STRATON
OEM Libraries

User’s guide

Content

1 Introduction 3
2 Running the Library Manager 3
3 Exploring libraries 3
4 Managing library items 4
5 Parameters of a function or function block 4
6 Definition of a profile 6
7 Definition of a datatype 7
8 Description texts 7
9 Exchanging library items 7

1 Introduction

The Library Manager is a visual tool that enables you to define and exchange the definition of embedded components so that they are known by the Workbench. The following types of components are supported:

- Functions and function blocks written in "C" language
- I/O boards (deprecated)
- Profiles

The Workbench works with a collection of libraries, stored on the hard disk with the Workbench. Each library may contains a list of "C" functions and function blocks, plus a set of I/O drivers, plus profiles.

2 Running the Library Manager

To run the Library Manager, open the Start menu of Windows, and then click on the "**Libraries**" item of the Workbench installation menu.

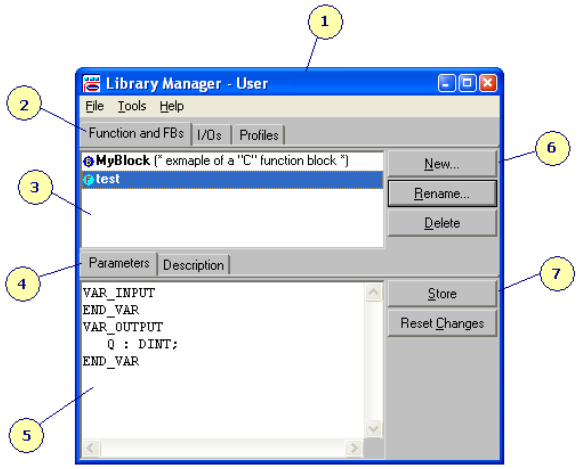
Important notes:

- You cannot run the Library Manager when the Workbench is open.
- Libraries are used by all projects developed with the installed Workbench. Changing or deleting library items may lead to errors when re-compiling projects that use these items.

When the Library Manager starts, it shows the contents of the "**User**" library. Use the "Open" command in the File Menu to select another library.

3 Exploring libraries

The Library Manager provides an easy-to-use interface for exploring the contents of the libraries, and for creating or changing registered components. Below is the frame of the Library Manager:



1. Edited library: The name of the open library is always visible in the caption bar of the manager.
2. Type of components: Use these tabs for selecting the type of components that you want to list: either the functions and function blocks, or the I/O boards (deprecated), or the definitions of profiles.
3. List of components: List of existing components in the open library, according to the selection tab (functions, profiles, types). Click on an element to see its details in the lower part of the window.
4. Selection of view for details: Select one these tabs for viewing/editing in the lower edit control either the parameters of the selected item, or its multiline description text.
5. Details: This box contains the detailed text corresponding to the selected tab (parameters or description). You can directly enter text in this box for defining or documenting the selected item.
6. Managing items: These buttons are main commands for creating and managing library components.
7. Editing details: Use these buttons to validate or cancel the changes you entered in the details box.

4 Managing library items

Use the buttons on the right of the main list to manage library items. You can:
- create a new item in the library
- rename the item selected in the list
- delete the item selected in the list

When you define a new item, you must give it a name and a description text. The description text will be displayed together with the name in all the selection menus of the Workbench.

The name of an item must fit to the following rules:

- The name cannot exceed 32 characters. It must begin with a letter or an underscore sign ('_'). Following characters must be letters, digits or underscores. There cannot be two consecutive underscore signs in the name.
  IMPORTANT: Although the name can be longer, the first 15 characters must be unique.
- The name of a function or function block must be unique within all libraries. You cannot define two blocks having the same name, even if they are located in different libraries.
- The name of an I/O driver must be unique within all libraries. You cannot define two drivers having the same name, even if they are located in different libraries.

5 Parameters of a function or function block

Each function or function block is described as a list of input and output parameters. When a function or function block is selected in the list, select the "Parameters" tab for viewing its parameters in the lower box. You can directly enter changes in the text definition of the parameters, and then press "Store" to validate the changes or "Reset" to cancel the changes.

Below is the syntax for defining the input and output parameters of a function or block:

```
VAR_INPUT
   definition of input parameters
END_VAR
VAR_OUTPUT
   definition of output parameters
END_VAR
```

Each parameter is identified by a name and a data type. It is described on one line of text according to the following syntax:

name **:** *type* **;**

Naming conventions:

- The name of a parameter must begin with a letter or an underscore sign ('_'). Following characters must be letters, digits or underscores. There cannot be two consecutive underscore signs in the name.
- The name must be unique in the list of parameters.
- A function has one and only one output parameter.
- A function or a function block cannot have more than 32 input parameters
- A function block cannot have more than 32 output parameters

Below are possible data types for a parameter:

BOOL	boolean (TRUE / FALSE)
SINT	short signed integer on 8 bits
INT	signed integer on 16 bits
DINT	signed integer on 32 bits
LINT	long signed integer on 64 bits
USINT	unsigned interger on 8 bits
UINT	unsigned interger on 16 bits
UDINT	unsigned interger on 32 bits
ULINT	unsigned interger on 64 bits
REAL	single precision floating point on 32 bits
LREAL	double precision floating point on 64 bits
TIME	Time of day (less than 24h - accuracy is 1ms)
STRING	Variable length character string - You do not have to specify a length
ANY	Any data type (see notes below)

If the name of an input pin is followed by "**[]**"characters, it means that the user must connect an array on this pin. Note that the connected array is considered as an IN/OUT parameter for the "C" function and or block.

