment.

The HART *Foundation* provides a computational tool called DD-Edit Software (shown in Figure 5) to standardize and facilitate the creation of a DDL. Besides this computational application allows the developer to write, test and

```
COMMAND command152_setup_command
{
    NUMBER 152;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
            var_setup_valve
        }
        REPLY
        {
            response_code, device_status,var_setup_valve
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,            [no_command_specific_errors];
        2,  DATA_ENTRY_ERROR,   [invalid_selection];
        5,  DATA_ENTRY_ERROR,   [too_few_data_bytes_recieved];
        7,  MODE_ERROR      ,   [in_write_protect_mode];
    }
}
```

Figure 4. Example of a command in the equipment


DDL_TTS501_v2.ddl - C:\HCF\DDL\dev\DDL_TTS501_v2.ddl - DD Edit
File Edit Search View Project Build Debug Document Macro Tools Window Help
no current context
Open
File name:
Options
C:\HCF\DDL\dev\
Files of type:
Proj Defs Clas Sym Ope
DDL_TTS501_v2.ddl
Build
Search Results Preview References Build Message List Output
Line 1 Col 1 No Selection RW REC Ins U+000D

| Name | Path |
|---|---|
| .. | |
| DDL_TTS501_v2.ddl | C:\HCF\DDL\dev\ |
| DDL_TTS501_v2.i | C:\HCF\DDL\dev\ |
| DDLTeste-2.ddl | C:\HCF\DDL\dev\ |
| DDLTeste-2.i | C:\HCF\DDL\dev\ |
| DeviceImage.png | C:\HCF\DDL\dev\ |
| Devices.cfg | C:\HCF\DDL\dev\ |
| include.mk | C:\HCF\DDL\dev\ |
| Macros.h | C:\HCF\DDL\dev\ |
| makefile | C:\HCF\DDL\dev\ |
| Methods.h | C:\HCF\DDL\dev\ |
| Sample1_r1.ddl | C:\HCF\DDL\dev\ |
| Sample1_r2.ddl | C:\HCF\DDL\dev\ |
| Sample1_r3.ddl | C:\HCF\DDL\dev\ |
| Sample1_r4.ddl | C:\HCF\DDL\dev\ |
| Sample2_Wireless.ddl | C:\HCF\DDL\dev\ |
| Sample3_Adapter.ddl | C:\HCF\DDL\dev\ |
| sdcErr.txt | C:\HCF\DDL\dev\ |
| sdcLog.txt | C:\HCF\DDL\dev\ |
| sdcOut.txt | C:\HCF\DDL\dev\ |
| setenv.bat | C:\HCF\DDL\dev\ |
| sSDC-625.log | C:\HCF\DDL\dev\ |
| standards.mk | C:\HCF\DDL\dev\ |
| Std.sy8 | C:\HCF\DDL\dev\ |
| Std.sym | C:\HCF\DDL\dev\ |
| user.mk | C:\HCF\DDL\dev\ |
| wsdcErr.txt | C:\HCF\DDL\dev\ |
| wsdcLog.txt | C:\HCF\DDL\dev\ |
| wsdcOut.txt | C:\HCF\DDL\dev\ |

```
MANUFACTURER SENSE, DEVICE_TYPE _PFLEX, DEVICE_REVISION 1, DD_REVISION 1

#include "macros.h"
#include "methods.h"


IMPORT STANDARD _UNIVERSAL, DEVICE_REVISION 7, DD_REVISION 2
{
    EVERYTHING;
    REDEFINITIONS
    {
        VARIABLE longTag
        {
            REDEFINE LABEL "Long Tag";
        }
        VARIABLE tag
        {
            REDEFINE HANDLING READ & WRITE;
        }
        VARIABLE date
        {
            REDEFINE LABEL "Date (MM/DD/YY):";
            REDEFINE HANDLING READ&WRITE;
        }

        VARIABLE message
        {
            REDEFINE LABEL "Message:";
            REDEFINE HANDLING READ&WRITE;
        }
        VARIABLE final_assembly_number
        {
```

```
VSLICKERRORPATH="C:\HCF\DDL\dev"

C:\HCF\DDL\dev>make Execute
SDC625 -v 0 -G sdcLog.txt -E sdcErr.txt -O sdcOut.txt
ERROR: set event failed.

C:\HCF\DDL\dev>
```

