Working with



SIGMATEK GmbH & Co KG

**Contents**

[Preface 4](#_Toc318983837)

[First Steps 5](#_Toc318983838)

[Opening an Existing Project. 5](#_Toc318983839)

[Creating a New Project 6](#_Toc318983840)

[The Project Window 7](#_Toc318983841)

[Introduction 7](#_Toc318983842)

[Components and Meanings 7](#_Toc318983843)

[Working with the Clipboard 10](#_Toc318983844)

[Method for Placing Text 11](#_Toc318983845)

[Placing Variables 13](#_Toc318983846)

[Layers 14](#_Toc318983847)

[Global - Local Button Assignments. 15](#_Toc318983848)

[Variable List 16](#_Toc318983849)

[Variable List Components 16](#_Toc318983850)

[Variable Properties 17](#_Toc318983851)

[Indexed Text 22](#_Toc318983852)

[Procedure 22](#_Toc318983853)

[Language Management 24](#_Toc318983854)

[Button Functions 25](#_Toc318983855)

[Assigning a Button 25](#_Toc318983856)

[Available Button Functions 26](#_Toc318983857)

[Automatically Define Number Block 29](#_Toc318983858)

[Integration into a LASAL Text Project 30](#_Toc318983859)

[Step 1: Integrating a Hardware Class 30](#_Toc318983860)

[ET Terminal (ET081,ET243,ET244) 30](#_Toc318983861)

[DET Terminal (DET081, DET243, DET244, DET245) 32](#_Toc318983862)

[Step 2: Integrate ET\_RunTime Class 33](#_Toc318983863)

[Step 3: Connecting Classes 34](#_Toc318983864)

[Step 4: Integrating LASAL Text Files 35](#_Toc318983865)

[Interpreter Functions 36](#_Toc318983866)

[Control\_Led Function 36](#_Toc318983867)

[Control\_Pic Function 37](#_Toc318983868)

[Control\_Lan Function 37](#_Toc318983869)

[Control\_Inc Function 38](#_Toc318983870)

[Control\_Dec Function 38](#_Toc318983871)

[Control\_RetPic Function 38](#_Toc318983872)

[Control\_Add Function 39](#_Toc318983873)

[Control\_Sub Function 39](#_Toc318983874)

[Control\_Set Function 40](#_Toc318983875)

[Control\_Toggle Function 40](#_Toc318983876)

[Control\_EditIO Function 41](#_Toc318983877)

[Control\_VarNeg Function 41](#_Toc318983878)

[Control\_VarPos Function 42](#_Toc318983879)

[Number of Bytes in LASAL Text Classes 43](#_Toc318983880)

## Preface

LASAL Text is a mask editor for text terminals from SIGMATEK. Depending on the type of terminal used, various display sizes and number of buttons are available. With the present mask editor, variables and text for displays and input can be placed in the screen of a terminal through multi-layering.

Each individual button of the terminal can be assigned predefined functions. The button LEDs can also be controlled individually.

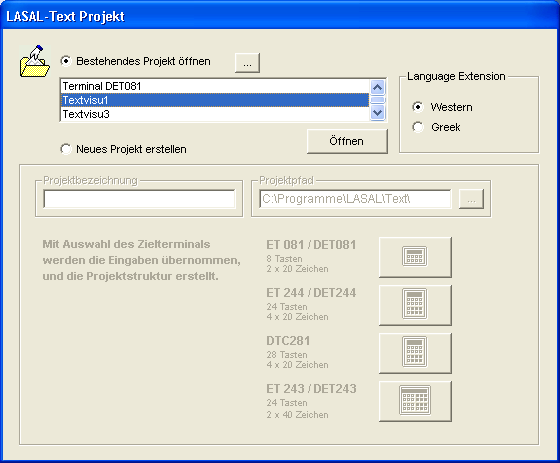
To implement a visualization project, various configurations of text and variable lists are available to choose from. The text lists can be easily integrated into a visualization project from conventional Windows applications.

To design multi-language projects, lists for translation are available and can be added to an existing project. With language conversion, existing text are replaced with text in the selected language.

## First Steps

When the LASAL Text program is opened, an existing project can be loaded or a new visualization project created.

### **Opening an Existing Project.**

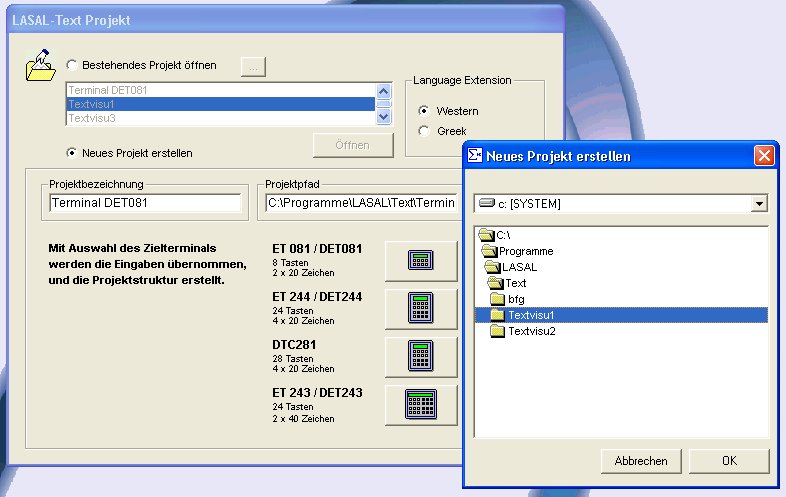


To open an existing project, select **Open existing project**.