If the name of an input pin is preceded by the "**@**" sign, it means that the user must directly connect a variable on the pin: The input cannot be a constant or complex expression. This allows definition of IN/OUT parameters for the "C" function and or block.

If one or more pin has the "ANY" data type, the user can connect any data to those pins, with the following restriction: All the pins having the "ANY" type must be connected to variable having the same data type.

6 Definition of a profile

A profile identifies a set of properties to be attached to a variable. The properties defined in the profile are filled by the user and embedded in the target with the compiled application. Using the Library Manager, you must define each property using the following syntax:

VAR *name* **:** *type* [**(** *bounds* **)**] [**:=** *value*] **;** [**(*** *description* ***)**]

name: The name of the property.

type: Must be a valid IEC 61131-3 data types. The following data types are supported:
BOOL : boolean
SINT / INT / DINT / LINT : signed integers
USINT / UINT / UDINT / ULINT : unsigned integers
REAL / LREAL : floating point reals
TIME : time of day
STRING : character string

bounds: In case of STRING data types, "bounds" is mandatory. It is a number that indicates the maximum length of the string. This number must be in between 1 and 15. In case of a numerical data type, "bounds" is optional, and indicates minimum and maximum allowed values, separated by ".."

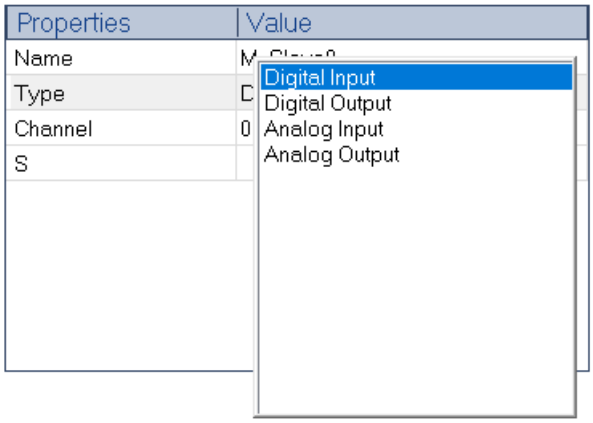
value: Default value for the property (optional), expressed using IEC syntax.

description: Text description to be displayed with the property in the grid where the user enters property values.

In case of an integer property, you can define a enumerated list of values, so that the property is edited as a drop list in the editor. Enumerated values are inserted in the comment of the property, using the syntax: **<<** *value* **=** *name* **>>**. Example:

```
VAR Type : DINT := 0; (* Type of IO
<< 0 = Digital Input >>
<< 1 = Digital Output >>
<< 2 = Analog Input >>
<< 3 = Analog Output >>
*)
VAR Channel : INT := 0; (* Point number *)
VAR S : USINT;
```

Will be edited as:



7 Definition of a datatype

Using the Library Manager, you can create some user-defined datatypes. A datatype is created with a type name, and a description. By default, the editor proposes you to declare a structure type. The type is entered as text, in the bottom area of the Library Manager, using the IEC 61131-3 language. Example:

```
STRUCT
   b : bool;
   i : int;
   arr : array [0 .. 9] of real;
   s : string(100);
END_STRUCT
```

Array dimensions and STRING maximum sizes should be specified. A member of a structure can have the type of another structure defined in the Library Manager (nesting structures).

You also can create a enumerated datatype, using the following syntax,

```
ENUM (VAL_A, VAL_B, … )
```

Enumerated values will internally be implemented as 8 bit unsigned data. Enumerated values correspond to 0, 1, … internal values. You can omit some values (consecutive coma signs).

8 Description texts

Each function, function block or I/O driver is documented with a multiline description text. When an item is selected in the list, select the "Description" tab for viewing its description text in the lower box. You can directly enter changes in the text, and then press "Store" to validate the changes or "Reset" to cancel the changes.

9 Exchanging library items

The commands of the "Tools" menu enables you to export or import library items so that you can exchange them among installed workbenches.

You can export several items together in the same export file. However, you cannot group function / function blocks together with I/O drivers or with profiles in the same export file. When importing items, you will be prompted to confirm any possible overwrite of the currently installed items. It is not possible to import in a library an element already defined in another library. To do that, you need to delete the existing element in the appropriate library before importing the item description.