Figure 5. Screen of the *DD Edit* computational tool

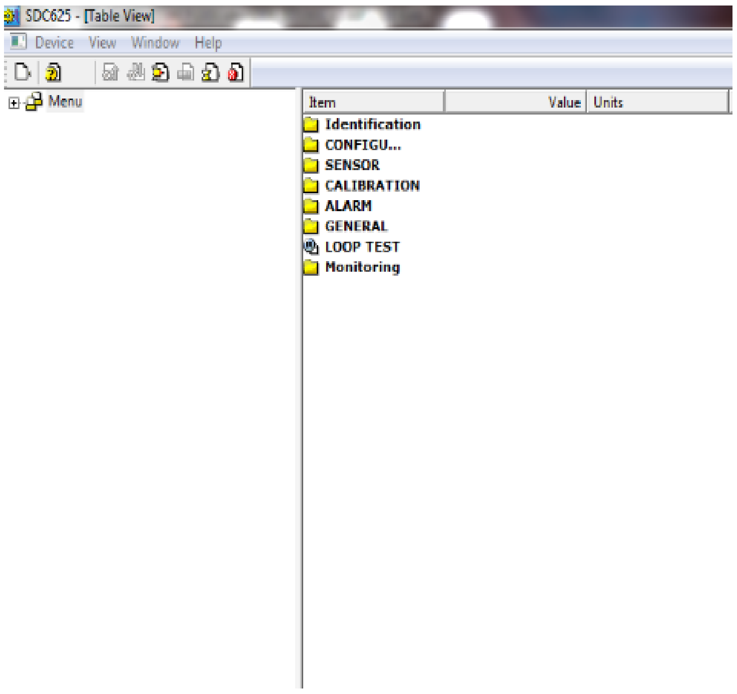


Figure 6. Screen of the *SDC625*

analyze method errors (through breakpoints), and it also allows to simulate the field equipment if not connected (off-line mode) to test the DDL.

In order to test the DDL with the field equipment connected to the application, the used tool is the *SDC625* (illustrated in Figure 6) that acts as a *host*. All the objects defined in the DDL, such as variables, menus, commands, methods and graphic functions, are interpreted by the *SDC625* and displayed to the user in real time. This application must be used to validate a DD file before being registered on the HART Foundation for the purpose of the DDL to be free of errors and to behavior as expected.

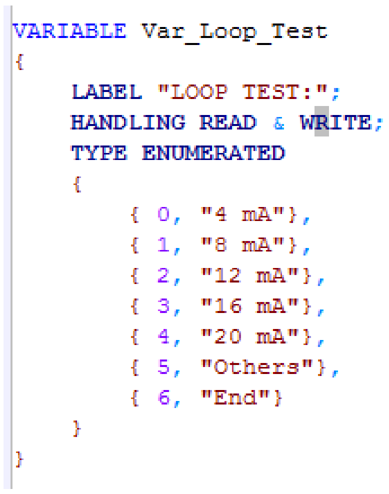
3.2. **Loop test.** The *Loop Test* function is intended to check the accuracy of the output current or the analog output. Because it is an evaluation of the equipment calibration, it is tested if the output circuit is closed and if the series connected instruments are set to a fixed current value (source simulation). This fixed current mode is assigned as a test command, which is common, but not mandatory. In the HART standard of test command specification, it is referred to as command 40 [13].

Figures 7 and 8 show how the codes were typed to create the *Loop Test* function. The attributes, previously explained, were implemented for the purposes of facilitating the handling and configuring the equipment. Figure 9 shows how the created parameter is displayed to the user through the *SDC625 host*.

3.3. **Assembly and testing.** All the assembly and testing of the developed DDL were conducted in the test environment of the company that developed the test equipment.