An existing project can now be selected from a list or using the "..." button, the path of the desired project can be entered.

With help from **Language Extension**, the character set used in the project can be specified. If Greek is selected, Greek characters can then be displayed.

### **Creating a New Project**



To create a new project, select **Create new project**. Select the project path on the hard drive by pressing the "..." button and entering the project name in the project description.

E.g.: „**Terminal DET081**“

Once the entries are made, the target terminal for the new project can be selected in the next step. The input buttons for the target terminal options are used to confirm the previously made entries and create a new project.

With this step, four directories are created in the selected path for the project.

.\Pictures

.\Project

.\Tables

.\Release

All files for the current project are located in these subdirectories. This directory should not be deleted or renamed manually, since it could cause an unpredictable error in the mask editor. Once all required entries have been completed, the **project window** appears.

## The Project Window

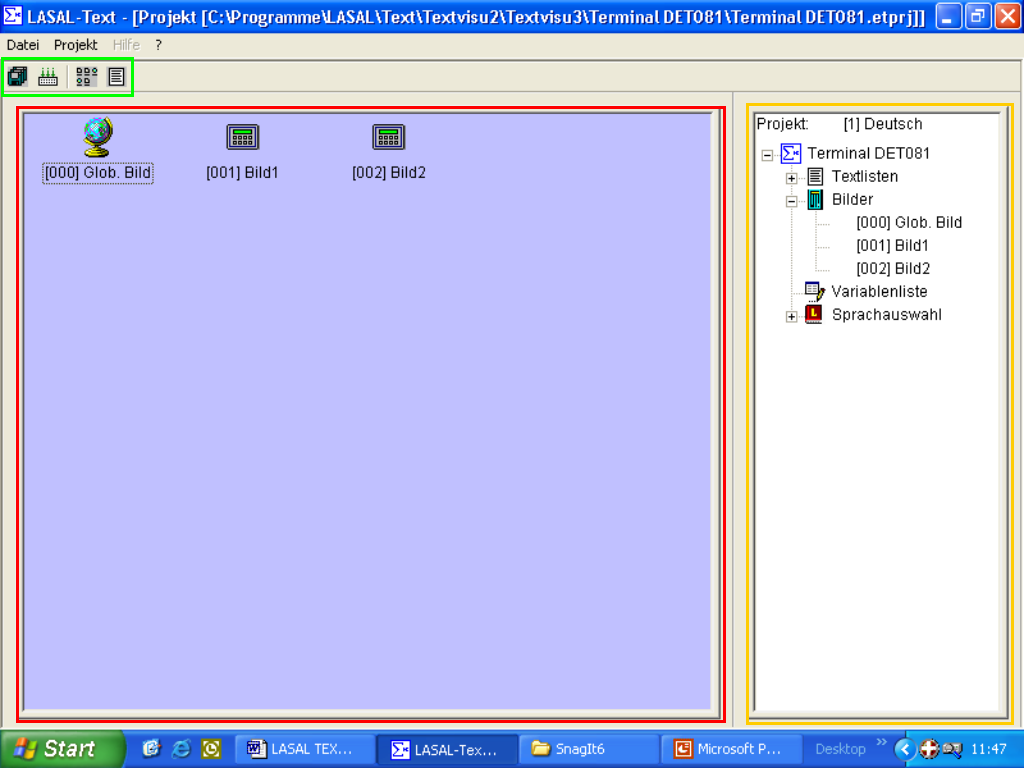
### **Introduction**

As previously explained, this window can be used to edit a new or existing project. If the steps from chapter 2 have been completed, the user interface for creating a visualization project appears.

b

a

c



### **Components and Meanings**

**a: Quick select symbols** : - save project

- compile project

- switching the screen view symbols/list

**b: User interface**: here, all images contained in the project are shown. these can be called with a mouse click and edited.

**c: Project tree**: contains all project components

**The Global Screen**

If a new project is created, the standard screen is shown in the user interface; the global screen. This screen is used to define a global button layout, which is applied in all project screens.

The remaining features are handled the same as with all other project screens.

It is also possible to create local text in the global screen. Local text is assigned the same as in all other project screens. In this case, the global screen is handled like a project screen with the screen number 0 and the name "Glob. Screen".

**The Project Tree**

The project tree mainly consists of the following components:

System text list info: To output date format that includes the day of the week, So Mo Tu We Th Fr Sa must be entered in the first line of the system text list.

**Text lists** - system text list

- Alarm text list

- Message text list

The text lists mentioned above are language-dependent(!) and displayed in the selected language.

**Screens**

The defined images are found in this entry and in the user interface. When an image is clicked with the left mouse button, it is opened in the current language.

To create a new image, press the right mouse button in the project screen or on **Images** in the project tree and select **Create new image** in the context menu.

