Borst Automation Embedded Solutions
Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Technical Article

Device Description Language

The HTML of the Fieldbuses

Walter Borst
Fachingen, October 15, 2011

The first Idea

To answer the question about the origin of the Device Description, one could refer to various references, none of which, however, determine the exact point in time of the actual idea. But who is better to answer the question of the origin than one of the inventors?

Strictly speaking, the model of the Device Description Language as it is used today was created in the summer of 1989. The companies Eckardt, Endress+Hauser, Esters Elektronik, Krohne, Neles-Jamesbury, Rosemount, Samson and Valmet undertook a project in which a fully functioning fieldbus was to be presented for the first time. The aim was to show how such a system must be structured in order to ensure the so-called interoperability of devices from different manufacturers.

"*Anyone who wanted to integrate a device into the system should describe this so that the visualization software could be tailored to it. While the technical implementation was clear to me, I had to come up with something to collect the necessary data. A description in prose seemed too imprecise for me and would have meant countless queries. This would have been very tedious, especially with the partners from the USA and Sweden. So I designed a form with the help of which the parameters could be formally described.*"

Parameter Name:	_____
Type:	_____
Label:	_____
Number of Digits:	_____
Max Fract Digits:	_____
Sign:	_____
Unit:	_____
Class:	_____
Maximum Value:	_____
Minimum Value:	_____

So far the report by Walter Borst, who at the time was working at Endress+Hauser as the head of the microprocessor system technology department and who was responsible for the overall technical project management of the fieldbus demonstration in 1989 on behalf of E+H. The form that was used at that time looked like the graphic on the left.

The use of such a form guaranteed the unambiguous determination of all necessary properties of a parameter. Here's an example:

Parameter Name:	*Measured Level* _____
Matrix Line:	*0* ____
Matrix Line:	*0* ____
Type:	*Float* _____
Label (16 Chars):	*MEASURED VALUE* _____
Number of Digits:	*4* ____
Max Fract Digits:	*1* ____
Sign:	*None* _
Unit:	*"%"* _____
Class:	*Dynamic* _____
Maximum Value:	*120.0* _____
Minimum Value:	*0.0* _____

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

The early 'Fieldbus Systems'

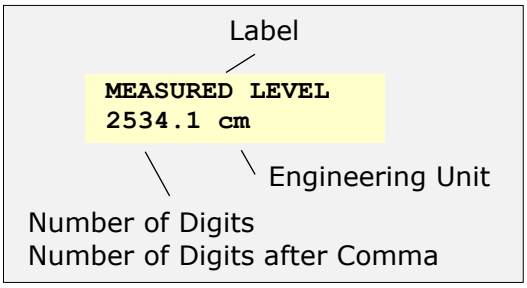
To fully understand the development of DD, however, one has to go back to the early 1980s. Microprocessors had already taken a central place in measuring device technology. Since these processors also had external interfaces, the developers were already familiar with various options for digital communication. At the INTERKAMA 1983, Honeywell presented a HandHeld terminal for the first time, with which a transmitter installed in the field could be parameterized. That in itself was nothing special. However, the fact that the necessary communication was superimposed on the 4..20 mA signal made the presentation a sensation.

The disadvantage of the Honeywell system was that the communication signal caused the power signal to fluctuate significantly. Companies such as E+H and Rosemount took up the idea and developed systems that did not have this disadvantage and showed the results at INTERKAMA 1986.

In addition to the field-mounted compact converters, E+H also had a system in which the signal-processing microprocessor was housed in a component that was mounted in the 19" rack. All slots in one or more racks were combined via the RACKBUS. Of course, there was also a system a pocket calculator-like configuration device called HandHeld for short.

Rosemount and Siemens were developing on similar systems, and the designers of all of these companies faced the same problem. How do you bring new devices into the system without changing the HandHeld software?

At E+H, the problem was initially solved at the parameter level. There were no menus in the system but the so-called E+H matrix, the size of which was limited to 10 x 10. The HandHeld therefore had to have information about a matrix position as to whether a parameter existed or not. The HandHeld could then have further detailed information about a possibly existing parameter, which was transmitted by the device.



If you look at the visualization of a parameter, you can see what information is needed in addition to the actual floating point number.

In the implemented systems, however, this parameter description was encoded in binary form and originally realized in some C sources that were part of the device software.

Nevertheless, these parameter descriptors have already fulfilled the same function that the DD has today. There was only one difference: the parameter descriptors were understood only by the microcomputers, and were unreadable by humans. In addition, they were not standardized.