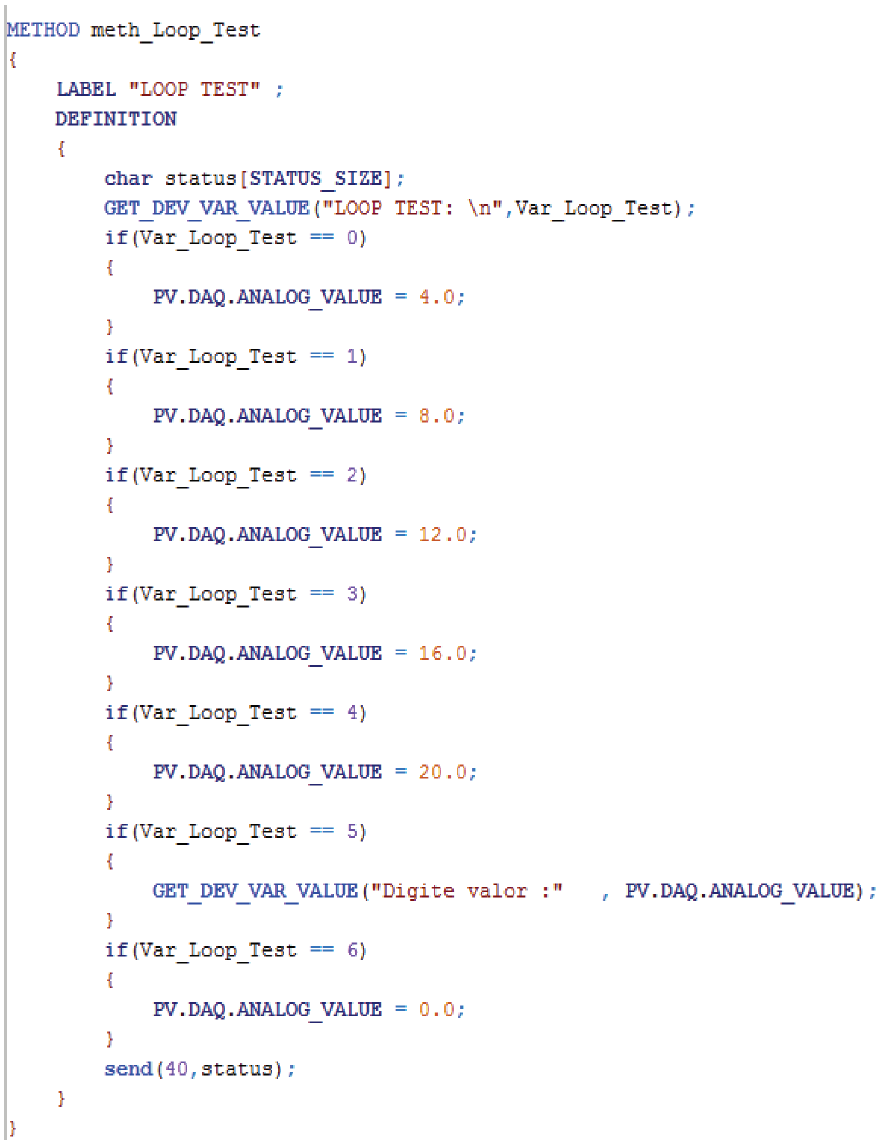
The source of the temperature transmitter was the computer USB (Universal Serial Bus) port through a communication interface which operates with low power, insulation, and is compatible with the HART protocol. Figure 10 shows the equipment that facilitates the commissioning, operation, maintenance and calibration processes, diagnosis and any HART instrument data acquisition.

An ammeter was placed in series with the transmitter power to visualize the output current of the field equipment. To measure the temperature, an instrument called PRESYS