**Variable Lists**

In these lists, variables defined in LASAL can be integrated. **Caution**, the total length of a variable cannot exceed 40 characters. The variable consists of the object name.server name combined.

**Note:**

Only server channels from LASAL can be used! Always define a variable with the object name and server name separated by a period.

Example:

**Ram.Data** (Ram = Object, Data = Server of the RAM Object Ram)

**Language Selection**

If the project should be defined in multiple languages, this entry can be used to change the current language.

**Note:**

A language change affects the display of all text in the entire project. Variables are not language-dependent.

## Working with the Clipboard

 = copy element to clipboard  = copy element from clipboard

To place text and variables in images, they must first be copied to the clipboard. In the desired image, the contents of the clipboard can be placed in the display and moved to the desired location.

If a copy process has been performed, the image appears in a new layer.

**The clipboard technology used is different from the general clipboard in the Windows platform!**

**Variables and text cannot be copied to or from other Windows**   
**applications through the LASAL Text clipboard.**

**Note:**

With the text lists, text can be moved or copied to or from other Windows applications using the keyboard combination Ctrl+C, Ctrl+V or Ctrl+X.

Further information can be found in the general Windows help.

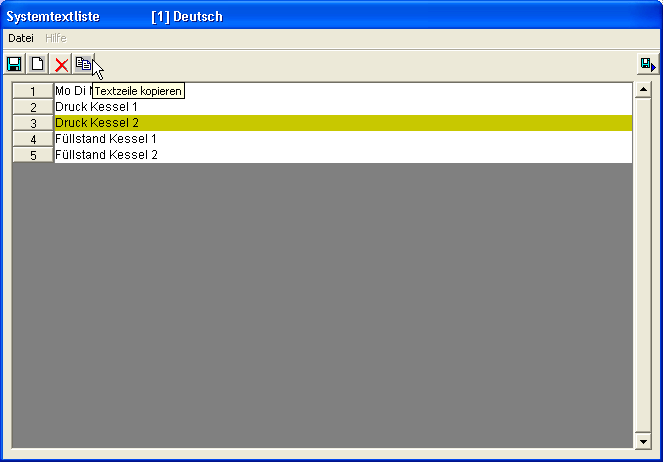
## Method for Placing Text

Open a text list by selecting a list from the Text Lists menu in the project window. In the respective image window, text can be added to the local text list and placed in the display.

If the selected list does not contain any text, a message appears with the instructions to select an available list or enter a new file name.

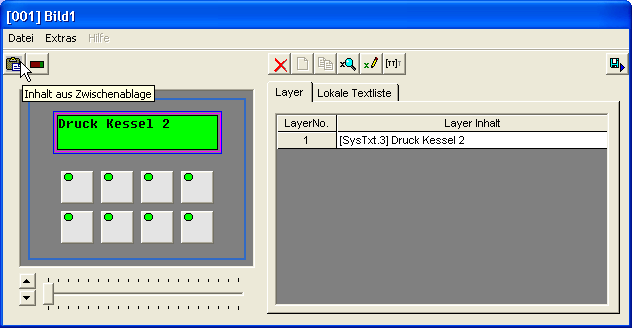
**Note:**

If a file name is selected for a text list that is not available on the hard drive, it is created and linked to the project.



Next, select an item from the text list by pressing the button with the line numbers.

The text line to copy is now identified by a change in the background color. Press the **Copy to clipboard** button in the menu bar of the text list. The current window now closes and the selected text is available for placement.



Open the image and place the copied text by pressing the **Insert from clipboard** button in the menu bar of the image.

The text is now shown with the source text list in the Layer window and can be moved to the desired position in the display.

**When no text line is selected, the first entry in the text list**  
**is copied to the clipboard.**

## Placing Variables

Functions similar to placing text.

Open the variable list by pressing the **"Variable List"** button in the project tree. Mark the previously defined variable by selecting an entry from the list on the left (variables).

The selected entry can now be copied to the clipboard by pressing the **"Copy variable"** button. The variable list now closes and the selected variable can be placed in an image.

 With these buttons, whether the variable is used for display or input of data can be specified. If input is selected, the placed variable can be edited. For this purpose however, the write protection of the respective server in LASAL must be disabled.

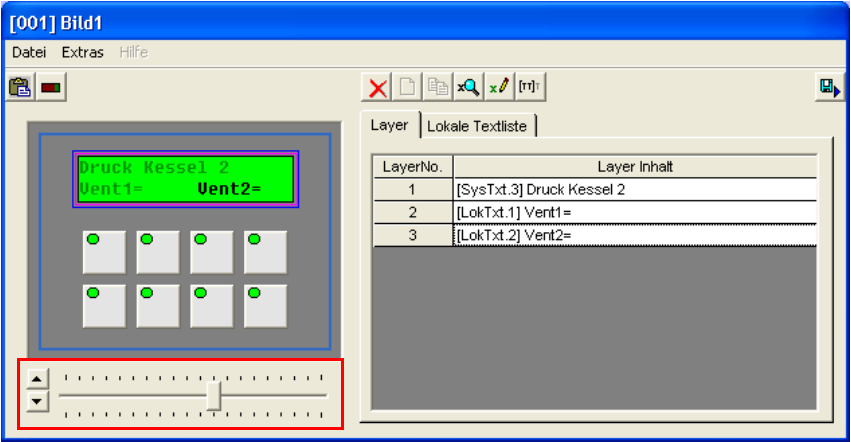
The variable can also be displayed as a bar. For this, press the button. Select a variable from the newly opened window, enter the bar width and click Place.



