TECHNICAL REPORT

IEC TR 61804-4

First edition
2006-12

Function blocks (FB) for process control –

Part 4: EDD interoperability guideline

Reference number
IEC/TR 61804-4:2006(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**
- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.
- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/ justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.
- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

TECHNICAL REPORT

IEC TR 61804-4

First edition
2006-12

Function blocks (FB) for process control –

Part 4: EDD interoperability guideline

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XA

For price, see current catalogue

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

———————

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL –

Part 4: EDD interoperability guideline

FOREWORD

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example, "state of the art".

Technical reports do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful by the maintenance team.

IEC 61804-4, which is a Technical Report, has been prepared by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

The text of this Technical Report is based on the following documents:

Enquiry draft	Report on voting
65C/410/DTR	65C/417/RVC

Full information on the voting for the approval of this Technical Report can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The list of all the parts of the IEC 61804 series, under the general title *Function blocks (FB) for process control*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

INTRODUCTION

This Technical Report

- contains an overview of the use of EDDL;
- provides examples demonstrating the use of the EDDL constructs;
- shows how the use cases are fulfilled; and
- shows the proper EDD application interpretation for each example.

This Technical Report is not an EDDL tutorial and is not intended to replace the EDDL specification.

Instructions are provided for the EDD application, which describe what is to be performed without prescribing the technology used in the host implementation. For example, the FILE construct describes data that is to be stored by the EDD application on behalf of the EDD. The FILE construct does not specify how the data is to be stored. The EDD application can use a database, a flat file, or any other implementation it chooses.

FUNCTION BLOCKS (FB) FOR PROCESS CONTROL –

Part 4: EDD interoperability guideline

1 Scope

This part of IEC 61804 is a guideline to support EDD interoperability.This Technical Report is intended to ensure that field device developers use the EDDL constructs consistently and that the EDD applications have the same interpretations of the EDD. It supplements the EDDL specification to promote EDDL application interoperability and improve EDD portability between EDDL applications.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60050-351, *International Electrotechnical Vocabulary (IEV) – Part 351: Automatic control*

IEC 61804-2:2006, *Function blocks (FB) for process control – Part 2: Specification of FB concept*

IEC 61804-3:2006 *Function blocks (FB) for process control – Part 3: Electronic Device Description Language (EDDL)*

ISO/IEC 15948:2004, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG) – Functional specification*

3 Terms, definitions, abbreviated terms and acronyms

3.1 Terms and definitions

For the purposes of this document, some of the terms and definitions in IEC 60050-351, as well as the following, some of which have been compiled from the referenced documents, apply.

3.1.1
EDD developer
individual or team that develops an EDD

3.1.2
EDD application
software or programmes utilizing the EDD to guide the operation of the application

NOTE These applications include configuration tools, calibrators, instrument management packages, and instrument simulators

3.1.3
end user
individual using the field device and/or the EDDL application

3.2 Abbreviated terms and acronyms

CP	Communication Profile
CPF	Communication Profile Family
EDD	Electronic Device Description. A platform and application-independent model or description of a field device written using EDDL. An EDD describes the properties, standard procedures and status associated with a device.
EDDL	Electronic Device Description Language
FF	Fieldbus Foundation; see www.fieldbus.org
HCF	HART® Communication Foundation; see www.hartcomm.org
OPC	Open connectivity; OPC Foundation; see www.opcfoundation.org
PC	Personal computer
PNO	PROFIBUS Nutzerorganisation e.V.; see www.profibus.com

4 User interface support

4.1 Overview

Most EDD applications can be characterized as either a PC application or a hand-held application. Due to the relatively small screen of a hand-held device, hand-held applications can only display a small amount of information at any given time. On the other hand, PC applications can provide a much more beneficial user interface, largely due to their larger screen size.

To support the capabilities of PC applications, the MENU construct has been extended in IEC 61804-3 compared to IEC 61804-2. Due to the differences in the user interfaces of PC applications and hand-held applications, it is expected that many devices will define two MENU hierarchies – one for hand-held applications and the other for PC applications. Some MENUs may be used in both hierarchies. Therefore, the entire hierarchy need not be specified twice.

Different menu structures for different classes of applications are possible. This guideline may be used to create menu structures in an EDD that are interpreted by applications in an unambiguous way. To promote interoperability across applications, it is highly recommended that all EDD applications follow this guideline.

4.2 Menu conventions for applications

There are existing solutions for hand-held application menus and these solutions can be used. This technical report does not provide specific conventions for hand-held applications.

4.3 Menu conventions for PC-based applications

4.3.1 Overview

EDD applications use special menus in the EDDs to show the user interface of the device. Such menus are defined for diagnostic, process variables, device features, and offline configuration. Defined identifiers are defined for the different root menus. Other submenus can be underneath these root menus.

4.3.2 Diagnostic

The diagnostic menu includes views that show the device state, detailed diagnostic information and may include graphical views that show, for example, a valve signature.

4.3.3 Process variables

The process variable menu includes views that show process measurements and set points with their quality and important information for process operators, for example, ranges.

4.3.4 Device features

The device feature menu includes features for device support. These features can be split into process-related and device-specific features. This structure is not required if the number of features is too small for splitting. Submenus that represent any structuring are allowed on the device feature menu. In case of such submenus, the menus that are underneath can be split into process-related and device-specific features.

4.3.5 Offline configuring

This menu hierarchy includes parameters, variables, and methods for offline configuration. It contains, in particular, all of the application-specific parameters of the device and may also contain important read-only and writeable variables. For more information about application-specific parameters, see 4.4. The menu can have offline methods, for example, configuration assistants.

4.3.6 Overview defined identifiers

Table 1 lists defined menu identifiers.

Table 1 – List of defined menu identifiers

Menu Identifier	Short description
diagnostic_root_menu	Diagnostic view
process_variables_root_menu	Process variable views
device_root_menu	Device feature views
offline_root_menu	Menu hierarchy for offline configuration

4.3.7 Example of EDD menu structure

This is an example of additional menus that can be added to an EDD for PC applications. These menus are additional to the existing menus for the applications. This example is HART-specific. Examples for Foundation Fieldbus[1] and PROFIBUS[2] would be very similar.

```
MENU diagnostic_root_menu
{
    LABEL "Diagnostics";
    STYLE MENU;
    ITEMS
    {
        status_window,              /* menu: style=window */
          self_test                 /* method */
    }
}

MENU status_window
{
    LABEL "Status";
    STYLE WINDOW;
    ITEMS
    {
        standard_diagnostics_page,  /* menu: style=page */
        devspec_diagnostics_page    /* menu: style=page */
    }
}
```

1 Communication Profile Family CPF 1 according to IEC 61784-1:2003, *Digital data communications for measurement and control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems.*

2 CP3/1 and CP3/2 of Communication Profile Family CPF 3 according to IEC 61784-1 (see footnote 3).

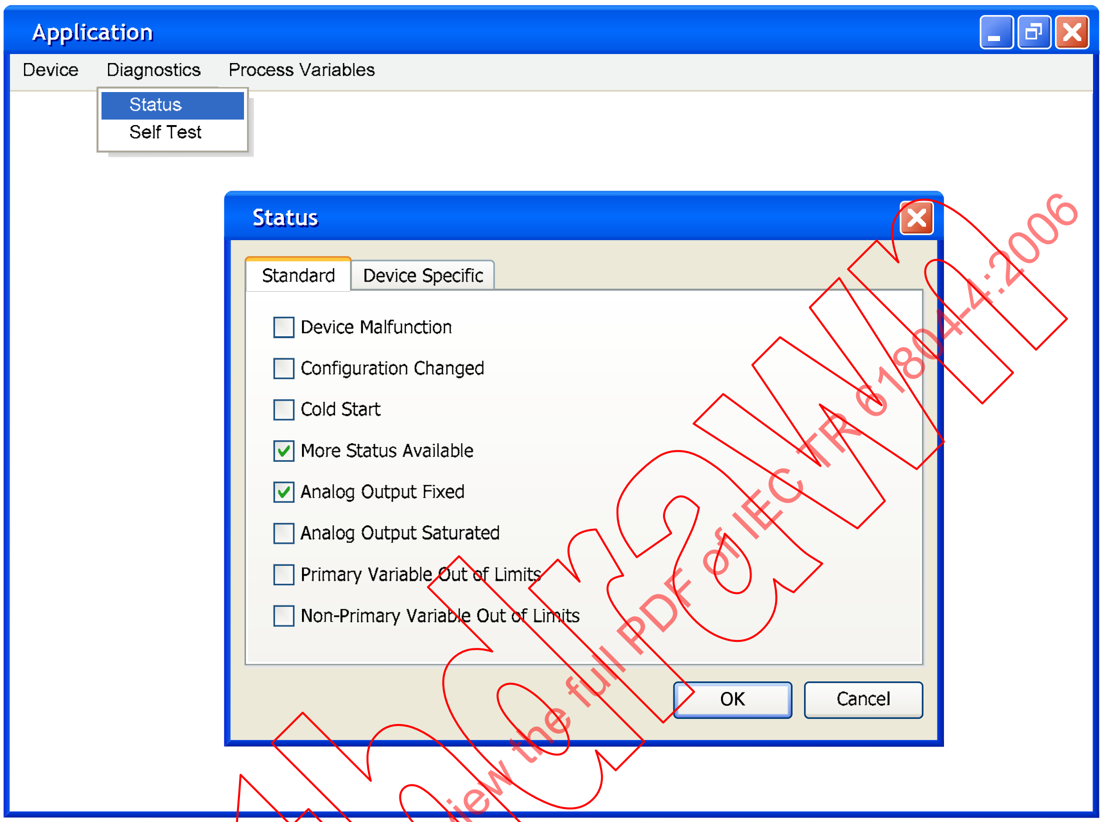
```
MENU standard_diagnostics_page
{
    LABEL "Standard";
    STYLE PAGE;
    ITEMS
    {
         device_status              /* variable */
    }
}

MENU devspec_diagnostics_page
{
    LABEL "Device Specific";
    STYLE PAGE;
    ITEMS
    {
        xmtr_specific_status_1,     /* variable */
        xmtr_specific_status_2      /* variable */
    }
}

METHOD self_test
{
    LABEL "Self Test";
    DEFINITION
    {
         /* elided */
    }
}

MENU process_variables_root_menu
{
    LABEL "Process Variables";
    STYLE MENU;
    ITEMS
    {
        overview_window,            /* menu: style=window */
        primary_vars_window         /* menu: style=window */
    }
}

MENU overview_window
{
    LABEL "Overview";
    STYLE WINDOW;
    ITEMS
    {
        process_vars_page                  /* menu: style=page */
    }
}

MENU process_vars_page
{
    LABEL "Process Variables";
    STYLE PAGE;
    ITEMS
    {
        pressure_group,                    /* menu: style=group */
        temperature_group                  /* menu: style=group */
    }
}

MENU pressure_group
{
    LABEL "Pressure";
    STYLE GROUP;
    ITEMS
    {
        pv_digital_value,           /* variable */
        pv_upper_range_value,       /* variable */
        pv_lower_range_value        /* variable */
    }
}

MENU temperature_group
{
    LABEL "Temperature";
    STYLE GROUP;
    ITEMS
    {
        sv_digital_value,           /* variable */
        sv_upper_range_value,       /* variable */
        sv_lower_range_value        /* variable */
    }
}

MENU primary_vars_window
{
    LABEL "Primary Variables";
    STYLE WINDOW;
    ITEMS
    {
        pressure_chart_page,         /* menu: style=page */
        temperature_chart_page       /* menu: style=page */
    }
}

MENU pressure_chart_page
{
    LABEL "Pressure";
    STYLE PAGE;
    ITEMS
    {
        pressure_chart               /* chart */
    }
}

CHART pressure_chart
{
    /* elided */
}

MENU temperature_chart_page
{
    LABEL "Temperature";
    STYLE PAGE;
    ITEMS
    {
        temperature_chart            /* chart */
    }
}

CHART temperature_chart
{
    /* elided */
}

MENU device_root_menu
{
    LABEL "Device";
    STYLE MENU;
    ITEMS
    {
        process_related_window,      /* menu: style=window */
        device_specific_window,      /* menu: style=window */
        master_reset                 /* method */
    }
}

MENU process_related_window
{
    LABEL "Process Related";
    STYLE WINDOW;
    ITEMS
    {
        identification_page,         /* menu: style=page */
        output_info_page             /* menu: style=page */
    }
}

MENU identification_page
{
    LABEL "Identification";
    STYLE PAGE;
    ITEMS
    {
        tag,                         /* variable */
        manufacturer,                /* variable */
        device_type,                 /* variable */
        device_revision,             /* variable */
        descriptor,                  /* variable */
        message                      /* variable */
    }
}
```

```
MENU output_info_page
{
    LABEL "Output Information";
    STYLE PAGE;
    ITEMS
    {
        range_values_group,           /* menu: style=group */
        sensor_limits_group           /* menu: style=group */
    }
}

MENU range_values_group
{
    LABEL "Range Values";
    STYLE GROUP;
    ITEMS
    {
        pv_units,                     /* variable */
        pv_urv,                       /* variable */
        pv_lrv                        /* variable */
    }
}

MENU sensor_limits_group
{
    LABEL "Sensor Limits";
    STYLE GROUP;
    ITEMS
    {
        sensor_units,                 /* variable */
        upper_sensor_limit,           /* variable */
        lower_sensor_limit            /* variable */
    }
}

MENU device_specific_window
{
    LABEL "Device Specific";
    STYLE WINDOW;
    ITEMS
    {
        identification_page,          /* menu: style=page */
        calibration_page              /* menu: style=page */
    }
}

MENU calibration_page
{
    LABEL "Calibration";
    STYLE PAGE;
    ITEMS
    {
        sensor_limits_group,          /* menu: style=group */
        sensor_trim_group             /* menu: style=group */
    }
}

MENU sensor_trim_group
{
    LABEL "Sensor Trim";
    STYLE GROUP;
    ITEMS
    {
         upper_sensor_trim_point,     /* variable */
         lower_sensor_trim_point,     /* variable */
         sensor_trim                  /* method */
    }
}

METHOD master_reset
{
    LABEL "Master Reset";
    DEFINITION
    {
        /* elided */
    }
}
```

4.3.8 User interface

4.3.8.1 Diagnostics

Figure 1 shows an example of an EDD application for diagnostics.