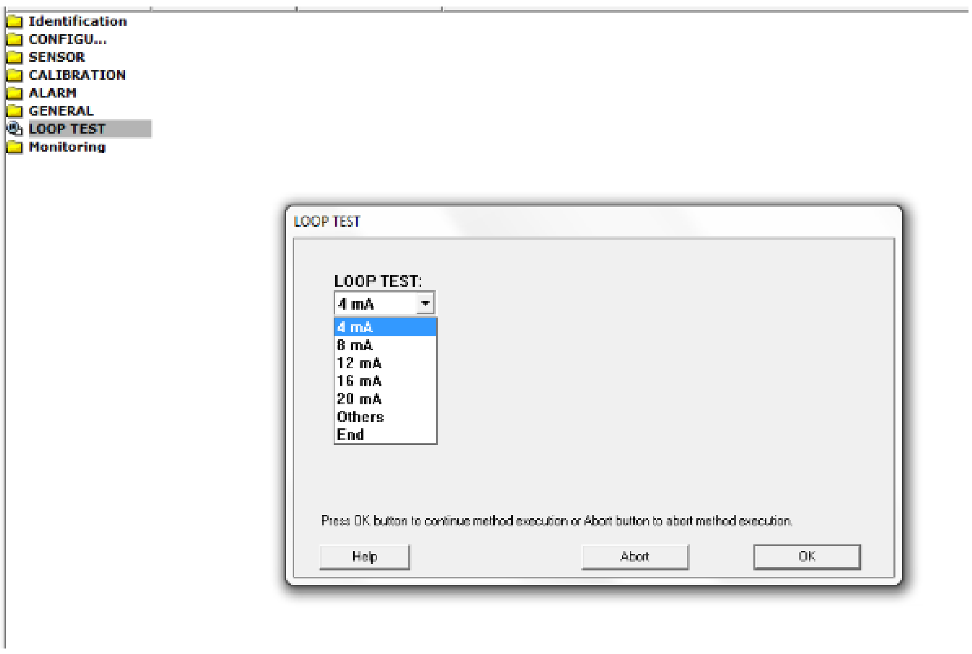
```
VARIABLE Var_Loop_Test
{
    LABEL "LOOP TEST:";
    HANDLING READ & WRITE;
    TYPE ENUMERATED
    {
        { 0, "4 mA"},
        { 1, "8 mA"},
        { 2, "12 mA"},
        { 3, "16 mA"},
        { 4, "20 mA"},
        { 5, "Others"},
        { 6, "End"}
    }
}
```

Figure 7. Lines of code to create a *Loop Test* variable



```
METHOD meth_Loop_Test
{
    LABEL "LOOP TEST" ;
    DEFINITION
    {
        char status[STATUS_SIZE];
        GET_DEV_VAR_VALUE("LOOP TEST: \n",Var_Loop_Test);
        if(Var_Loop_Test == 0)
        {
            PV.DAQ.ANALOG_VALUE = 4.0;
        }
        if(Var_Loop_Test == 1)
        {
            PV.DAQ.ANALOG_VALUE = 8.0;
        }
        if(Var_Loop_Test == 2)
        {
            PV.DAQ.ANALOG_VALUE = 12.0;
        }
        if(Var_Loop_Test == 3)
        {
            PV.DAQ.ANALOG_VALUE = 16.0;
        }
        if(Var_Loop_Test == 4)
        {
            PV.DAQ.ANALOG_VALUE = 20.0;
        }
        if(Var_Loop_Test == 5)
        {
            GET_DEV_VAR_VALUE("Digite valor :"    , PV.DAQ.ANALOG_VALUE);
        }
        if(Var_Loop_Test == 6)
        {
            PV.DAQ.ANALOG_VALUE = 0.0;
        }
        send(40,status);
    }
}
```

Figure 8. Lines of code to define the *Loop Test*



Press OK button to continue method execution or Abort button to abort method execution.

FIGURE 9. How the *Loop Test* function is displayed to the user

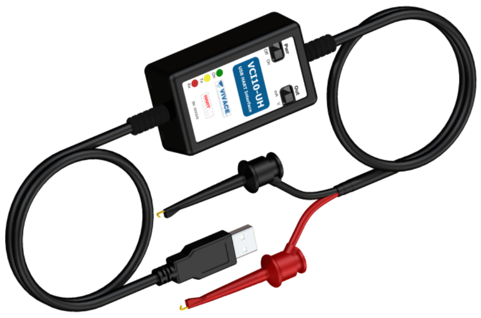


FIGURE 10. USB-HART communication interface [16]