For more on creating and editing variables, see the chapter **Variable Lists**.

## Layers

LASAL uses layering to display text and variables. The layers can be imagined as transparent sheets stacked on top of one another, which can be moved individually.



Select the desired layer by clicking in the list. In the display, all layers except for the selected one are shown in a light color. With the **navigation help**, the layer can be moved vertically and horizontally.

**Layers can only be created in the LASAL Text clipboard!**

## Global - Local Button Assignments

When creating a new project, a screen is already predefined - the global screen. All further screens created later are called project screens.

In the global screen, button definitions can be defined that apply for the entire project. For all other screens, button definitions are local.

**Local button definitions are a higher priority than global button definitions.**

This means - if no local buttons are defined, definitions from the global screen are used. As soon as a button is placed in a project screen, it has priority over the globally defined button!

**Example**

Create a screen project in which each screen uses the numbers 0 - 9 (buttons 0 - 9) to enter values. The buttons in the project screens however, should perform a special function.

**Procedure**

The buttons 0 - 9 (ET 243 / ET 244 / DTC 281) and C, E should be valid in every screen. The keypad marked in white should have its own function depending on the screen in which it is placed.

When allocating the buttons, select "alphanumeric" from the left list. In the lower frame of the window, the option to define a number or character string for the button is given.

Once the global buttons have been assigned - the numbers 0 - 9 can be applied throughout the entire project. As soon as one of these buttons is defined in a project screen, it has priority!

## Variable List

In the variable list, the variables required for display and input can be created and edited. To open the variable list, click on the entry **"Variable list"** in the project window.

**Only variables the exist in a LASAL projects**  
**can be displayed or edited**

To use a number value in the display, a server channel must be defined in LASAL.

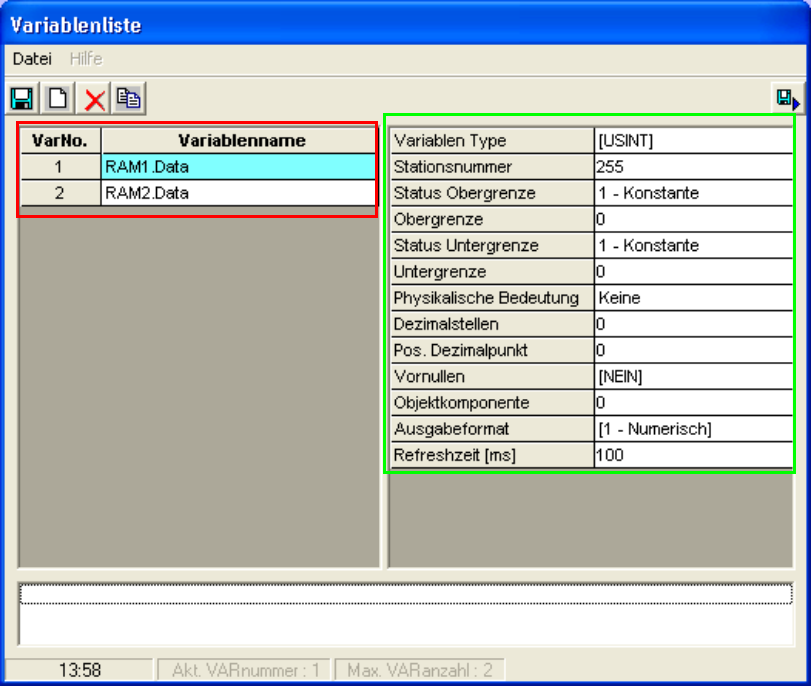
Definition: **<Objectname>.<Servername>**

Example: **Ram0. Data**

### **Variable List Components**

b

a



A variable consists mainly of the **variable name (a)** and the corresponding **properties (b)**. The name of a variable must correspond to the label in LASAL.

### **Variable Properties**

The properties of a variable specify the origin and appearance when displayed.

**Variable Type**

The variable type describes the maximum size of the value that can be entered.

|  |  |  |
| --- | --- | --- |
| USINT | Unsigned Short Integer | **0 ... 255** |
| SINT | Signed Short Integer | **-128 ... 127** |
| UINT | Unsigned Integer | **0 ... 65536** |
| INT | Signed Integer | **-32768 ... 32767** |
| UDINT | Unsigned Double Integer | **0 ... 4294967295** |
| DINT | Signed Double Integer | **-2147483648 ... 2147483647** |
| BOOL | Boolean Data Type | **True / False (1/0)** |
| TIME | see output format |  |
| DATE | see output format |  |
| TEXT | general text variable |  |
| OBJECT | object variable |  |
| STRUCTURE | structure variable |  |

**Station Number**

The station number assignment describes the origin of the variable in the CAN network or from a local environment.