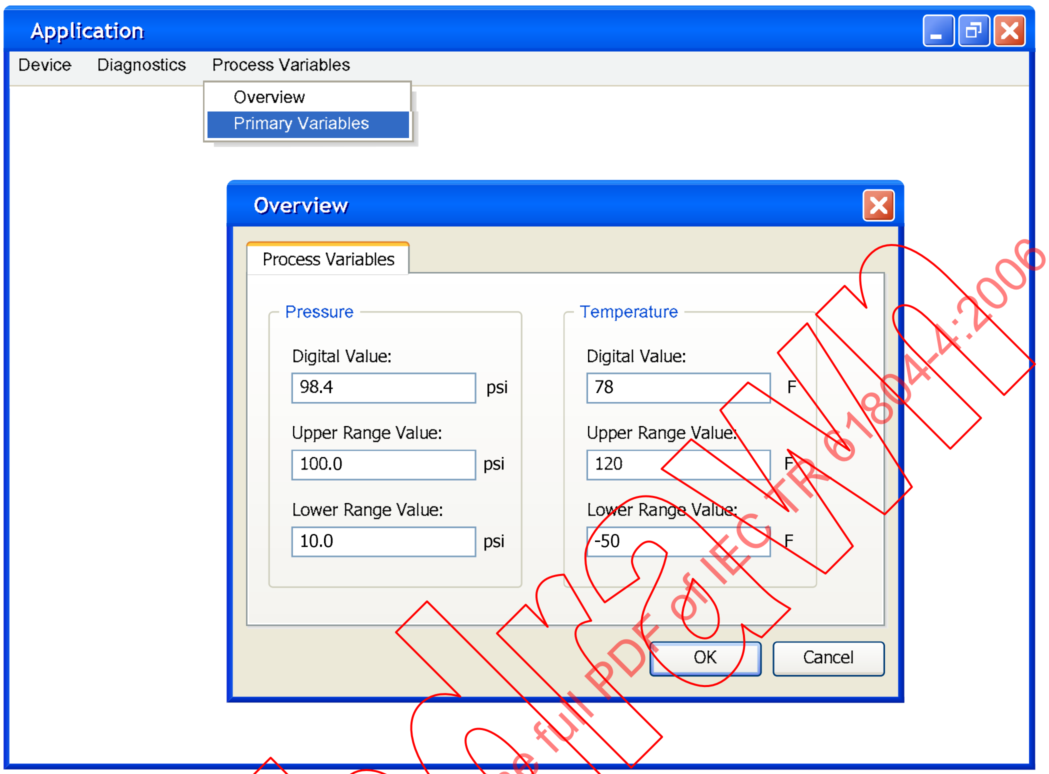


- [ ] Device Malfunction
- [ ] Configuration Changed
- [ ] Cold Start
- [✓] More Status Available
- [✓] Analog Output Fixed
- [ ] Analog Output Saturated
- [ ] Primary Variable Out of Limits
- [ ] Non-Primary Variable Out of Limits

Figure 1 – Example of an EDD application for diagnostics

4.3.8.2 Process variables

Figure 2 and Figure 3 show examples of EDD applications for process variables.



Pressure

Digital Value: 98.4 psi

Upper Range Value: 100.0 psi

Lower Range Value: 10.0 psi

Temperature

Digital Value: 78 F

Upper Range Value: 120 F

Lower Range Value: -50 F

Figure 2 – Example of an EDD application for process variables

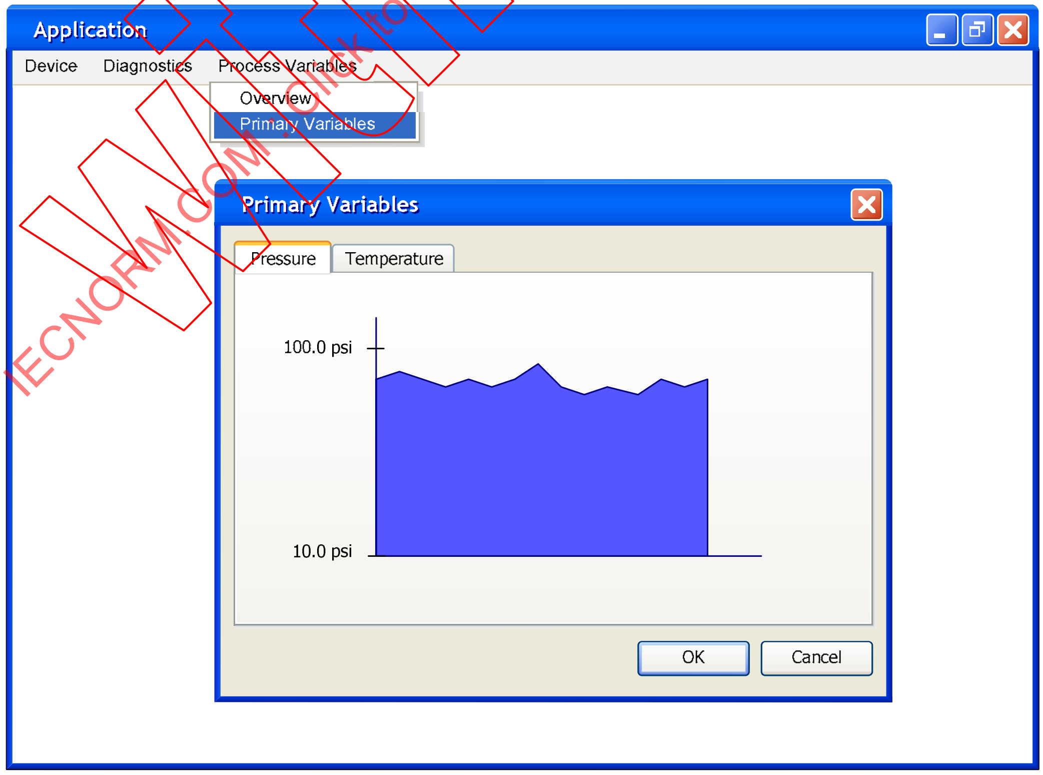
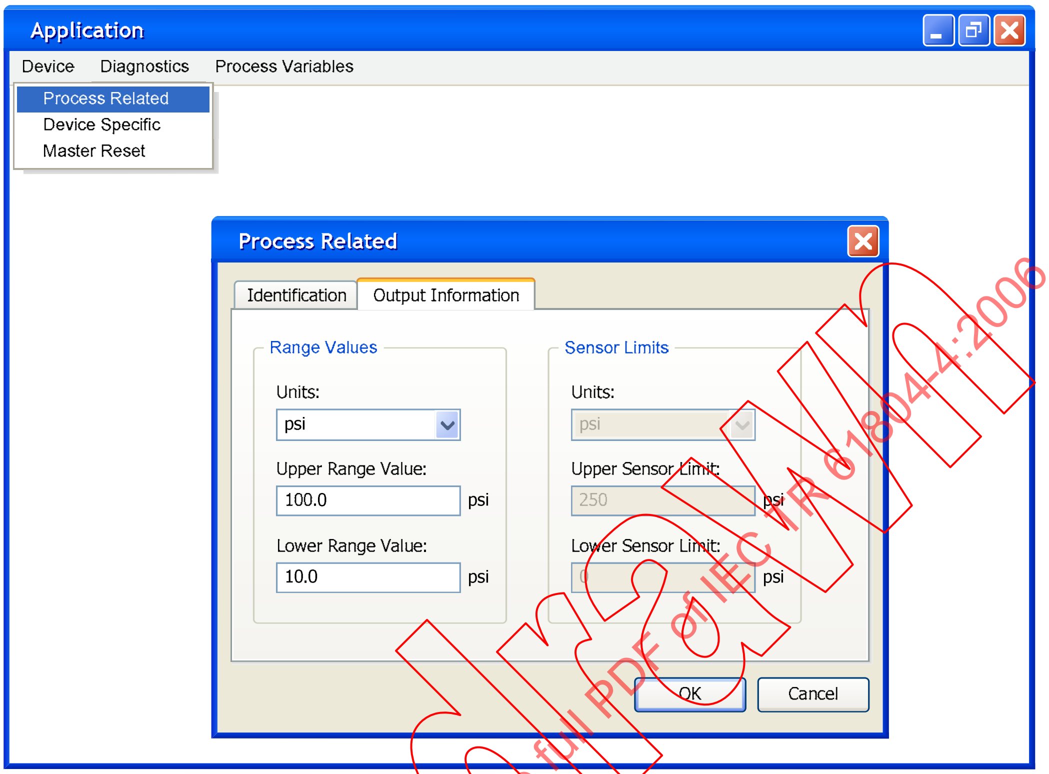


Figure 3 – Example of an EDD application for primary variables

4.3.8.3 Device features

Figure 4 and Figure 5 show examples of EDD applications for device features.



Range Values

Units: psi

Upper Range Value: 100.0 psi

Lower Range Value: 10.0 psi

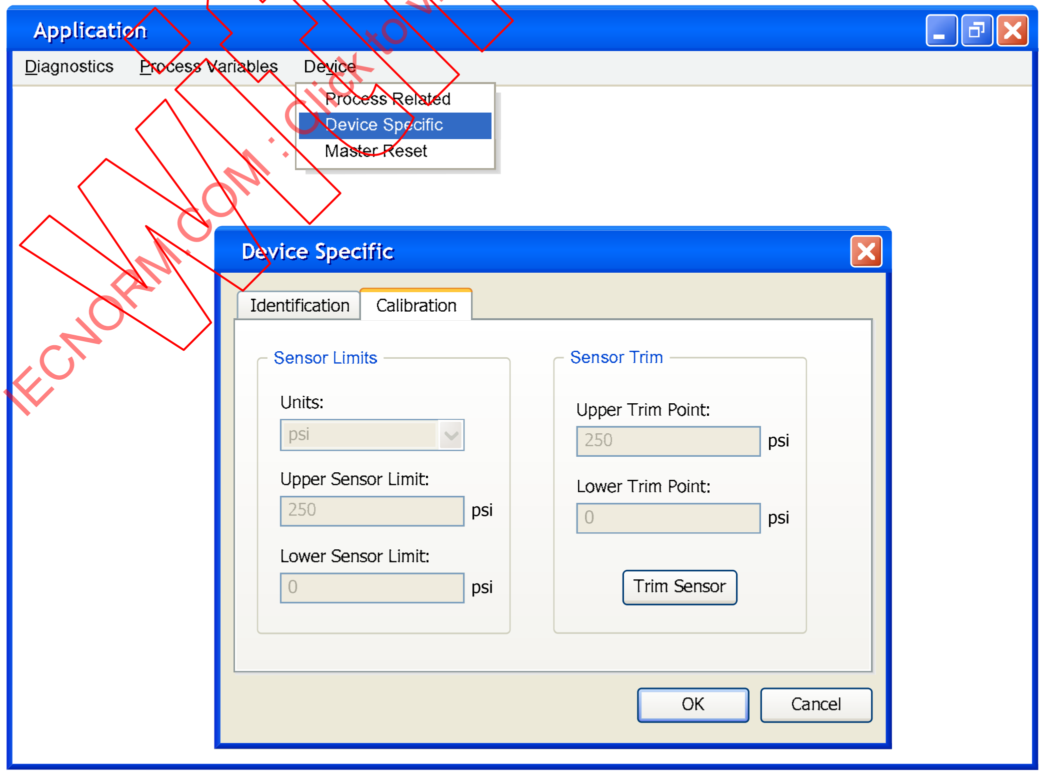
Sensor Limits

Units: psi

Upper Sensor Limit: 250 psi

Lower Sensor Limit: 0 psi

Figure 4 – Example of an EDD application for process-related device features



Sensor Limits

Units: psi

Upper Sensor Limit: 250 psi

Lower Sensor Limit: 0 psi

Sensor Trim

Upper Trim Point: 250 psi

Lower Trim Point: 0 psi

Figure 5 – Example of an EDD application for device features

4.4 Menu conventions for all applications

4.4.1 Overview

This menu is used by hand-held and PC applications.