What was added in the summer of 1989 as part of the fieldbus demonstration was the idea of formalizing the whole thing and making it understandable for people.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Device Description in the Fieldbus

Triggered by the success of the fieldbus demonstration, the so-called International Fieldbus Group (IFG) was founded at the end of 1989 and began active work to continue what had been created and bring it into standardization. In addition to working groups for pure communication technology, there was also a working group on the subject of HMI (Human Machine Interface). In this working group, E+H submitted a proposal for a parameter description language. The Rosemount developers had also solved the same problems for themselves in the past and also brought a proposal to the table.

```
PARAMETER MeasuredLevel
{
  LABEL "MEASURED LEVEL";
  UNIT LevelUnit;
  FLOAT
  {
    MAX 100.0;
    MIN   0.0;
    FORMAT "5.1f";
  }
}
```

According to E+H's suggestion (A), the description of the parameter shown in the image above would have looked something like the graphic on the left.

```
FLOAT_PARAM MeasuredLevel =
{
  "MEASURED_LEVEL",
  100.0,
    0.0,
  "5.1f",
  &LevelUnit
}
```

Rosemount's proposal (B) could have had source code similar to this.

While solution A was a new language with its own keywords, solution B was more of a variant that only applied to C code.

```
LABEL
  if(MeasurementMode == LEVEL)
  {
  "MEASURED_LEVEL";
  }
  else
  {
  "MEASURED_HEIGHT";
  }
```

Another advantage of Solution A was that Solution A provided conditional expressions.

This makes the second major difference between solutions A and B clear. The parameter description according to solution A was interpretable. This should make it possible to adapt a visualization program's behavior and displays at runtime.

At that time, the HMI working group of the IFG decided on solution A as an approach because this proposal represented the most flexible version.

In the course of 1990, the first complete specification of a device description language was created in the IFG. The document was countersigned on 11/16/1990 by the following persons:

- Craig Tielens, Rosemount,
- Jon Westbroke, Rosemount,
- Edmund Linzenkirchner, Siemens,
- Werner Stadter, Siemens,
- Walter Borst, Endress+Hauser und
- Luchiana Cinghita, Endress+Hauser.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

When the IFG and the OFC (Open Fieldbus Consortium) merged to form the IFC (International Fieldbus Consortium) shortly thereafter, work was stopped because no one knew exactly how things around the international fieldbus would develop.

Only Rosemount took up the idea of the device description and implemented it in the HART protocol. Today it can be assumed that the extraordinary spread of HART in the field of field device technology is essentially due to the availability of a DD and a suitable HandHeld (DXR 275) with a suitable DD interpreter.

Thanks to the DD, configuration programs such as PDM (Siemens) and AMS (Emerson) quickly established themselves on the market.

Special Features of the DD

- A binary coded DD can be transmitted from the field device to the visualization device.
- The DD is interpretable.
- The source code can be edited with any ASCII editor.
- The source code is easy to read and understand.
- The DD allows methods to be executed in a Java-like language. A corresponding API is defined for this purpose.
- Texts in different languages are supported.
- The DD is independent of the communication technology. It is part of the application.

HTML versus DDL

When the device description was created, there was no HTML and no Java. So we had to create our own tools and language.

Today, XML would probably be used to solve the device description problem. However, this could only have changed something in the syntax. The content would certainly be the same.

But what does HTML have to do with the DDL? This comparison was chosen to make the function of DDL comparable to something that everyone knows today. While C or PASCAL are programming languages, HTML and DDL could be defined as application languages.

A programming language usually represents a syntax definition that is processed by a compiler that creates a computer program from the textual input. As you can easily see from the example of the languages C and PASCAL, the source code of a programming language does not necessarily have to be interpretable.

This is different for application languages. In principle, application languages are used by different computers with different operating systems and must therefore be interpretable.

The history of the origin of one of the first application languages goes back to the year 1969. At that time there was neither INTERNET nor fieldbus. In 1969 "GML" (Generalized Markup Language) was developed at IBM. This language was intended to store the information about a document in such a general syntax that a document could be processed on computers with different operating systems and, above all, with different hardware such as monitors and printers. In 1978,

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

ANSI started a standardization activity that established an international standard with the abbreviation "SGML" in 1986 with ISO 8879.

While SGML not only encodes the content of the document but also defines the syntax of the source, HTML reduced SGML to such an extent that only certain syntax structures were permitted. This is the prerequisite for implementing an explorer, because SGML itself does not define any semantics. Thus, for the first time, HTML allows any document (in the form of homepages) to be exchanged over the INTERNET, regardless of which computer is used.

DDL, the HTML of the Fieldbuses

In addition to the digital transmission of measured and control values, fieldbus technology offers the basis for the remote parameterization of field devices in connection with the possibility of implementing very informative diagnostic functions in the devices. So far so good. The problem begins when - as on the INTERNET - you want to access this information and functions with a uniformly designed program; because how is the program supposed to know how the data of a device is actually encoded in the fieldbus system?