Possible range CAN: station number **0 to 31**

Local origin: station number **255**

**Upper Status Limit**

With the available data range, each variable can be assigned maximum and minimum value. This option limits the input range for a variable.

**Selection Options**

0 Maximum limited by the maximum data length of the selected

data type.

1 Constant upper limit fixed upper limit

2 Variable upper limit defined by another variable

**Upper Limit**

Numerical value of the upper limit. Negative values can also be entered.

**Example:**

A variable with the value range from -10000 to -100

Upper limit: -100

Lower limit: -10000

**Note:**

If the upper limit status was selected for the variable, the numerical value of the variable number defines the upper limit.

**Lower Limit Status**

See upper limit status

**Lower Limit**

Numerical value of the lower limit

**Note:**

If the lower limit status was selected for the variable, the numerical value of the variable number defines the lower limit.

**Physical Meaning**

Not yet available In this software version (V1.12)!

In the future: e.g. pressure, distance, speed, ...

**Decimal Places**

The number of decimal places in the output including after the decimal point.

**Example:**

5 decimal places

2 Pos.Decimalpoint

Output: **000.00**

**Position of Decimal Point**

Number of post decimal places.

With 0, the numerical value is displayed as a whole number expression.

**Example:**

6 decimal places

0 Pos.Decimalpoint

Output: **000000**

**Leading Zeros**

Specify whether leading zeros should be displayed.

**Example:**

YES 0000345.67

NO 345.67

**Object Components**

As previously explained, the variables must be defined as LASAL server channels. Thereby, the object name must be followed by the server name.

**Example:** Ram1.Data

A different method of definition can be performed through the object components.

**Example:** 6 Ram1 Definition as[Object]

7 Data1 Object components = 6 (based on variable 6)

8 Data2 Object components = 6 (based on variable 6)

9 ....

In this example, Data1 and Data2 are server channels of the object Ram1.

**Output Format**

The output format defines the appearance of the variable in the display.

0 - None no output format for hiding a placed variable.

1 - Numeric general numeric output format defined by

decimal places and decimal point position.

2 - HH:MM time format HH 2-digit hour value, MM 2-digit minute value

Example: **09:32**

3 - HH:MM:SStime format with 2-digit seconds display.

Example: **09:32:12**

4 - DD.MM date format DD 2-digit day value, MM 2-digit month value.

Example: **26.05**

5 - DD.MM.YY date format with 2-digit year value.

Example: **26.05.01**

6 - DD.MM.YYYY date format with 4-digit year value.

Example: **26.05.2001**

7 - WD:DD.MM **note:** enter week day in the system text list. date format with 2-place weekday value

In the first line, enter: SoMoTuWeThFrSa

Example: **Mo:26.05**

8 - WD:DD.MM.YYdate format with weekday and 2-digit year value.

Example: **Mo:26.05.01**

9 - WD:DD.MM.YYYY date format with weekday and 4-digit year value.

Example: **Mo.26.05.2001**

10 - Text general text format for variable.

Example: **Variablentext1**

11 - Message text through the numeric value of the variable, a text line can be

output from the message text list.

Example: Ram1.Data = 0 output: **Text1**

Ram1.Data = 1 output: **Text2**

....

(see indexed text)

12 - Alarmtext see indexed text

13 - Systemtext see indexed text

14 - Codeformat numeric code entry and code format display.

Example: button input 1234

display \* \* \* \*

**Refresh Time**

Specified time for cyclic update of a variable in milliseconds [ms].

Note: Refresh time > 0 ms whole number

## Indexed Text

Through the output formats **Message text**, **Alarm text** and **System text** in the variable list, text lines based on a variable value can be shown on the display.

Selecting these output formats opens a view of the corresponding list.

With the selection of a text entry from this list, the start line is defined with the variable value 0.

**Example:**

Motor.Data = 0 output **OFF**

= 1 **RUN**

= 2 **STANDBY**

= ...

**Note:**

The application of this output format is language-dependent.

**Tip: It should be noted that the same number of text lines are defined in each language.**

### **Procedure**

Create a message text list and enter several lines of text.

**Example:**

2. ...OFF

4. RUN

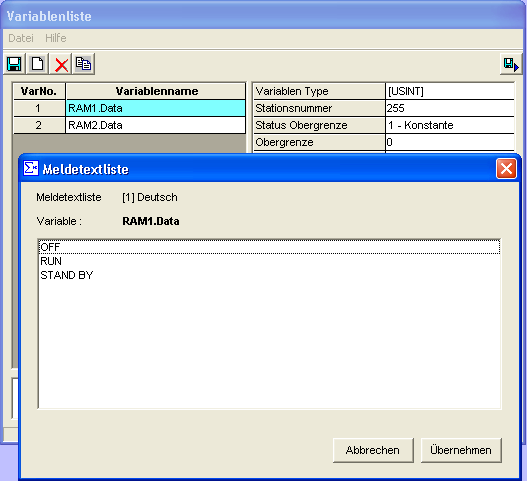
5. STANDBY

Next, open the variable list and assign a variable. Select an unsigned numeric data type, e.g. USINT. In this example, the upper limit of the variable should be set to 2 and the lower limit to 0. The decimal places and position of the decimal point are not required here.