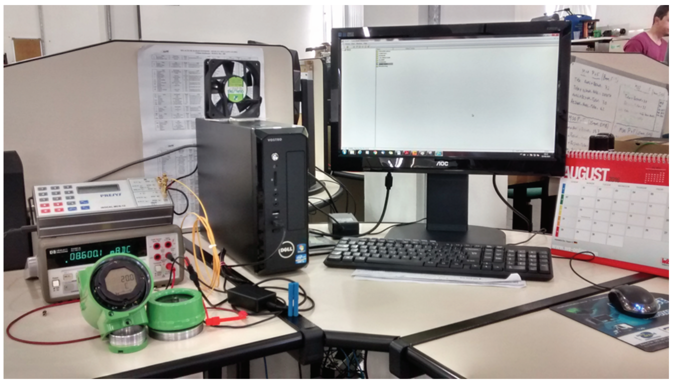
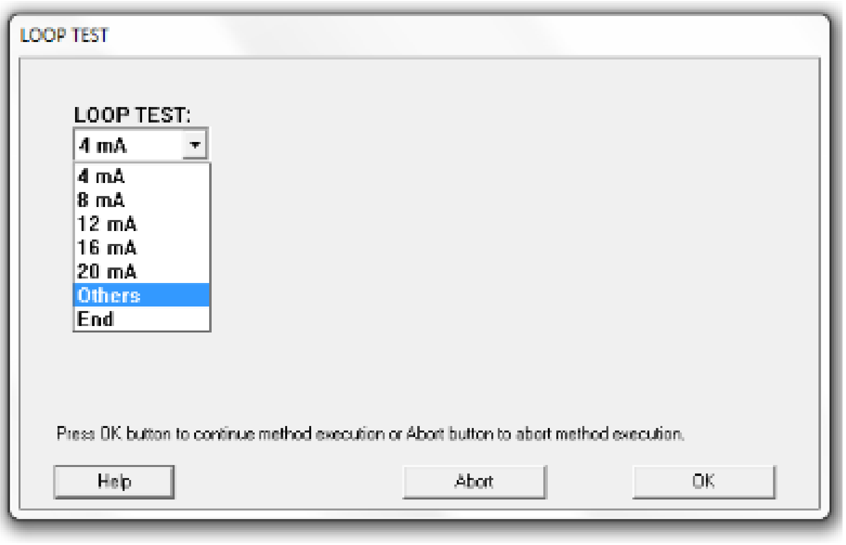


FIGURE 11. Settings used for testing

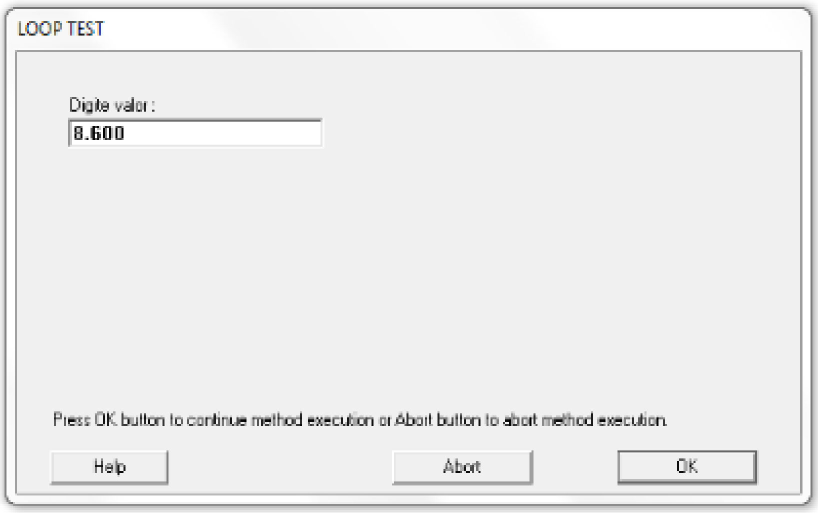
was used to simulate the two-wire connection of a thermo-element, where the output mili-voltage changes whenever the temperature varies.

Figure 11 shows how the equipment was assembled for the practical application of the device description created for the *Loop Test* function.



Press OK button to continue method execution or Abort button to abort method execution.

FIGURE 12. Screen to choose other options of output current



Digite valor:
8.600

Press OK button to continue method execution or Abort button to abort method execution.