4.4.2 Upload and download

There are two kinds of parameters in a device:

- application-specific parameters; and
- device-specific parameters.

When a device is replaced at a particular plant location, the application-specific parameters from the old device should be downloaded into the new device. However, the device-specific parameters are never downloaded because they are specific to the physical device.

For loading the application-specific parameter two menus should be used.

- The “download_variables” menu specifies the order and the total set of the application-specific parameters that should be read from the device.
- The “upload_variables” menu specifies the order and the total set of the application-specific parameters that should be loaded to the device.

Both the upload and download menus should contain all of the application-specific parameters for the device. If the device has special needs in terms of handling the data, errors, timing of the commands, and so on, the upload and download menus may contain methods in addition to, or instead of, variables.

HART and PROFIBUS supports the menu called upload_variables and download_variables. The definition of these menus for both HART and PROFIBUS is consistent with the definition of the upload and download menus described above. Therefore, these well-known names for the upload and download menus are used. If upload and download menus are not included in the EDD, then this capability, if supported by the device, is EDD application-specific.

The ordering in the off-line configuration menu (offline_root_menu) is user-aspect-orientated in contrast to the upload/download menus that are ordered from a communications point of view.

4.5 User interface extensions

4.5.1 Overview

The user interface extensions are based on a simple user interface model. The model consists of two concepts:

- containers; and
- contained items.

Containers are so named because they contain other user interface elements. Containers may include: menus, windows, dialogue boxes, tables, pages, groups, and other containers. Containers correspond to a MENU. They are distinguished from one another via the STYLE attribute of the MENU. This STYLE attribute indicates how the MENU will be displayed. EDD constructs for these user interface components are the same as a MENU except for their display.

Contained items may contain: variables, methods, edit displays, graphs, charts, images, static text and separators. The contained items correspond to various EDD constructs. For example, a variable corresponds to a VARIABLE, a method corresponds to a METHOD, an edit display corresponds to an EDIT_DISPLAY, a graph corresponds to a GRAPH and a chart corresponds to a CHART.

4.5.2 Containers

4.5.2.1 Overview

This clause describes the general layout of containers.

4.5.2.2 Menu

A menu takes the form of a drop-down menu, a pop-up menu or a navigation bar. A menu may contain windows, dialogue boxes, tables, methods, and separators.

4.5.2.3 Window

A window takes the form of a modeless window. A window may contain dialogue boxes, tables, pages, groups, methods, variables, edit displays, graphs, charts, images, static text, separators, and other windows. Windows may not contain menus.

When a container contains a window, the container should render this window as either a button or a hyperlink. The LABEL of the corresponding MENU should appear on the button or as the hyperlink text. When the button or hyperlink is pressed, the corresponding window should be opened on top of the calling dialogue or window.

4.5.2.4 Dialogue box

A dialogue box takes the form of a modal window. It may contain windows, tables, pages, groups, methods, variables, edit displays, graphs, charts, images, static text, separators, and other dialogue boxes. Dialogue boxes may not contain menus.

When a container contains a dialogue box, the dialogue box should be rendered by the container as either a button or a hyperlink. The LABEL of the corresponding MENU should appear on the button or as the hyperlink text. When the button or hyperlink is pressed, the corresponding dialogue box should be opened on top of the calling dialogue or window. If a dialogue is called from a window, no interactions are possible unless the dialogue is closed.

4.5.2.5 Table

A table takes the form of a grid. A table may contain variables, methods and other menus. The submenus build a hierarchy, which may show the device to the user in a compact way, for example, a table in which each row is a variable, method or menu.

The STYLE attribute (STYLE = TABLE) should be defined only on the root menu of a hierarchy. The EDD application also uses the style of the root for its items unless another style is defined.

4.5.2.6 Page

The pages within a container form a tabbed dialogue. A page may contain windows, dialogue boxes, tables, groups, methods, variables, edit displays, graphs, charts, images, static text, and separators. Pages may not contain menus of STYLE MENU or other pages.

A row break, a column break, and a SEPARATOR between pages will be ignored by the EDD application.

4.5.2.7 Group

A group takes the form of a group box. A group may contain windows, dialogue boxes, methods, variables, edit displays, graphs, charts, images, static text, separators, and other groups. In order to maintain a clear arrangement on the user interface, the following restricting rules apply.

- The nesting level of GROUP declarations is restricted to two. In other words, a GROUP may contain further GROUPs but these may not contain further GROUPs.
- Groups may not contain menus or pages.

Within a GROUP, usually logically related parameters and methods are grouped. The title of the GROUP indicates this relation, for example, "AnalogInput 1". Often the same string is also used in the labels of the GROUP items, for example, "AnalogInput1_StaticRevision" or "Analog Input1_Blockmode". In order to avoid this duplication of information, the EDD-editor may assemble the items of a GROUP in a COLLECTION in order to override the original labels. If the parameters are to be referred to in a GROUP the parameters can be referenced using the COLLECTION; in other cases, especially without additional semantic context, the parameters can be referenced direct.

Example of the use of a COLLECTION within a GROUP style MENU.

```
VARIABLE ai1_strev
{
       LABEL "Analog Input 1: Static Revision";
       TYPE INTEGER(4);
}

VARIABLE ai1_blomo
{
       LABEL "Analog Input 1: Block Mode";
       TYPE INTEGER(1);
}

MENU ai1_group_double                      /* Leads to double display of */
"Analog_Input1" label
{
    LABEL "Analog Input 1";
    STYLE GROUP;
    ITEMS
    {
        ai1_strev,
        ai1_blomo
    }
}

COLLECTION ai1_groupcol
{
    MEMBERS
 {
     strev,ai1_strev,"Static Revision";
     blomo,ai1_blomo,"Block Mode";
 }
}

MENU ai1_group_single                      /* Leads to single display of */ "Analog_Input1"
label
{
    LABEL "Analog Input 1";
    STYLE GROUP;
    ITEMS
    {
        ai1_groupcol.strev,
        ai1_groupcol.blomo
    }
}
```

4.5.3 Contained items

4.5.3.1 Overview

This subclause describes how a container renders contained items. All containers render contained items in the same way. Contained items, such as methods, variables and others, have a LABEL attribute. In order not to disturb the layout of the EDD application, for example, with oversized buttons for methods, the EDD developer should not use very long strings for LABELs. For this reason, some EDD applications may truncate these labels.

4.5.3.2 Methods

A method should be rendered as a button or a hyperlink. The LABEL of the corresponding METHOD should appear on the button or as the hyperlink text. When the button or hyperlink is pressed, the corresponding METHOD should be executed.

4.5.3.3 Variables

The label, the value, and, if defined, the units of the variable should be displayed in a manner consistent with the definition of the corresponding VARIABLE.

The general handling of variables is as follows.

- HANDLING = READ or menu item attribute is READ_ONLY: the value of the variable should not be editable.
- CLASS = DYNAMIC: the value should be updated continuously with current read values from the device.

4.5.3.3.1 Variable of type BIT_ENUMERATED

Multiple bits can be packaged in a single-bit enumerated variable. Each bit could have a distinct meaning. It is possible to display each bit separately and, in addition, to display the whole BIT_ENUMERATED variable.

In this case, the variable reference should be extended with a mask in square brackets and the LABEL of the variable is not shown.

EXAMPLE Displaying single bits of a BIT_ENUMERATED variable.

```
VARIABLE var1
{
        LABEL   "Var 1";
        TYPE BIT_ENUMERATED
        {
                { 1, "Bit 0"},
                { 2, "Bit 1"},
                { 4, "Bit 2"}
}
}

MENU diagnostic_info
{
        LABEL "Diagnostic";
        ITEMS
        {
                var1[2],        // Bit 2
                var1[4]         // Bit 3
        }
}
```

BIT_ENUMERATED variables are always displayed as checkboxes, independent of whether one bit or all bits are shown.

EXAMPLE The following EDD shows the differences between the full display of a BIT_ENUMERATED variable and how it could be displayed if all bits are explicitly addressed.

```
VARIABLE var1
{
        LABEL   "Var 1";
        TYPE BIT_ENUMERATED
        {
                { 1, "Bit 0"},
                { 2, "Bit 1"},
                { 4, "Bit 2"}
}
}
```

```
MENU var1_group
{
        LABEL var1.LABEL;
        STYLE GROUP;
        ITEMS
        {
                var1[1],        // Bit 0
                var1[2],        // Bit 1
                var1[4]         // Bit 2
        }
}

MENU diagnostic_info
{
        LABEL "Diagnostic";
        STYLE PAGE;
        ITEMS
        {
                var1,                   // all bit should be shown
                COLUMNBREAK,
                var1[1],                // Bit 0
                var1[2],                // Bit 1
                var1[4],                // Bit 2
                COLUMNBREAK,
                var1_group
        }
}
```

This EDD will result in an EDD application as shown in Figure 6.

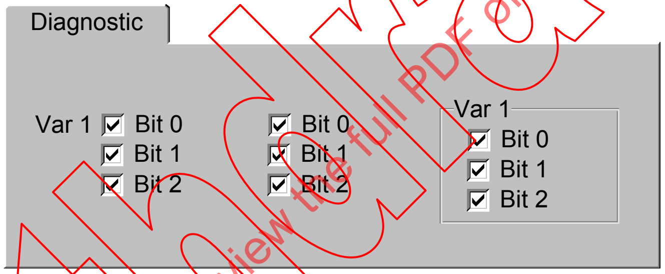


Figure 6 – Example of an EDD application for a variable of type BIT_ENUMERATED

4.5.3.3.2 Variable of type INDEX

When a VARIABLE of type INDEX is presented to the user, the text associated with the indices of the array should be displayed. If the variable is editable, it should be presented as a combo box.

4.5.3.4 Edit displays

When a container contains an edit display, the container should render the edit display as either a button or a hyperlink. The LABEL of the corresponding EDIT_DISPLAY should appear on the button or as the hyperlink text. When the button or hyperlink is pressed, the corresponding dialogue box should be opened.

4.5.3.5 Graphs

A graph should be displayed in a manner that is consistent with the definition of the corresponding GRAPH. The layout rules defined in 5.2.3 should be referred to for information on the effect of the WIDTH attribute on the layout of the GRAPH.

4.5.3.6 Charts

A chart should be displayed in a manner that is consistent with the definition of the corresponding CHART. The layout rules defined in 5.2.2 should be referred to for information on the effect of the WIDTH attribute on the layout of the CHART.

4.5.3.7 Images

The images that are referenced by the MENU should be displayed. The image width will be adapted on the dialogue, window, page or group width. With the menu item attribute INLINE the width of the image will be adapted on the width of the items in the same column. If an image is used without an INLINE attribute, the EDD application makes an implicit row break.

4.5.3.8 Static text

The text that is referenced by the MENU should be displayed. The text will be wrapped in multiple lines, if the text is longer than the width of the dialogue, window, page, group, or column.

4.5.3.9 Grid

The LABEL of the grid is shown at first and below the grid area with the width of the dialogue, window, page, group, or column and the height that is needed to show the data or what is available on the screen. Scroll bars are used to scroll the grid area, if the width or height is too small to show the complete data.

4.6 Layout rules

4.6.1 Overview

The general layout of containers is described above. This subclause describes in more detail how the contents of the container are displayed.

Each container defines the bounding box of the container. This is the area of the container where its contents may be displayed. For example, the bounding box of a window or dialogue box is the entire window except the border and title bar.

The contents of the container should be displayed starting at the upper left-hand corner of the container. The items should be displayed vertically down the left side of the container. When a separator is encountered, a new column should be created. The items following the separator should be displayed in the next column starting at the top of the page. This process continues until all items have been displayed. Column breaks should only be introduced when a separator is encountered, i.e., the EDD application should not introduce columns breaks on its own.

There is one exception to the layout rules defined in the previous paragraph. If a graph or chart has a WIDTH attribute that equals LARGE, X_LARGE, and XX_LARGE, the graph and chart should be displayed across the entire width of the container. Any items following the graph or chart should be displayed starting at the left side of the container (effectively restarting a new column beneath the graph or chart).

If static text contains carriage return and line feeds, then the text should be wrapped in lines if the text is long.

4.6.2 Layout examples

4.6.2.1 Single-column layout

The simplest layout is a list of variables, methods, edit displays, static text, graphs, and charts.

```
MENU overview_window
{
        LABEL "Overview";
        STYLE WINDOW;
        ITEMS
        {
                primary_variable,                /* variable */
      upper_range_value,                      /* variable */
      lower_range_value,                      /* variable */
      master_reset,                           /* method */
      self_test                               /* method */
        }
}
```

Figure 7 – Example of an EDD for an overview menu

The EDD in Figure 7 displays the items vertically in a window. The items are displayed starting at the left side of the screen and take up as much of the window as needed; see Figure 8.