In the output format, the entry **11-message text** is found. If this entry is selected, a view of the message text appears in the current enabled language.

Click on the line with the text entry **OFF**. This text entry is represented in the display with the variable value 0.

Confirm this entry by pressing the **Apply** button.



Entries from the alarm and system text lists can also be indexed.

**Note:**

If changes are made in the message text list, these are also updated in an indexed variable.

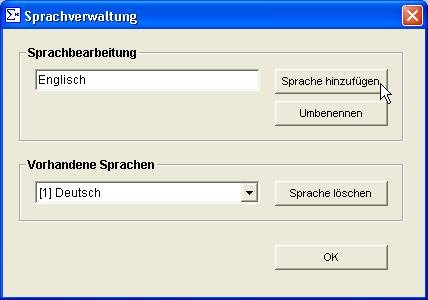
## Language Management

The text in a visualization project can defined with multiple languages. When creating a new project, German is preset as the default language.

To edit the language of a project, click **Project / Language management** in the main window of the mask editor. The language conversion window now opens, in which languages can be added, renamed or deleted.

To add a new language to the project, enter the desired language name under language editing and confirm with the **Add Language** button.

To rename a language, select the one form the list of available languages, enter the new language name and confirm by pressing **Rename**.



**A language can only be defined once in a project!**

**In a visualization project, at least one language must be defined!**

When deleting a language, it should be noted that the existing file connections to the text lists are lost.

## Button Functions

Each button in a terminal can be assigned with several functions. Up to a maximum of 100 functions per button is possible.

### **Assigning a Button**

In the image editing window, left-click on the desired button. In the window that now opens, the selection of available button functions can be seen on the left. Mark the desired function and confirm with the button.



### **Available Button Functions**

**Change Screen**

Open a screen with the specified name.

**Return**

Returns to the last screen opened.

**Language**

Changes the active language.

**Function**

Calls a user function through the function number with the input of an additional parameter [DINT]. In the ET\_User class, a user-defined function can be called in the ET\_USER:*FCAWL* method.

no function number

para additional input parameter

**Inc**

Increment (+1) a variable; whereby the upper limit of the variable cannot be exceeded.

**Dec**

Decrement (-1) a variable; whereby the lower limit of cannot be exceeded.

**Add**

Add a constant value to a variable.

**Sub**

Subtract a constant value from a variable.

**Set**

Set a variable with a constant value.

**Beep**

Not available in this software version.

**Toggle**

Change the value of a variable between 0 and 1 or 1 and 0.

**Note:**

If the variable has a value greater than 1, it is overwritten with 0.

**LED**

Sets the status of a button LED on the terminal with the specified LED number starting with 0.

0 = Off

1 = On

2 = Blink

**Note:**

The LED status is global and remains the same when the screen is changed.

**Enter**

Confirms a number entry in the display.

**Note:**

The change mode is ended with this button assignment.

**Cancel**

Cancels a number entry in the display, whereby the old number value is reassigned to the variable.

**Edit Next**

Change mode for variables, whereby the first variable is the start point and the cursor can move to the next in the positive direction.

**Edit Previous**

Opposite of the **Edit Next** function.

Starting from the last variable placed, variables can be selected for editing (value input).

**Alphanumeric**

Input numbers and character strings. With numeric variables, the first number of the entered sequence is used.

**Example:**

Number-only allocation character string = **6**

Application for text and numeric variables character string = **ABCabc1**

or character string = **1ABCabc**

**Cursor left**

Move the input cursor in a text variable one input space to the left.

**Cursor right**

Move the input cursor in a text variable one input space to the right.

**Delete Character**

Deletes the character under the input cursor.

### **Automatically Define Number Block**

The number block of a terminal can be predefined using the menu item [Extras/assign number block] in one step. Existing defined functions for these buttons are retained. In the function list for these buttons, the required assignments are added on the end.

The CANCEL and ENTER buttons are also defined.

## Integration into a LASAL Text Project

To create an operational visualization project, the LASAL Text project must be translated into text lists and integrated into a new or existing LASAL project. This process is explained in the following four steps.

### **Step 1: Integrating a Hardware Class**

Open the LASAL library and import the required hardware class.

**Caution:**

It should be noted that only one object of the terminal exists. This means that if a terminal is replaced with another, the object to replace must be removed from the **project workspace** and not just the **project network**.

Possible selection of hardware classes: (found in the hardware library under Sigmatek => Hardware => Display => Text terminal

ET081

ET244

ET243

DET081

DET244

DET243

DTC281

After adding the required hardware class, it can then be found under **User Classes**.

### **ET Terminal** (ET081,ET243,ET244)

**Hardware Configuration**

An ET terminal is connected to the PLC over a CAN connection. When connecting the terminal, ensure that the CAN station number, baud rate and TX/RX objects on the terminal are initialized correctly.

**Note:**

To complete the settings on an ET terminal, the terminal-specific button combination must be pushed when the supply voltage is applied. The respective button combination can be found in the hardware description provided with the terminal used.