FIGURE 13. Screen that allows the user to type any current

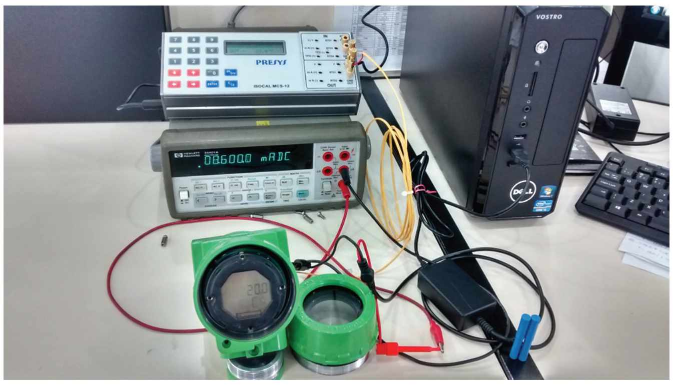


FIGURE 14. Ammeter showing the fixed 8.6mA output current

In the DDL created, it implemented an "Others" option that allows the user to choose which current value he wants to set at the equipment output, besides the options provided by the manufacturer. To exemplify this application, a value of 8.6mA was selected as shown in Figures 12, 13 and 14.

If the user wants to leave the option *Loop Test*, just choose the option "End", and the output returns to the current corresponding to the temperature measured by the equipment.

4. **Conclusion.** The industrial protocols have become essential in industrial automation projects in the last decade. The field instruments must provide the user a wide range of diagnostic, parameters and settings in order to be performed remotely and quickly. Therefore, this manuscript illustrates the complete development of a DD file, using DDL language, applied to HART protocol. It is notable by the practical application results that there is a very large range of parameters, diagnostics or configurations that can be developed by the manufacturer. So, the DDL file was tested and validated in an application with measure temperature, where the measures were satisfied with the project.

Thus, after the conclusion of this work, it is remarkable of the huge contribution of its development if compared to point-to-point systems, where the PLC (Programmable Logical Controller) is the central element.

## REFERENCES

[1] M. Felser and T. Sauter, The fieldbus war: History or short break between battles?, *The 4th IEEE International Workshop on Factory Communication System*, Sweden, pp.73-80, 2002.

[2] M. Felser and T. Sauter, Standardization of industrial Ethernet – The next battlefield?, *The 6th IEEE International Workshop on Factory Communication Systems*, Sweden, pp.413-421, 2004.

[3] F. Y. Zulkifli and G. Schneiderheinze, Generic device description for complex HART field devices, *The 8th International Conference on IEEE Article Communication Systems*, 2002.

[4] W. Kastner and F. Kastner-Masilko, EDDL inside FDT/DTM, *Proc. of IEEE International Workshop on Factory Communication Systems*, 2004.

[5] A. M. Mulaosmanović, Application of the HART protocol for communication with smart field devices, *Vojnotehnički Glasnik NewsPaper*, vol.63, no.3, pp.160-175, 2015.

[6] P. Pornchai, T. Teerawat, P. Sawai and R. Apinai, Integration of wireless HART network system into SCADA software for operation & management, *The 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2016.

[7] HART Protocol Norm, *Universal HART Communication Protocol Specification (HCF_SPEC-13)*, HART Communication Foundation, 2013.

[8] J. Liu et al., PROFIBUS DP and HART protocol conversion and the gateway development, *IEEE Conference on Industrial Electronics and Applications*, pp.15-20, 2007.

[9] *VIVACE Process Instruments – HART 7: Protocol Detail*, http://www.vivaceinstruments.com.br/pt/artigo/hart-7-detalhando-o-protocolo, 2016.

[10] *SENSE Eletrônica – Install and Operation Datasheet on HART*, Sense Sensores & Instruments, 2016.

[11] HART Protocol Norm, *Temperature Device Family Specification (HCF_SPEC-160.4)*, HART Communication Foundation, 2011.

[12] HART Protocol Norm, *Device Description Language Specification (HCF_SPEC-500)*, HART Communication Foundation, 2011.

[13] HART Protocol Norm, *Device Description – Language Method Standard – Library Specification (HCF_SPEC-501)*, HART Communication Foundation, 2011.

[14] HART Protocol Norm, *Common Practice Command Specification (HCF_SPEC-151)*, HART Communication Foundation, 2012.

[15] HART Protocol Norm, *Universal Command Specification (HCF_SPEC-127)*, HART Communication Foundation, 2008.

[16] VIVACE Process Instruments, *Install and Operation Datasheet on Hart® USB*, VIVACE, 2016.