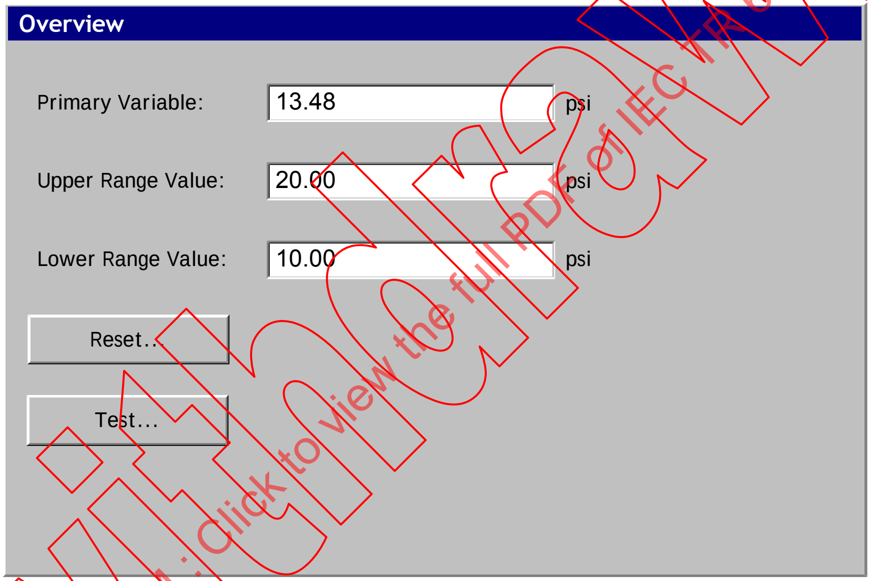


Figure 8 – Example of an EDD application for an overview window

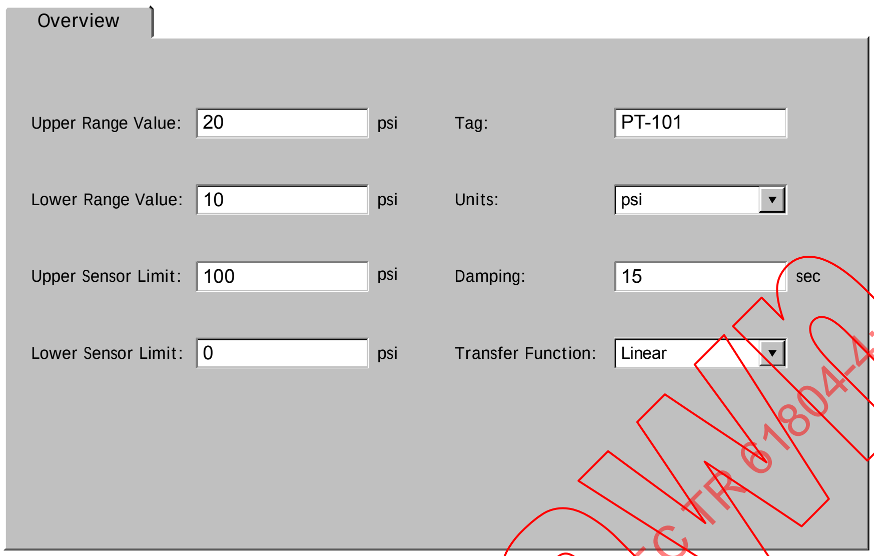
4.6.2.2 Multi-column and -row layout

Multiple columns of variables, methods, groups, images, graphs and charts may also be used.

```
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     upper_range_value,                     /* variable */
     lower_range_value,                     /* variable */
     upper_sensor_limit,             /* variable */
     lower_sensor_limit,             /* variable */
     COLUMNBREAK,                                          /* column break */
     tag,                                                  /* variable */
     units,                                                /* variable */
     damping,                                       /* variable */
     transfer_function               /* variable */
        }
}
```

Figure 9 – Example of an EDD using COLUMNBREAK

A COLUMNBREAK in Figure 9 is used to indicate a column break. All of the elements preceding the COLUMNBREAK will be placed in one column and all the elements following the COLUMNBREAK will be placed into the next column; see Figure 10.



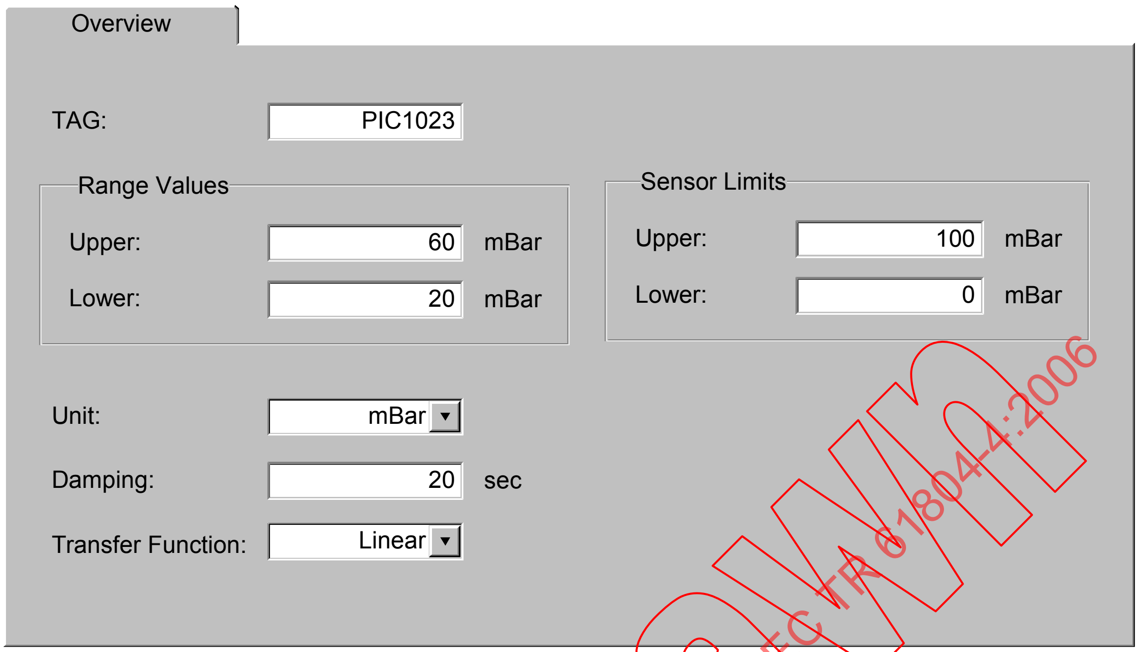
Upper Range Value: 20 psi
Lower Range Value: 10 psi
Upper Sensor Limit: 100 psi
Lower Sensor Limit: 0 psi
Tag: PT-101
Units: psi
Damping: 15 sec
Transfer Function: Linear

Figure 10 – Example of an EDD application for an overview window

A ROWBREAK in Figure 11 is used to place the following menu items beginning on the left side; see Figure 12.

```
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     tag,                                   /* variable */
     ROWBREAK,
     range_values,                          /* group */
     COLUMNBREAK,
     sensor_limits,                         /* group */
     ROWBREAK,
     units,                                 /* variable */
     damping,                               /* variable */
     transfer_function                      /* variable */
        }
}

MENU range_values
{
        LABEL "Range Values";
        STYLE GROUP;
        ITEMS
        {
     upper_range_value,                     /* variable */
     lower_range_value,                     /* variable */
        }
}


MENU sensor_limits
{
        LABEL "Sensor Limits";
        STYLE PAGE;
        ITEMS
        {
     upper_sensor_limit,            /* variable */
     lower_sensor_limit,            /* variable */
        }
}
```

Figure 11 – Example of an EDD application for an overview window



TAG: PIC1023

Range Values

Upper: 60 mBar

Lower: 20 mBar

Sensor Limits

Upper: 100 mBar

Lower: 0 mBar

Unit: mBar

Damping: 20 sec

Transfer Function: Linear

Figure 12 – Example of an EDD application for an overview window

4.6.2.3 In-line graphs and charts

Graphs and charts may appear within the columns just like variables and methods; see Figure 13 and Figure 14.

```
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     primary_variable,                /* variable */
     upper_range_value,               /* variable */
     lower_range_value,               /* variable */
     SEPARATOR,                       /* column break */
     pv_graph                         /* graph */
        }
}
```

Figure 13 – Example of an EDD for in-line graphs and charts

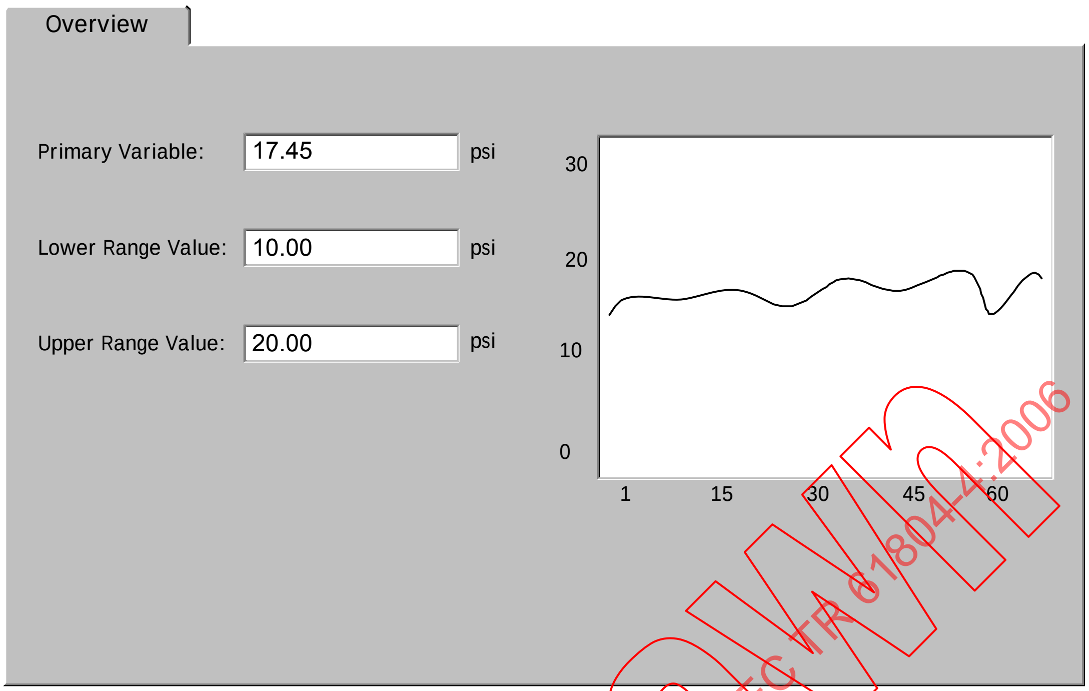


Figure 14 – Example of an EDD application for an in-line graph

If a graph and chart has with a WIDTH attribute that is equal to XX_SMALL, X_SMALL, SMALL or MEDIUM, it will be displayed within the column. Subclause 4.6.2.4 describes what happens when the WIDTH equals LARGE, X_LARGE, or XX_LARGE.

4.6.2.4 Full-width graphs and charts

Graphs and charts may also span multiple columns, see Figure 15 and Figure 16.

```
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     primary_variable,                       /* variable */
     SEPARATOR,                              /* column break */
     upper_range_value,                      /* variable */
     SEPARATOR,                              /* column break */
     lower_range_value,                      /* variable */
     pv_graph,                               /* graph */
     reload_pv_graph,                        /* variable */
     SEPARATOR,                              /* column break */
     save_pv_graph                           /* variable */
        }
}
```

Figure 15 – Example of an EDD for full-width graphs and charts

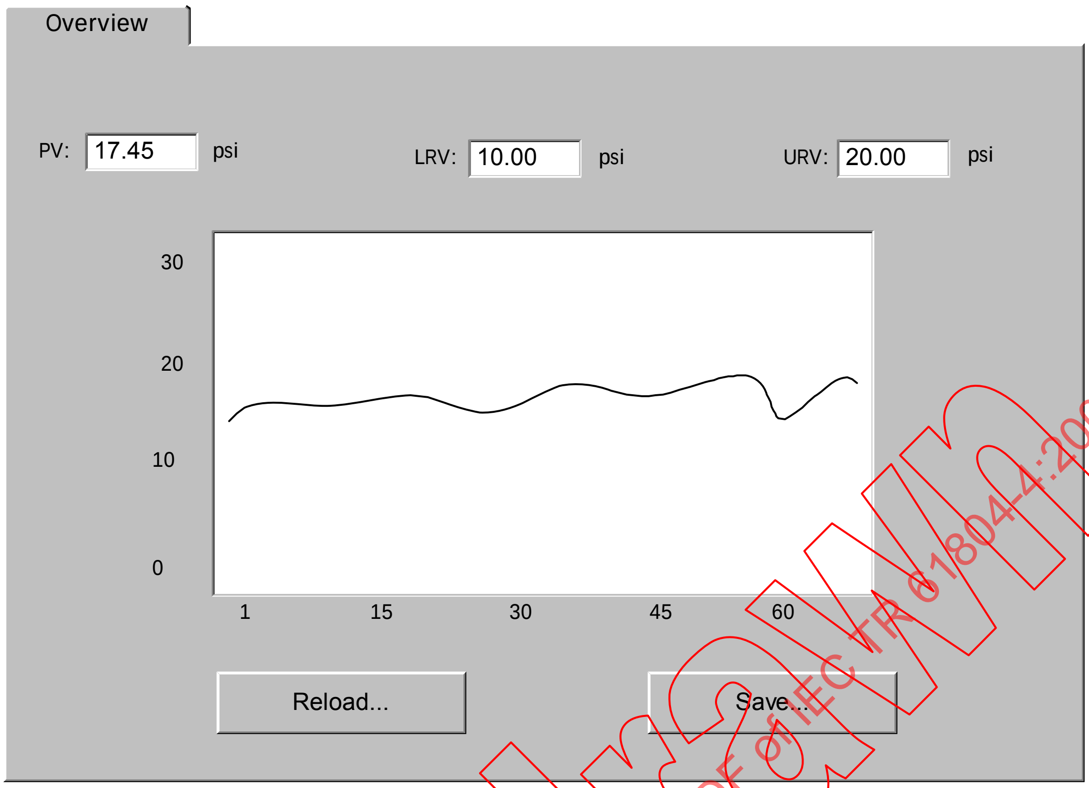


Figure 16 – Example of an EDD application for a full-width graph

If a graph or chart has a WIDTH attribute that is equal to LARGE, X_LARGE, or XX_LARGE, it will span the width of the window, dialogue, page or group. There may be additional elements before and after the graph, in which case the elements before the graph or chart are broken into one set of columns and the elements following the graph or chart are broken into another set of columns.