The terminal should be initialized with the following settings.

Mode **CAN object**

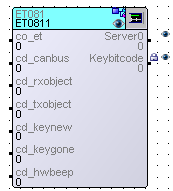
Data transfer **1 [500 kBaud]**

TX Object **e.g. 1001**

RX Object **e.g. 1002**

**Hardware Configuration in LASAL**

The following settings must be made in LASAL for the hardware class object. For error-free communication, the settings must be the same as it the terminal.



**Clients:**

|  |  |
| --- | --- |
| **co\_et** | Establish connection to ET\_RUNTIME object |
| **cd\_can** | Number of the CAN system used. |
| **cd\_rxobj** | CAN object number (RX object in terminal) |
| **cd\_txobj** | CAN object number (TX object in the terminal) |
| **cd\_keynew** | Shows the button currently pressed. |
| **cd\_keygone** | Shows the button pressed last. |
| **cd\_hwbeep** | A connection can be made to a beeper (digital output). |

**Server:**

|  |  |
| --- | --- |
| **Server0** | With read access, a pointer to the terminal data is retrieved. |
| **Keybitcode** | Binary code of the pressed button. |

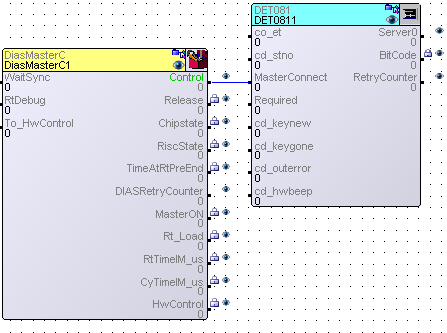
### **DET Terminal** (DET081, DET243, DET244, DET245)

**Hardware configuration**

A DET terminal is connected to the PLC over the SIGMATEK DIAS Bus. A node switch, with which the station number of the bus can be set, is located on the back of the terminal.

**Hardware Configuration in LASAL**

The following settings must be set in LASAL for the hardware class object.



**Clients:**

|  |  |
| --- | --- |
| **cd\_stno** | DIAS Bus station number set on the terminal (e.g. 16#01) |
| **cd\_outerror** | DIAS Bus error (e.g.: no terminal available...) |

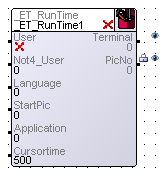
See ET class for remaining clients.

**Server:**

See ET class.

### **Step 2: Integrate ET\_RunTime Class**

Import the ET\_RunTime Class from the LASAL library into the project. During this process, several classes are added. Objects of the ET\_RunTime and ET\_User classes are needed for an operational project.



**Clients:**

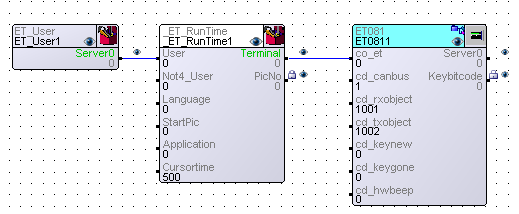
|  |  |
| --- | --- |
| User | A function for the user can be called in **ET\_User** with the press of a button. |
| Not4\_User | For type initialization only; of no interest to the user. |
| Language | Shows the currently selected language. |
| StartPic | Here, the desired start screen can be specified. |
| Application | Here, the **Virtual\_User** object for programming the user service can be connected. |

**Server:**

|  |  |
| --- | --- |
| Terminal | Connected to the co\_et server of the terminal class. In addition, various methods of the ET\_RunTime class can be called from other objects over an object channel from this server. |
| PicNo | The current screen number is output to this server. |

### **Step 3: Connecting Classes**

Place the hardware, runtime and user classes in a new object network. Connect the object as shown in the following screenshot.



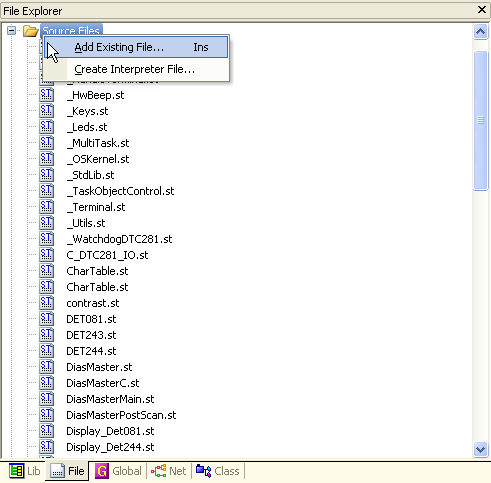
For the exact meanings of the possible object settings, see the core software documentation.

### **Step 4: Integrating LASAL Text Files**

In a Lasal Text project, lists containing information on screens, variables, text and button functions are created. These lists must be integrated into the LASAL project. A newly created LASAL project with a text kernel already contains lists with this information. The base image is contained therein.

To integrate user-defined lists into the LASAL project, follow the steps below: In the file explorer the base lists **Varlist.st**, **PictList.st**, **KeyTable.st**, **Textlists.st** and **CharTable.st** must first be selected and deleted.

Next, add the user-defined lists from the LASAL Text project with "Add Existing File...".