In the early 1990s, the majority of professionals believed that defining communication profiles would solve this problem. Communication profiles define exactly how individual parameters and functions are to be encoded and how they are to be transmitted via the fieldbus. Especially the pioneer HART and the experience with it brought the realization that profiles alone do not solve this problem. The devices are just too different.

Even a simple level gauge can be based on numerous physical principles: capacitance, conductivity, ultrasound, microwave, pressure, radioactivity, weight, light, laser, electrical impulse, etc.. The necessities and possibilities in the design of Functions and parameters are so diverse that comprehensive standardization is out of the question. The attempt to limit the possibilities through standardization would not only impede the further technical development of the devices, but would also make them impossible in some cases.

Here is a small example from HTML: Would you find it advantageous if you only had ten different backgrounds available for your homepage? Certainly not!

In practice, the problem, which initially went unnoticed by the standardization committees, arose much earlier than was generally known. As early as the late 1980s, Endress+Hauser developed a parameter description language that was primarily intended to provide a HandHeld terminal with information about the parameters of a transmitter. The background was the need not to have to provide a new HandHeld terminal every time a new transmitter is launched on the market.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

How the DDL Works

While in a language like HTML some types of representation such as character formats, bold type and the like are also taken into account, the DDL defines completely different things. However, the syntax of both makes it clear what the basic principle is. DDL and HTML are highly specialized application languages whose syntax is precisely tailored to the application. A DDL construct could also be written in HTML like this.

```
<script language="DDLscript">
  VARIABLE temperature
  {
    LABEL „Temperatur"
    TYPE FLOAT
    {
      DISPLAY_FORMAT „###.#"
      MIN_VALUE „-273,2"
    }
    CONSTANT_UNIT „°C"
  }
</script>
```

Although this looks a lot like a script embedded in an HTM, the syntax construct is useless because neither Internet Explorer nor Firefox would understand this syntax.

So two things are required for an application-oriented description language. On the one hand there is the syntax with the corresponding semantics that describes the application, on the other hand there is an interpreter that understands the application language and converts its content.

Here again it becomes clear:

- An application language not only lays the foundation for a programmatic implementation, but also defines the semantics (meaning) of the syntactical elements!

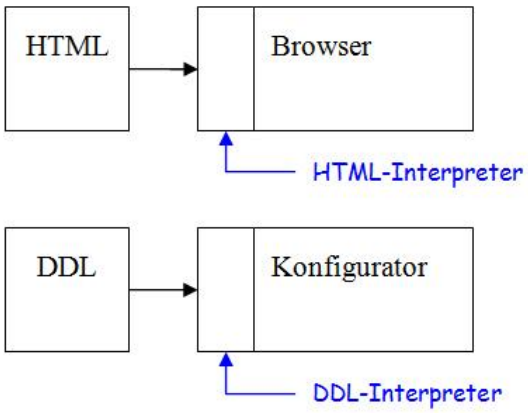
Furthermore, an application language ensures the highest conceivable degree of platform independence, but requires a special application program. Internet Explorer and Firefox show that these application programs can be implemented differently for HTML, to name just two examples.

In the case of DDL, even more glaring differences can be identified. On the one hand, there are highly complex graphic interfaces such as SIPROM (Siemens) and AMS (Fisher-Rosemount), on the other hand, there is also a very simple DXR 275 (Rosemount) HandHeld terminal, which only has a 4-line LC display. All three application programs fully support the DDL.

The example of a HandHeld terminal versus a Windows PC shows the extent to which application languages are particularly highly specialized. The DDL was designed to do just that. This means that the interfaces to the outside world must be taken into account when designing an application language. The application languages are correspondingly different.

Nevertheless, the question may arise at this point: why not create a language for everything? Why not integrate DDL into HTML to capture this application as well? The answer to this can be derived from the graphic below.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/



A web browser that - of course - works with HTML needs an interpreter that understands the entire scope of HTML syntax. Anyone who has ever seen the message "Your browser does not support frames!" while surfing the Internet clearly understands what happens when the interpreter does not have the full syntax.

Likewise, a program for configuring field devices requires an interpreter that understands DDL. If you were to try to integrate DDL into HTML, this would mean that even a simple hand-held device would become an Internet browser.

However, this is not desirable because the complexity of a DDL interpreter is already more than enough to fit in the memory of a relatively simple microcomputer.

The other possibility of hoping that one day an Internet browser would understand DDL fails with the current level of technology because, firstly, one cannot see the need for why someone should provide an explorer of this kind, and secondly it fails because of it that the definitions for the connection to the communication in HART are complete enough for an implementation.