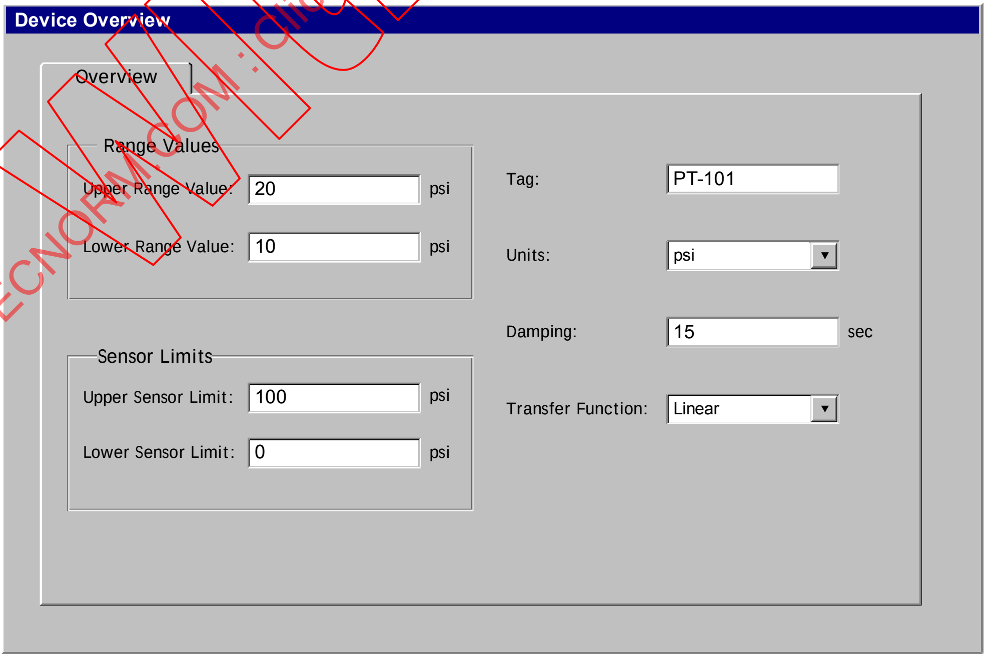
NOTE There are three columns prior to the graph and two columns following the graph.

4.6.2.5 Nested containers

Containers can be nested within other containers. Figure 17 and Figure 18 show an example of a page that is nested within a window and two groups are nested within the page.

```
MENU overview_window
{
        LABEL "Device Overview";
        STYLE WINDOW;
        ITEMS
        {
     overview_page                        /* page */
        }
}
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     range_values,                        /* group */
     sensor_limits,                       /* group */
     SEPARATOR,                           /* column break */
     tag,                                 /* variable */
     units,                               /* variable */
     damping,                             /* variable */
     transfer_function                    /* variable */
        }
}
MENU range_values
{
        LABEL "Range Values";
        STYLE GROUP;
        ITEMS
        {
     upper_range_value,                   /* variable */
     lower_range_value                    /* variable */
        }
}
MENU sensor_limits
{
        LABEL "Sensor Limits";
        STYLE GROUP;
        ITEMS
        {
     upper_sensor_limit,                  /* variable */
     lower_sensor_limit                   /* variable */
        }
}
```

Figure 17 – Example of an EDD for nested containers


Device Overview

Overview

Range Values

Upper Range Value: 20 psi

Lower Range Value: 10 psi

Sensor Limits

Upper Sensor Limit: 100 psi

Lower Sensor Limit: 0 psi

Tag: PT-101

Units: psi

Damping: 15 sec

Transfer Function: Linear

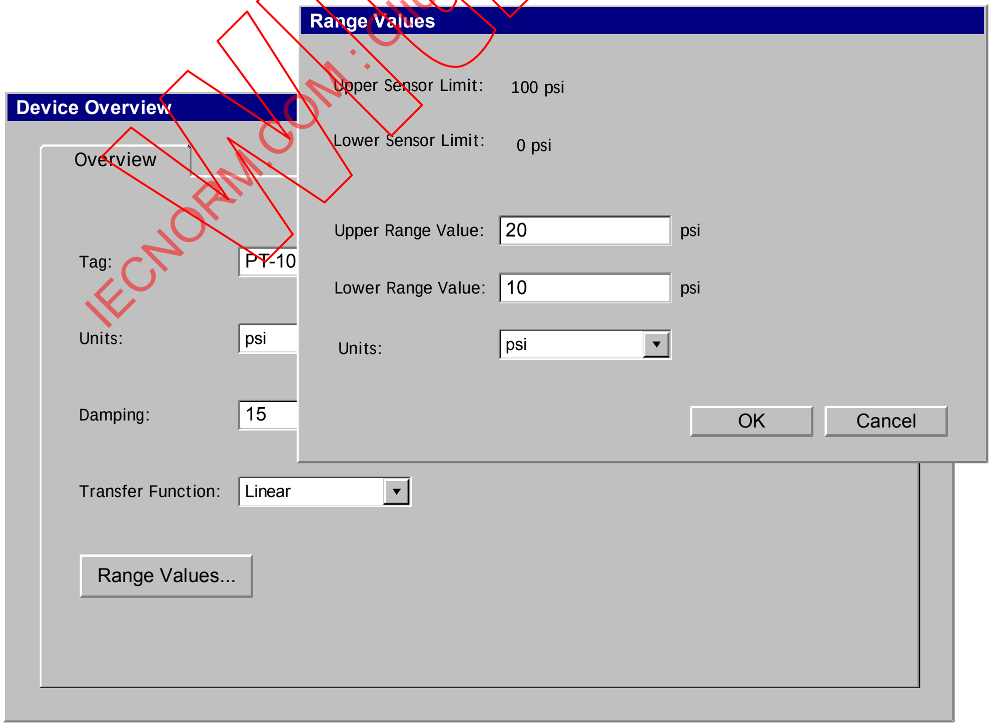
Figure 18 – Example of an EDD application for nested containers

4.6.2.6 Edit displays

EDIT_DISPLAYS can be placed in containers. When an edit display appears in a container, it is represented as a button. When the button is pressed, a dialogue box that displays the contents of the edit display appears; see Figure 19 and Figure 20.

```
MENU overview_window
{
        LABEL "Device Overview";
        STYLE WINDOW;
        ITEMS
        {
     overview_page                         /* page */
        }
}
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     tag,                                  /* variable */
     units,                                /* variable */
     damping,                              /* variable */
     transfer_function,                    /* variable */
     range_values                          /* edit display */
        }
}
EDIT_DISPLAY range_values
{
        LABEL "Range Values";
        DISPLAY_ITEMS
  {
    upper_sensor_limit,                    /* variable */
    lower_sensor_limit                     /* variable */
  }
  EDIT_ITEMS
  {
    upper_range_value,                     /* variable */
    lower_range_value,                     /* variable */
    units                                  /* variable */
  }
}
```

Figure 19 – Example of an EDD for EDIT_DISPLAYS



Upper Sensor Limit: 100 psi
Lower Sensor Limit: 0 psi
Upper Range Value: 20 psi
Lower Range Value: 10 psi
Units: psi
Tag: PT-10
Units: psi
Damping: 15
Transfer Function: Linear

Figure 20 – Example of an EDD application for EDIT_DISPLAYS

4.6.2.7 Images

Images can also be placed in containers. If an image is referenced with an INLINE qualifier, the image is displayed within the current column of the container; see Figure 21 and Figure 22. Otherwise, the image is displayed across the width of the container. The images are displayed in their original size; the EDD application should not perform any scaling on the image. Device manufacturers should, therefore, be careful about the size of the images they include in their EDD.

```
MENU overview_page
{
        LABEL "Overview";
        STYLE PAGE;
        ITEMS
        {
     spannung,          /* Reference to an image */
     anfangsverrundung,        /* variable */
     endverrundung,            /* variable */
     logo(INLINE)              /* Reference to an image which is displayed inline */
        }
}
```

Figure 21 – Example of an EDD for images

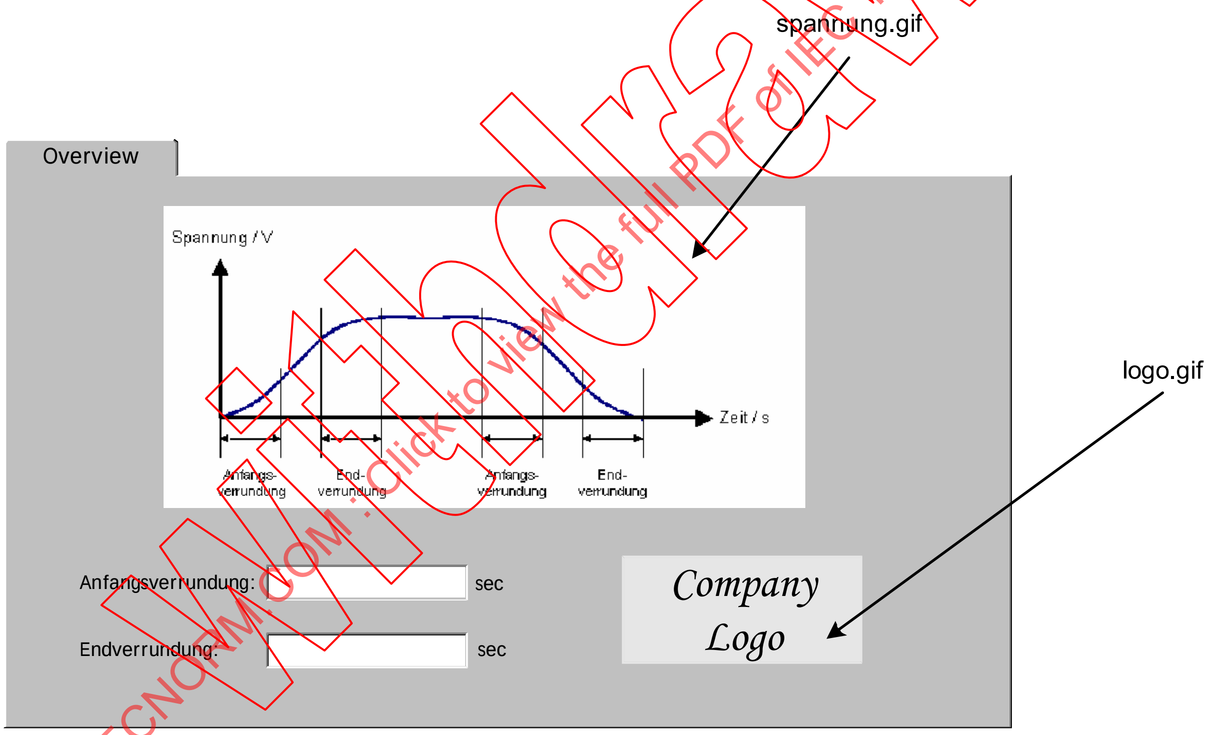


Figure 22 – Example of an EDD application for images

4.7 Default menu styles

If the menu style is not defined, then a default style may be used depending on the menu and if the menu is contained; see Table 2.

Table 2 – List of defined menu identifiers

Parent menu STYLE	Default STYLE of submenu
DIALOG, WINDOW	PAGE
PAGE	GROUP
GROUP	GROUP if the parent of the parent group is not of style group, otherwise the style is DIALOG A group may have a subgroup. Group nesting is limited to one level
TABLE	TABLE
MENU	MENU, if no variables are in the menu DIALOG, if variables are in the menu

The default style of a menu that is not contained in any menu has the style TABLE. The HART root_menu is shown as a parameter table.