The entire LASAL project should be recompiled. In the online mode, send the entire project to the PLC.

If the project has been successfully compiled, an operational visualization project is now available in LASAL when the start button is pressed.

## Interpreter Functions

The following interpreter functions can be called from an object through an o**bject channel** (**\_Et*\_*RunTime**). Connect the **Terminal** server to the **\_Et*\_*RunTime** object.

**Important:** Send instructions only once!

### **Control\_Led Function**

With this function, the terminal LEDs can be controlled as desired.

**Input Parameters**

LED Number: Led\_Number

LED mode: Led\_Mode (0 => LED off, 1 => LED on, 2 => LED blinking)

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Switch Led \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 1 then

Et\_Connect.Control\_Led(Led\_Number := 0, Led\_Mode := 0);

server0 := 0;

elsif server0 = 2 then

Et\_Connect.Control\_Led(Led\_Number := 0, Led\_Mode := 2);

server0 := 0;

end\_if;

### **Control\_Pic Function**

With **Control\_Pic**, a screen can be opened. If an unavailable screen number is entered, the last screen is displayed.

**Input Parameters**

Screen number: Pic\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Switch Picture \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 5 then

Et\_Connect.Control\_Pic(Pic\_Number := 1);

server0 := 0;

elsif server0 = 6 then

Et\_Connect.Control\_Pic(Pic\_Number := 0);

server0 := 0;

end\_if;

### **Control\_Lan Function**

With this function, the project language can be changed. If an unavailable language is entered, the first language is displayed.

**Input Parameters**

Number of languages: Lan\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Switch Language \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 10 then

Et\_Connect.Control\_Lan(Lan\_Number := 0);

server0 := 0;

elsif server0 = 11 then

Et\_Connect.Control\_Lan(Lan\_Number := 1);

server0 := 0;

end\_if;

### **Control\_Inc Function**

This function is used to increment a variable.

**Input Parameters**

Variable number: Var\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Increment Variable \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 15 then

Et\_Connect.Control\_Inc(Var\_Number := 1);

server0 := 0;

end\_if;

### **Control\_Dec Function**

This function is used to decrement a variable

**Input Parameters**

Variable number: Var\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Decrement Variable \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 20 then

Et\_Connect.Control\_Dec(Var\_Number := 1);

server0 := 0;

end\_if;

### **Control\_RetPic Function**

This function is used to page back through the last 10 screens .

**Input Parameters**

None

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Return Picture \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 25 then

Et\_Connect.Control\_RetPic();

server0 := 0;

end\_if

### **Control\_Add Function**

With the Control\_Add function, a value can be added to a variable.

**Input Parameters**

Variable number: Var\_Number

Value added to variable: Value

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Add value to variable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 30 then

Et\_Connect.Control\_Add(Var\_Number : = 1, value : = 100);

server0 := 0;

end\_if

### **Control\_Sub Function**

With the Control\_Sub function value can be subtracted from a variable.

**Input Parameters**

Variable number: Var\_Number

Value subtracted from the variable: Value

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sub value to variable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 35 then

Et\_Connect.Control\_Sub(Var\_Number : = 1, value : = 200);

server0 := 0;

end\_if

### **Control\_Set Function**

Control\_Set is used to set a variable to a value

**Input Parameters**

Variable number: Var\_Number

New value of variable Value

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Set variable to value\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 40 then

Et\_Connect.Control\_Set(Var\_Number : = 6, value : = 0);

server0 := 0;

end\_if

### **Control\_Toggle Function**

Control\_Toggle toggles the value of a variable between 0 and 1.

**Input Parameters**

Variable number: Var\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Toggle a variable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 45 then

Et\_Connect.Control\_Toggle(Var\_Number : = 6);

server0 := 0;

end\_if

### **Control\_EditIO Function**

Sets the cursor to the output variable selected on the screen. The variable 0 is the same as variable 1 (screen-based) in ET\_MAE.

**Input Parameters**

Variable number: Var\_Number

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Edit variable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 50 then

Et\_Connect.Control\_EditIO(Var\_Number : = 0);

server0 := 0;

end\_if

### **Control\_VarNeg Function**

By calling this function, the sign of the variable can be changed to negative. The sign is only changed when the cursor is on the entry.

**Input Parameters**

None

Example:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Negative sign\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 55 then

Et\_Connect.Control\_VarNeg();

server0 := 0;

end\_if;

### **Control\_VarPos Function**

With this function call, the sign is reset to positive The sign is only changed when the cursor is on the entry.

**Input Parameters**

None

Example:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Positive sign\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if server0 = 60 then

Et\_Connect.Control\_VarPos();

server0 := 0;

end\_if;

These two function can also be called through the **FC\_User** function. The sign is only changed when the cursor is on the entry.

To call using a button, the minus and plus buttons can be assigned the function **FC** **Function** in the LASAL TEXT EDITOR.

Example of the program code in the ET\_User class (**FC\_User** function):

para := parameter;

case no of

1:

2:

3:

1000: toEt.Control\_VarNeg();

1001: toEt.Control\_VarPos();

end\_case;

## Number of Bytes in LASAL Text Classes