5 Additional user interface elements

5.1 Overview

Smart microprocessor-based field devices continue to get more sophisticated and complex. In some cases, measurements or controllers that used to be impractical or required many pieces of equipment have been incorporated into a single device. EDDL supports these very sophisticated devices.

Of course, these constructs can be used in many other types of devices. The EDD developer has complete control over the content of the EDD and ultimately decides what EDDL constructs shall be used.

To address the needs of these complex devices, support for the following capabilities was added:

- graphing;
- charting;
- tabbed windows with optional pictorial content;
- access to EDD-application stored data.

In all cases, the additions are in keeping with the spirit of the EDDL. In other words, instructions are provided for the EDD application that describe what is to be performed independent of the technology used in the host implementation. For example, the FILE construct describes data that is to be stored by the EDD application on behalf of the EDD. The FILE construct does not specify how the data is to be stored. The EDD application can use a database, a flat file, or any other implementation it chooses.

EDDL supports two kinds of graphical data visualizations. These are GRAPH and CHART. A CHART is used to display actual values and a GRAPH is used to display stored data that may be read from the device or persistent storage. One of the common uses is to compare a waveform from the device to a reference waveform or waveform from a specific date/operation, which is stored by the EDD application. The main difference between both constructs is that the EDD application handles the data of a CHART and for a GRAPH the EDD itself defines, reads and updates the data. Methods for data handling are optional for CHARTs but they are typically needed for GRAPHs.

A GRAPH contains a list of WAVEFORMs that are plotted together. The GRAPH may be referenced from a MENU or from within a METHOD. WAVEFORMs have a minimal number of mandatory attributes in order to allow a graph to be easily produced by the EDD developer.

For those wanting more control, a wide range of optional attributes is available (display size, axis, key points, etc.).

These constructs are particularly useful for plotting valve signatures and radar signals. Both data from the field device and data stored by the EDDL application may be plotted on the same GRAPH. For example, data from the field device can be specified in one WAVEFORM and persistent data from a FILE in another WAVEFORM. The two WAVEFORMs can be plotted on the same GRAPH.

While a GRAPH is a snapshot of the device or process properties, a CHART provides a view of a time varying trend. A CHART has SOURCEs that are sampled periodically. Strip, sweep and scope style plots are supported so as to show trends over time. Horizontal, vertical, and gauge types allow graphical presentation of a single instantaneous value.

For some devices, the condition and performance cannot be determined empirically. For these devices, the relative performance is the indicator of the condition of the device. The FILE construct allows access to field device data stored by the EDDL application. The DD specifies the data, which is to be stored with a similar syntax to COLLECTION. The EDD developer defines the data to be stored in any combination of VARIABLEs, ARRAYs, COLLECTIONs, or LISTs.

The LIST construct was added to improve language flexibility when specifying persistent data stored in a FILE. The LIST is a variable length array. Data can be added to, or removed from, the list over time. Each entry in the LIST is of the same type specified by the EDD developer.

The COLLECTION construct supports any combination of member types. Previously, collections were only allowed to contain references of one type (VARIABLE, ARRAY, COLLECTION, MENU).

Clause 5 provides guidance and expectations for the use and support of EDDL constructs for graphical representations.

5.2 Graph and chart

5.2.1 Common attributes

5.2.1.1 Sizes

The HEIGHT and WIDTH attributes define the size of the CHART and GRAPH relative to the window rendered by the EDD application. HEIGHT and WIDTH do not provide absolute values for the graph window but rather provide a range of values from XX SMALL to XX LARGE. The EDD application is free to establish the actual graph window size. The EDD application should render CHARTs and GRAPHs that refer to the same HEIGHT or WIDTH value of the same proportion. The default setting for HEIGHT and WIDTH is MEDIUM (see Figure 23). The EDD application should render CHARTs and GRAPHs that have the same HEIGHT or WIDTH with the same proportion.

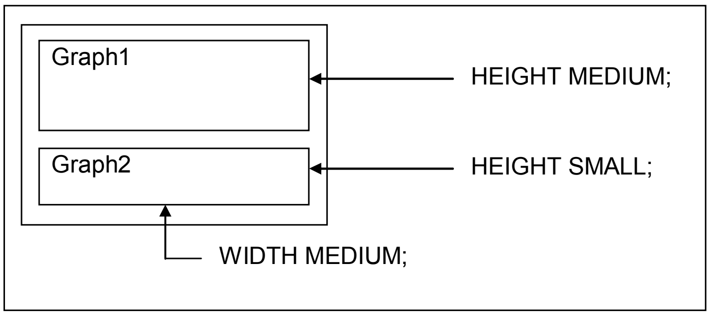


Figure 23 – HEIGHT and WIDTH attributes for CHART and GRAPH

5.2.1.2 Line characteristics

The LINE_TYPE attribute defines categories for display attributes for WAVEFORMs that are rendered on GRAPHs and SOURCEs that are rendered on CHARTs. Such display attributes may include colour and line styles (dash, dotted, etc.). The EDD application may provide user-configurable attributes for each LINE_TYPE. WAVEFORMs and SOURCEs with the same LINE_TYPE attribute should be rendered with the characteristics. Categories exist for data that are classified with DATA 1 to DATA 9, limits and TRANSPARENT, which only provides the display with the values and does not provide the connecting line.

The EMPHASIS attribute is used to differentiate between one or more SOURCEs or WAVEFORMs on a given CHART or GRAPH. By default, the WAVEFORM or SOURCE with the EMPHASIS attribute that is set to TRUE should be displayed with a larger weight (see Figure 24). The EDD application may provide user configurable attributes for EMPHASIS.

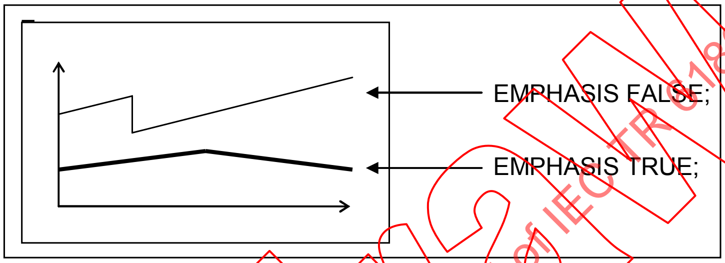


Figure 24 – EMPHASIS attribute to differentiate one or more SOURCEs or WAVEFORMs

5.2.2 CHART

5.2.2.1 Overview

A CHART is used to display continuous data values, as opposed to a GRAPH, which is used to display a dataset. The chart supports multiple data sources and multiple Y-axis. Methods can be defined for data retrieval, scrolling and zooming support.

The background colour of a CHART display is determined by the EDD application. To provide a common look and feel, it is recommended that the background colour be white. It is recommended that an EDD application support at least six curves to be displayed simultaneously.

The elements of CHART are SOURCE (this defines the source of data), visualization elements, chart type and actions.

5.2.2.2 Chart types

5.2.2.2.1 Overview

A chart can be shown in different ways. It can be shown as a horizontal or vertical bar chart, as a chart with continuously updated waveforms or as a gauge. If the type is not defined, the default is STRIP.

5.2.2.2.2 GAUGE

The actual visual element of a gauge is decided by the EDD application. A CHART with a type GAUGE can have only one data source.

5.2.2.2.3 HORIZONTAL_BAR

The actual format (visual elements) of the display is decided by the EDD application. This type dictates a horizontal bar graph type of display. This type of display can have multiple bar charts (SOURCE). A bar chart is displayed for each variable.

5.2.2.2.4 SCOPE

In SCOPE, when the source values reach the end of the display area of the CHART, the display area is erased. The new source values are once again displayed, starting at the beginning of the display area. This type of display can have multiple SOURCE definitions.

5.2.2.2.5 STRIP

In STRIP, when the source values reach the end of the display area of the CHART, the display area is scrolled. The oldest source values are then removed from the display area and the newest source values are added. This type of display can have multiple SOURCE definitions.

5.2.2.2.6 SWEEP

In SWEEP, when the source values reach the end of the display area of the CHART, the new source values are once again displayed, starting at the beginning of the display area. Unlike SCOPE, only the portion of the display area needed to display the new source values is erased. This type of display can have multiple SOURCE definitions.

5.2.2.2.7 VERTICAL_BAR

The actual format (visual elements) of the display is decided by the EDD application. This type dictates a vertical bar graph type of display. This type of display can have multiple bar charts. A bar chart is displayed for each variable.

5.2.2.3 EDDL application without full chart support

EDDL applications that do not support the graphical view of the chart should show the related variables in a standard numerical manner. For example, the data may be shown in tabular form. The format should be chosen by the EDD application developer.

5.2.2.4 Length and cycle time

The interval of time that is shown on the time axis is defined with the attribute LENGTH. The cycle time defines in ms the interval between the EDD application variable readouts. The EDD application updates the time axis with the time of the visible data. If the application cannot read as fast as defined by the cycle time it simply reads the data as fast as possible.

5.2.2.5 Data sources of a chart

A chart can display one or multiple curves. For each curve one variable is used. To support this, the CHART references one or multiple sources. The SOURCE can reference one or multiple variables.

Figure 25 shows a chart with one curve in a dialogue. The curve is refreshed in a cycle time of 1 s.

```
MENU measuring_values
{
        LABEL "Measuring Values";
        STYLE DIALOG;
        ITEMS
        {
                primary_value_view
        }
}

MENU primary_value_view
{
        LABEL "Primary Measuring Value";
        STYLE PAGE;
        ITEMS
        {
                primary_value_chart
        }
}

VARIABLE primary_value
{
        LABEL "Primary Value";
        CLASS DYNAMIC;
        TYPE FLOAT;
        CONSTANT_UNIT "Bar";
}

SOURCE primary_value_source
{
        MEMBERS
        {
                PRIM_VAL, primary_value;
        }
}

CHART primary_value_chart
{
        LABEL "Primary Value";
        MEMBERS
        {
                CHART1, primary_value_source;
        }
}
```

Figure 25 – Example of a chart with one curve in a dialogue

5.2.2.6 Legend of the curves

The EDD application may build a legend for the chart. The LABEL of the SOURCE defines this legend. If many variables are referenced in the MEMBERS of the SOURCE, then they have only one legend. If a legend is required for each curve, different sources should be defined. The LABEL of the SOURCE is an optional attribute; if the LABEL is not defined, the EDD application will show no legend for that SOURCE.

5.2.2.7 Zooming and scrolling

The EDD application can support zooming and scrolling. Therefore, the EDD requires no support. The EDD application reads the variables of a chart and stores them in an own storage. When zooming and scrolling the EDD application just shows less, more or other points of the store data in the display area.

5.2.2.8 Methods

Optional INIT_ACTIONS and/or REFRESH_ACTIONS can be defined in a SOURCE. The variables that are referenced in the SOURCE should be set in the methods. The value can be calculated by using device variables and/or local variables.

If INIT_ACTIONS are defined, the EDD application calls these methods instead of reading the variables.

If REFRESH_ACTIONS are defined, the EDD application calls these methods cyclically in the CYCLE_TIME interval instead of reading the variables.

The same method can be referenced within the INIT_ACTIONS and REFRESH_ACTIONS method lists.

Subclause 5.2.2.9 shows an example of how to use the methods in a chart. This example shows a chart in a dialogue.

5.2.2.9 Example of a chart in a dialogue

```
MENU measuring_values
{
        LABEL "Measuring Values";
        STYLE DIALOG;
        ITEMS
        {
                primary_value_view
        }
}

MENU primary_value_view
{
        LABEL "Primary Measuring Value";
        STYLE PAGE;
        ITEMS
        {
                primary_value_chart
        }
}

VARIABLE primary_value
{
        LABEL "Primary Value";
        CLASS DYNAMIC;
        TYPE FLOAT;
}

SOURCE primary_value_source
{
        LABEL "Primary";                        // this label is used in the legend
        LINE_TYPE DATA1;
        EMPHASIS TRUE;
        Y_AXIS measuring_values_axis;
        MEMBERS
        {
                PRIM_VAL, primary_value;
        }
}

METHOD CalculationMethod
{
        LABEL "";
        DEFINITION
        {
                calculated_value = primary_value * 0.12345;
        }
}

AXIS measuring_values_axis
{
        LABEL "measurement value";
        MIN_VALUE 0;
        MAX_VALUE 100;
}

VARIABLE primary_value_unit
{
        LABEL "Primary Value Unit";
        CLASS CONTAINED;
        TYPE ENUMERATED(1)
        {
                { 32,  [degC], [degC_help] },
                { 33,  [degF],    [degF_help] },
                { 35,  [Kelvin],  [Kelvin_help] }
        }
}
```

```
UNIT_RELATION
{
        primary_value_unit:
        primary_value,
        calculated_value,
        measuring_values_axis
}

SOURCE calculated_value_source
{
        LABEL "calculated";                    // this label is used in the legend
        LINE_TYPE DATA2;
        Y_AXIS measuring_values_axis;
        MEMBERS
        {
                CALC_VAL, calculated_value;
        }
        INIT_ACTIONS     {CalculationMethod}
        REFRESH_ACTIONS  {CalculationMethod}
}

CHART primary_value_chart
{
        LABEL "Primary Value";
        LENGTH 600000;
        TYPE SCOPE;
        WIDTH LARGE;
        HEIGTH SMALL;
        MEMBERS
        {
                GRAPH1, primary_value_source;
                GRAPH2, calculated_value_source;
        }
}
```

5.2.3 GRAPH

5.2.3.1 Overview

A scalable GRAPH solution is supported by the EDDL applications using waveforms and axis definitions. Multiple waveforms can be defined that are based on one or multiple y-axis and one x-axis. Methods may be used for data retrieval and for scrolling and zooming.

The background colour of a GRAPH display is defined by the EDD application. To provide a common look and feel, it is recommended that the background be white. An EDD application should support the simultaneous display of at least six curves.

A graph is made of two main elements, one is the WAVEFORM and the other is the AXIS. The WAVEFORM provides the source and type of data, emphasis to be provided, visual elements and any actions to be performed when initialized or refreshed. A single graph can have multiple WAVEFORMs. This could include data as well as the lines on a graph, which display limits and markers. There is no upper limit defined for the number of WAVEFORMs in a GRAPH. The WAVEFORMs also contain a reference to a Y_AXIS definition. When using more than one WAVEFORM, the same Y AXIS should be referenced. If the WAVEFORMs use different axis, each can have a different scaling und unit.

Figure 26 captures a graph and the visual elements, which are provided by the constructs defined in the EDDL language. Each of these elements will be shown in more detail in the relevant clauses.

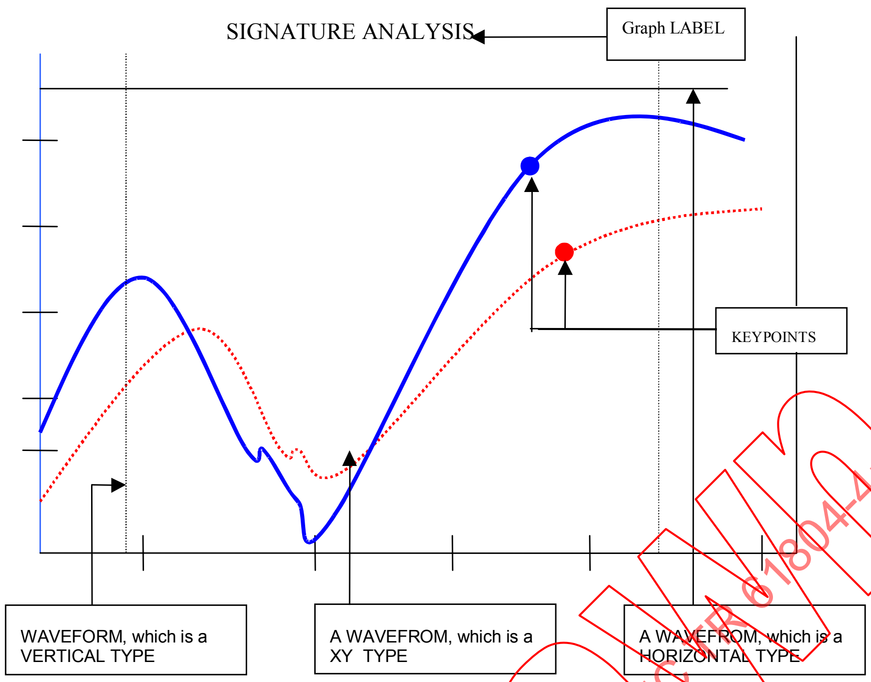

SIGNATURE ANALYSIS
Graph LABEL
KEYPOINTS
WAVEFORM, which is a VERTICAL TYPE
A WAVEFROM, which is a XY TYPE
A WAVEFROM, which is a HORIZONTAL TYPE

Figure 26 – Graph and the visual elements

5.2.3.2 Types

The GRAPH construct has WAVEFORM attributes, which define the data that can be shown on the GRAPH. The WAVEFORM can be used to plot limit values or envelopes in a display. The HORIZONTAL and VERTICAL types are used for this. The WAVEFORM construct provides the TYPE attributes that select between XY, YT, HORIZONTAL or VERTICAL.

The XY type provides a set of both X and Y point lists. The DD and device developer should ensure that the position of the “x value” in the X list is the same as the position of the corresponding “y value” in the Y list. The number of points is also defined in the construct. If no number is defined, the application sets this to the number of points in the list. This would be useful in instances where the number of points is not known.

A WAVEFORM with type YT provides a set of Y values. The initial X value and increment are defined in order to generate the subsequent X values. This would normally be used to represent waveforms, which are sampled on a periodic basis, so that x repeats at regular intervals. The number of points is defined in the waveform. In the absence of the number of points, the EDD application uses the number of y values as the number of points.

A WAVEFORM with type HORIZONTAL allows the user to plot horizontal lines on the graphical display. This would normally be used to indicate maximum or minimum bounds, or a bounding envelope. The construct has a series of Y values, with each Y value specifying a horizontal line.

A WAVEFORM with type VERTICAL allows the user to plot vertical lines on the graphical display. This would normally be used to indicate maximum or minimum bounds, or a bounding envelope. The construct has a series of X values, with each X value specifying a vertical line.

The visual characteristics of a WAVEFORM are specified by the LINE_TYPE attribute. A WAVEFORM is visualized as a series of points when the LINE_TYPE attribute is specified as TRANSPARENT.

5.2.3.3 Line display elements

HANDLING

This attribute defines the type of operations that can be performed on the WAVEFORM (read, read/write or write).

KEYPOINTS

This attribute defines certain points emphasized on the graph. The x and y values of the key point are provided in the WAVEFORM. The EDD application needs to provide emphasis on this point on the display. The determination of how the emphasis is provided is left to the EDD application.

5.2.3.4 Actions

5.2.3.4.1 INIT_ACTIONS

This attribute defines the set of actions that are to be executed before the initial display of the WAVEFORM. This is specified as references to METHOD instances, which need to be called in a particular order. If a METHOD fails or aborts for any reason then the display of the WAVEFORM is aborted. This construct can be used for the initial reading of the WAVEFORM from the device or a FILE and also any preprocessing needed (for example, averaging, or filtering of data), which is performed before the display.

5.2.3.4.2 REFRESH_ACTION

The refresh action provides a reference to the methods that are to be executed when changes are made to the X or Y axis. The methods are executed in order of definition. If a method execution fails or aborts, the remaining methods are not executed and the GRAPH reverts back to the previous display.

The method can read the data in sections if the data size is so great that it cannot be transferred within one communication transfer.

Within the method, the visible area can be calculated and set on the axis with VIEW_MIN and VIEW_MAX. All with the same axis associated waveforms will be shown with the same area in the graph.

Subclause 5.2.3.4.3 shows a simple use of a graph.

5.2.3.4.3 Example of a graph

```
VARIABLE        x_value
{
        LABEL "...";
        HELP "...";
        CLASS DYNAMIC;
        TYPE FLOAT;
        CONSTANT_UNIT [degC];
}

ARRAY           x_data
{
        LABEL "x-data";
        NUMBER_OF_ELEMENTS 1000;
        TYPE x_value;
}

VARIABLE        y_value
{
        LABEL "...";
        HELP "...";
        CLASS DYNAMIC;
        TYPE FLOAT;
        CONSTANT_UNIT [degC];
}

ARRAY           y_data
```

```
{
        LABEL "y-data";
        NUMBER_OF_ELEMENTS 1000;
        TYPE y_value;
}

COMMAND         read_data
{
        SLOT 1; INDEX 33;
        OPERATION READ;
        TRANSACTION
        {
                REPLY
                {
                        x_data, y_data,
                        device_x_min_value, device_x_max_value,
                        device_y_min_value, device_y_max_value
                }
        }
}

COMMAND         write_data_area
{
        SLOT...; INDEX...;
        OPERATION WRITE;
        TRANSACTION
        {
                REQUEST
                {
                        x_min_value, x_max_value,
                        y_min_value, y_max_value
                }
        }
}

VARIABLE        x_min_value
{
        LABEL "...";
        HELP "...";
        CLASS LOCAL;
        TYPE FLOAT;
}

WAVEFORM value_signature1
{
        TYPE            XY;
        {
                X_VALUES        { x_data }
                Y_VALUES        { y_data }
        }
        INIT_ACTIONS { refresh_signature }
}

x_max_value         LIKE x_min_value;
device_x_min_value LIKE x_min_value;
device_x_max_value LIKE x_min_value;

y_min_value         LIKE x_min_value;
y_max_value         LIKE x_min_value;
device_y_min_value LIKE x_min_value;
device_y_max_value LIKE x_min_value;

METHOD refresh_signature
{
        DEFINITION
        {
                // read data from the device depending of the current MIN_VALUE and MAX_VALUE
                // of the x axis
                x_min_value = x_axis_signature.MIN_VALUE;
                x_max_value = x_axis_signature.MAX_VALUE;
                y_min_value = y_axis_signature.MIN_VALUE;
                y_max_value = y_axis_signature.MAX_VALUE;
                write_command (write_data_area);
                read_command (read_data);
                if x_device_min_value > min_value
                        x_axis_signature.MIN_VALUE = x_device_min_value;
                if x_device_max_value < maxn_value
                        x_axis_signature.MAX_VALUE = x_device_max_value;
                if y_device_min_value > min_value
                        y_axis_signature.MIN_VALUE = y_device_min_value;
                if y_device_max_value < maxn_value
                        y_axis_signature.MAX_VALUE = y_device_max_value;
        }
}

GRAPH
{
        MEMBERS
        {
                VAL_SIG, value_signature1;
        }
}
```

5.2.3.4.4 EXIT_ACTIONS

The EXIT_ACTIONS are called if the waveform disappears from the screen.

Exit actions allow edited waveform data to be captured and manipulated as needed by the EDD.

Also if the device needs to be in any special (for example, diagnostic) mode to supply waveform data that mode can be cancelled upon closure of the graph.

5.2.3.5 Axis

The Y_AXIS definition is provided by the WAVEFORM construct whereas the X_AXIS definition is provided by the GRAPH construct itself.

Y_AXIS

This attribute provides a reference to the AXIS. When displayed, this AXIS is to be drawn with the waveform. If this is not defined, the EDD application constructs the AXIS based on maximum and minimum values and any other internal rules it may have.

X_AXIS

The AXIS construct defines how to construct the X or Y axis, which are to be displayed on a GRAPH or CHART. The AXIS has attributes which define its maximum and minimum values. In addition, there is a "SCALING" attribute, which defines whether the axis should be scaled "LINEAR" or "LOGARITHMIC". Logarithmic scaling is in base 10.

The DD developer can define a LABEL that can be displayed with the AXIS and also a string, which specifies the units in which the values are displayed on the GRAPH.

EXAMPLE The following EDD shows an example of how the application could generate a common axis for WAVEFORM w1 and w2 and a second axis for WAVEFORM w3.

```
AXIS    x1 { }
AXIS    y1 { }
AXIS    y2 { }

GRAPH graph1
{
        MEMBERS
        {
                W1, w1;
                W2, w2;
                W3, w3;
        }
        X_AXIS x1;
}

WAVEFORM w1
{
        Y_AXIS y1;
        TYPE XY
        {
                Y_VALES {w1_values}
                X_INCREMENT 1
                X_INITIAL 0
        }
}
```

```
WAVEFORM w2
{
       Y_AXIS y1;
       TYPE XY
       {
              Y_VALES {w2_values}
              X_INCREMENT 1
              X_INITIAL 0
       }
}

WAVEFORM w3
{
       Y_AXIS y2;
       TYPE XY
       {
              Y_VALES {w3_values}
              X_INCREMENT 1
              X_INITIAL 0
       }
}
```

5.2.3.6 EDDL application without graphical support

If an EDDL application does not support the graphical view of a graph then the values of the related variables can be shown in a table.

5.2.3.7 Standard usage

A GRAPH is invoked by a MENU or METHOD. A GRAPH references one or more WAVEFORMs, each representing a unique curve on the GRAPH. Each WAVEFORM describes the source of the data points. The EDD application executes the INIT_ACTIONS prior to displaying the WAVEFORM on the GRAPH. This permits calculations of the data prior to rendering. REFRESH_ACTIONS are executed by the EDD application in order to specify the visible area of the GRAPH, permitting a zoomed visualization.

5.2.3.8 Integration

A GRAPH is rendered by an EDD application through a MENU or METHOD.

A MENU may contain one or more GRAPHs. The graph can be integrated into all visible menus. It is possible to use one or more graphs together with any input and output fields within one menu. The EDD application executes the INIT_ACTIONS prior to rendering the GRAPH.

A GRAPH is rendered by a METHOD via builtins. The built-in MenuDisplay() is used to render the GRAPH. The INIT_ACTIONS of the GRAPH are called prior to displaying the menu.

5.2.3.9 Multiple waveforms and legends

The EDD application should display a legend in order to differentiate between different multiple waveforms on a single graph. The label for each waveform is derived from the LABEL attribute of the WAVEFORM. WAVEFORMS without a LABEL attribute should not be displayed on the legend.

5.2.3.10 Editable graphs

The HANDLING attribute of a WAVEFORM determines whether the user can edit the waveform. The EDD application should provide a mechanism for the user to change the waveform (direct drag and click, table entry, etc.) and a mechanism for the user to indicate that the waveform modification is complete. For GRAPHs that are generated by a MENU, the EDD application should provide a mechanism for the user to begin editing the WAVEFORM. The EDD application should update the WAVEFORMs data when the user indicates that the waveform modification is complete (for example, the save button).

The EDD application should provide a mechanism for the user to restore the original WAVEFORM without updating the data (for example, the cancel button).

5.2.3.10.1 POST_EDIT_ACTIONS on Variables

POST_EDIT_ACTIONS should be called for each change of a graphs waveform. Data inputs can be checked and any data can be modified dependent on a change.

5.2.3.11 Device-supported zooming and scrolling

The EDD application can support zooming and scrolling without support in the EDD. The EDD application shows less or more points, or other points of the WAVEFORM data on the display area. In addition to that, the REFRESH_ACTIONS in the EDD can support scrolling and zooming. The EDD application calls the REFRESH_ACTIONS if the user scrolls into an area that is not stored in the waveform data. It also calls the REFRESH_ACTIONS if the user is zooming and more points should be shown. Information about the position and zooming can be read from the axis with the VIEW_MIN and VIEW_MAX attributes.

5.2.3.12 EDD with device supported zooming and scrolling

```
VARIABLE       y_value
{
       LABEL "...";
       HELP "...";
       CLASS DYNAMIC;
       TYPE FLOAT;
       CONSTANT_UNIT [degC];
}

ARRAY          y_data
{
       NUMBER_OF_ELEMENTS 1000;
       TYPE y_value;
}

COMMAND read_data
{
       SLOT...; INDEX...;
       OPERATION READ;
       TRANSACTION
       {
              REPLY
              {
                     device_t_min_value, device_t_max_value,
                     device_y_min_value, device_y_max_value,
                     y_data
              }
       }
}

COMMAND read_current_data
{
       SLOT...; INDEX...;
       OPERATION READ;
       TRANSACTION
{
REPLY
              {
                     device_t_min_value, device_t_max_value,
                     device_y_min_value, device_y_max_value,
                     y_data
              }
       }
}

COMMAND write_data_area
{
       SLOT...; INDEX...;
       OPERATION WRITE;
       TRANSACTION
       {
              REQUEST
              {
                     t_min_value, t_max_value,
                     y_min_value, y_max_value
              }
```

```
        }
}

VARIABLE y_increment
{
        LABEL "..."; HELP "...";
        CLASS LOCAL;
        TYPE TIME;
}

VARIABLE current_date_time1
{
        LABEL "..."; HELP "...";
        CLASS LOCAL;
        TYPE DATE_TIME;
}

VARIABLE t_min_value
{
        LABEL "..."; HELP "...";
        CLASS LOCAL;
        TYPE DATE_TIME;
}

t_max_value LIKE t_min_value;
t_device_min_value LIKE t_min_value;
t_device_max_value LIKE t_min_value;

VARIABLE y_min_value
{
        LABEL "..."; HELP "...";
        CLASS LOCAL;
        TYPE FLOAT;
}

y_max_value LIKE min_value;
device_y_min_value LIKE min_value;
device_y_max_value LIKE min_value;

AXIS y_axis_signature
{
        LABLE           "...";
        HELP            "...";
        MIN_VALUE       y_min_value;
        MAX_VALUE       y_max_value;
        SCALING         LINEAR;
}

AXIS t_axis_signature
{
        LABLE           "...";
        HELP            "...";
        MIN_VALUE       t_min_value;
        MAX_VALUE       t_max_value;
        SCALING         LINEAR;
}

WAVEFORM value_signature1
{
        LABEL           "...";
        HELP            "...";
        HANDLING        READ;   // alternative READ & WRITE
        LINE_TYPE       DATA1;
        EMPHASIS        TRUE;

        TYPE            XY
        {
                X_INITIAL       current_date_time1;   // starting time in milliseconds
                X_INCREMENT     y_increment;          // time between two points in ms
                Y_VALUES        y_data;
        }

        X_AXIS          t_axis_signature;
        Y_AXIS          y_axis_signature;

        INIT_ACTIONS { init_signature}
        REFRESH_ACTIONS { refresh_signature}
}
```

```
METHOD init_signature
{
        DEFINITION
        {
                // read current data and storde area t_device_min_value,
                // t_ device_max_value,       y_ device_min_value, y_ device_max_value
                read_command (read_current_data);

                // calculation of the waveform data
                value_signature1.X_INITIAL = t_device_max_value;
                value_signature1.X_INCREMENT = (t_device_max_value-t_device_min_value)/
                                              ( y_data.NUMBER_OF_ELEMENTS - 1 );

                // set the visible area to the area from the device
                t_axis_signature.MIN_VALUE = t_device_min_value;
                t_axis_signature.MAX_VALUE = t_device_max_value;
                y_axis_signature.MIN_VALUE = y_device_min_value;
                y_axis_signature.MAX_VALUE = y_device_max_value;
        }
}

METHOD refresh_signature
{
        DEFINITION
        {
                // read data from the device depending of the current
                // MIN_VALUE and MAX_VALUE of the time axis
                t_min_value = t_axis_signature.MIN_VALUE;

                t_max_value = t_axis_signature.MAX_VALUE;
                y_min_value = y_axis_signature.MIN_VALUE;
                y_max_value = y_axis_signature.MAX_VALUE;

                // writing requested t_min_value, t_max_value, y_min_value, y_max_value
                write_command (write_data_area);

                // reading points and area    t_device_min_value, t_ device_max_value,
                //                            y_ device_min_value, y_ device_max_value from the
device
                read_command (read_data);

                // reduce the visible area to the from the device supported area
                if t_device_min_value > t_min_value
                        t_axis_signature.MIN_VALUE = t_device_min_value;
                if t_device_max_value < t_max_value
                        t_axis_signature.MAX_VALUE = t_device_max_value;
                if y_device_min_value > y_min_value
                        y_axis_signature.MIN_VALUE = y_device_min_value;
                if y_device_max_value < y_max_value
                        y_axis_signature.MAX_VALUE = y_device_max_value;
        }
}
```

5.2.4 Axis

Axes are used for charts and graphs. In a chart with curves, the x-axis is controlled by the EDD application. In a graph, only one x-axis can be referenced.

The y-axes of a chart are referenced in the SOURCE. The y-axes of a graph are referenced in the WAVEFORM. If some sources or waveforms reference to the same y-axis within one chart or graph, the EDD application should draw the y-axis only once.

The MIN_VALUE and MAX_VALUE are optional attributes. If they are not defined in the axis, the EDD application determines them. In the case of a graph, the EDD application can build the minimum and maximum x and y value of the arrays. In the case of a chart, the EDD application can read some values and build an initial minimum and maximum. The EDD application can recalculate the axes if the value goes out of range.

VIEW_MIN and VIEW_MAX are attributes that contain the current viewing area. The EDD application sets them before the INITIAL_ACTIONS are called and after the user has changed the zooming or positioning. Within a method, the VIEW_MIN and VIEW_MAX can be read and altered, for example, if the user sets the position to an area outside the available data, the method can set it to existing values.

The LABEL is used to give the axis a legend. If a CONSTANT_UNIT is defined, this unit will be used. However, if the axis is related in a UNIT_RELATION, this unit is used. Otherwise, the axis has no unit.

The SCALING of an axis can be LINEAR or LOGARITHMIC. The SCALING default is LINEAR.

The EDD application displays absolute or relative time stamps on the x-axis of a chart.

5.3 IMAGE

The IMAGE basic construct is used to display images in windows, dialogues, pages and groups. If an EDD application cannot display the image because of the required display space, the EDD application can show the LABEL instead of the image.

EXAMPLE An image definition of an EDD below.

```
IMAGE Sensor_Diagram
{
    LABEL "Sensor Diagram";
    HELP "This Diagram shows the sensor characteristic";
    PATH "|en|SenDiaEn.jpg|de|SenDiaDe.jpg";
}
```

An IMAGE definition can be used to have a graphical visualization of a method, subdialogue, or subwindow. In this case the image is referenced in the menu and in the image the method, subdialogue, or subwindow is referenced with optional attribute LINK.

EXAMPLE Usage of LINK-Attribute in a image of an EDD below.

```
MENU
{
    ITEMS
    {
        …,
        Self_Test_Image (INLINE),
        …
    }
}

IMAGE Self_Test_Image
{
    LABEL "Self Test";
    HELP "This Method execute the self test function of the device which needs some time";
    PATH "SelfTestImage.jpg";
    LINK Self_Test_Menu;
}

MENU Self_Test_Menu
{
    …
}
```

Image formats in Table 3 are allowed to be used within an image.

Table 3 – Image formats

JPG	JPEG (Joint Photographic Experts Group) File Interchange Format (JFIF) is specified in ISO/IEC 10918-1
PNG	The Portable Network Graphics format ISO/IEC 15948:2003
GIF	Graphics Interchange Format

Images for multiple locales may be specified using the same localization technique used when specifying string literals.

EXAMPLE |en|english.gif|de|german.gif

Specifies an image file to be used when the application is employing English and another when German is employed.

5.4 GRID

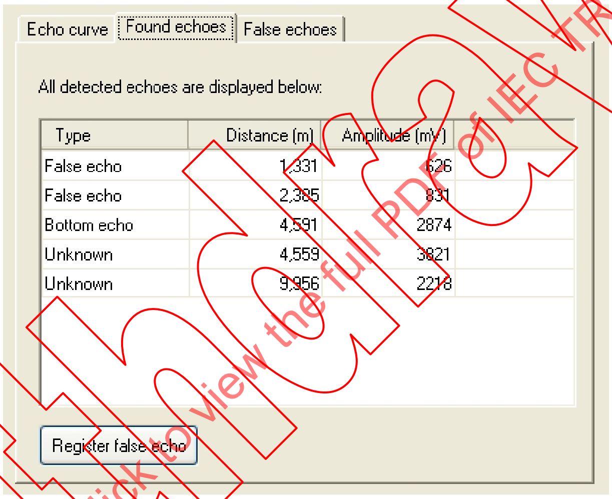
GRID describes a matrix of data from a device to be displayed by the EDD application. A GRID is used to display vectors of data along with the heading or description of the data in that vector. The vectors are displayed horizontally (rows) or vertically (columns) as specified by the ORIENTATION attribute.

EXAMPLE:

```
VARIABLE PeakType
{
        LABEL "Peak Type";
        CLASS DEVICE;
        TYPE ENUMERATED
        {
                { 1, "False Echo"},
                { 2, "Button Echo" },
                { 3, "Unkown" }
}

ARRAY arrPeakType
{
        NUMBER_OF_ELEMENTS 10;
        TYPE PeakType;
}

VARIABLE PeakDistance
{
        LABEL "Peak Distance";
        CLASS DEVICE;
        TYPE FLOAT
        {
                DEFAULT_VALUE 1.0;
        }
}

ARRAY arrPeakAmplitude
{
        LABEL "Peak Amplitudes";
        NUMBER_OF_ELEMENTS 10;
        TYPE PeakDistance;
}

VARIABLE PeakAmplitude
{
        LABEL "Device Amplitude";
        CLASS DEVICE;
        TYPE FLOAT
        {
                DEFAULT_VALUE 1.0;
}

ARRAY arrPeakAmplitude
{
        LABEL "Peak Amplitudes";
        NUMBER_OF_ELEMENTS 10;
        TYPE PeakAmplitude;
}

MENU DeviceEcho
{
        LABEL "Device Echo";
        ITEMS
        {
                EchoCurve,
                FoundEcho,
                FalseEchos
        }
}
```

```
MENU FoundEcho
{
        LABEL "Found Echo";
        STYLE PAGE;
        ITEMS
        {
                GridFoundEcho,
                RegisterFalseEchoes
        }
}

GRID GirdFoundEchoes
{
        LABEL "All detected echoes are displayed below";
   VALIDITY IF (varModelCode == 4711) { TRUE; } ELSE { FALSE; }
        VECTORS
        {
                {"Type", arrPeakType},
                {"Distance", arrPeakDistance},
                {"Amplitude", arrPeakAmplitude}
        }
}
```

Figure 27 shows a hard copy of the result of the EDD example above.



All detected echoes are displayed below:

Type	Distance (m)	Amplitude (mV)
False echo	1,331	626
False echo	2,385	831
Bottom echo	4,591	2874
Unknown	4,559	3821
Unknown	9,956	2218

Figure 27 – Result of the EDD example

6 EDDL data description

6.1 Variables

6.1.1 Variable of type BIT_ENUMERATED

Multiple bits can be packaged in a single-bit enumerated variable. Each bit could have a distinct meaning. If enforcement is required, then ENUMERATED should be used instead of BIT_ENUMERATED.

EXAMPLE Wrong usage of a BIT_ENUMERATED variable in an EDD.

```
VARIABLE Measuring_Mode
{
        LABEL " Measuring Mode";
        TYPE BIT_ENUMERATED
        {
                { 0x08, "Massflow"},
                { 0x10, "Flow" }
}
```