

TURCK

Your Global Automation Partner

ARGEE

Getting Started

Contents

1	About these instructions	3
1.1	Explanation of symbols	3
1.2	Target groups	3
1.3	Feedback about these instructions	3
2	Notes on the software	4
2.1	Product identification	4
2.2	Requirements	4
2.3	Turck service	4
2.4	Devices with ARGEE function	4
3	For your safety	5
3.1	Intended use	5
3.2	Obvious misuse	5
4	ARGEE system description.....	6
4.1	Functional principle	6
4.2	ARGEE feature overview.....	6
4.3	ARGEE PRO and ARGEE Flow operating modes	7
4.4	ARGEE mode selection	8
5	Commissioning ARGEE.....	10
6	Overview of the ARGEE Editor	11
6.1	Setting up a project	12
6.1.1	Project tab.....	12
6.1.2	Opening projects.....	13
6.1.3	Saving projects	14
6.1.4	Displaying or printing the program code as a PDF.....	15
6.1.5	Naming the project.....	15
6.1.6	Configuring the IO block	16
6.1.7	Information on the ARGEE version.....	16
6.2	Navigating the project environment	17
6.2.1	Context menus in ARGEE.....	17
6.2.2	Operating aids	19
6.3	Setting up the program	22
6.3.1	Program variables	22
6.3.2	Initializing variables	23
6.3.3	Creating an array	23
6.3.4	Creating alias variables.....	25
6.3.5	Creating function blocks, states and additional global variables	26
6.3.6	Bitfields and I/O mapping	28
6.3.7	Importing libraries	32
6.3.8	Program blocks	33
6.4	Running the program.....	36
6.4.1	Compiling the program	36
6.4.2	Debug options.....	36
6.4.3	HMI.....	39
7	Turck branches — contact data	40

1 About these instructions

This document describes the Integrated Development Environment (IDE) and the basic functions of ARGEE 4.

1.1 Explanation of symbols

The following symbols are used in these instructions:



DANGER

DANGER indicates a hazardous situation with a high level of risk, which, if not avoided, will result in death or serious injury.



WARNING

WARNING indicates a hazardous situation with a medium level of risk, which, if not avoided, will result in death or serious injury.



CAUTION

CAUTION indicates a hazardous situation with a medium level of risk, which, if not avoided, will result in moderate or minor injury.



NOTICE

CAUTION indicates a situation which, if not avoided, may cause damage to property.



NOTE

NOTE indicates tips, recommendations and important information about special action steps and issues. The notes simplify your work and help you to avoid additional work.



MANDATORY ACTION

This symbol denotes actions that the user must carry out.



RESULT OF ACTION

This symbol denotes the relevant results of an action.

1.2 Target groups

These instructions are intended for personnel with PLC or high-level-language programming skills. These instructions must be read carefully by any person who commissions or operates the software.

1.3 Feedback about these instructions

We make every effort to ensure that these instructions are as informative and as clear as possible. If you have any suggestions for improving the design or if some information is missing in the document, please send your suggestions to techdoc@turck.com.

2 Notes on the software

2.1 Product identification

This manual applies to all ARGEE-capable Turck devices.

These instructions have been created on the basis of ARGEE 4.

2.2 Requirements

Required hardware and software:

- PC
- Browser (with HTML5 and JavaScript support)
- ARGEE Integrated Development Environment

Optional hardware and software:

- Ethernet cable
- ARGEE-capable device
- Turck Automation Suite (TAS)

The ARGEE programming environment and the Turck Automation Suite (TAS) are available for download at www.turck.com.



NOTE

Turck provides programming examples and libraries for ARGEE on www.turck.com. For more information on libraries, see [► 32].

2.3 Turck service

Turck supports you in your projects – from the initial analysis right through to the commissioning of your application. The Turck product database at www.turck.com offers you several software tools for programming, configuring or commissioning, as well as data sheets and CAD files in many export formats.

The contact data for Turck branches is provided at [► 40].

2.4 Devices with ARGEE function

The ARGEE 4 functionality is supported by runtime version 4.1.0.0 and later. BLCEN devices in the BL compact device series support ARGEE 3 functionality.

The ARGEE 4 runtime is not supported by BL compact devices.

3 For your safety

3.1 Intended use

ARGEE is a self-contained software package consisting of ARGEE engineering (IDE) and ARGEE runtime that allows Turck's block I/O modules (ARGEE-capable devices) to be used as logic controllers.

The software must only be used as described in these instructions. Any other use is not in accordance with the intended use. Turck accepts no liability for any resulting damage.

3.2 Obvious misuse

- ARGEE is not intended to be used for programming and controlling safety applications and must not be used for the protection of persons or property.

4 ARGEE system description

These instructions are limited to the description of the functions of ARGEE PRO mode. A detailed description of the functions of ARGEE Flow mode can be found in the ARGEE V3 Reference Manual.

4.1 Functional principle

The ARGEE IDE can be used to create ARGEE projects for Turck's ARGEE-capable devices.

Thanks to the programmed logic functions, ARGEE-capable devices are able to:

- Perform arithmetic operations
- Use internal variables, timers and counters up to a total size of 6 KB
- Exchange more extensive data with a PLC
- Execute control structures (conditional statement, loop, block structure)



NOTE

When connecting an ARGEE project to a PLC with PROFINET, the ARGEE GSDML with simple byte structure must be used.

4.2 ARGEE feature overview

- ARGEE works as a standalone application. A PLC is not required to run logic.
- ARGEE can be used in conjunction with a PLC.
- If the application loses communication with the PLC, ARGEE can assist in putting the application into a controlled operating state.
- ARGEE can be used for pre-processing or for quick, local reactions in conjunction with a PLC.
- Local control (ARGEE can monitor an application and send updates back to the PLC).
- ARGEE can be used to program HMI screens.
- ARGEE simulation mode (allows parameterization without an ARGEE-capable device).
- Export functions of the ARGEE program code in various formats
- Import functions for ARGEE projects
- Debug functions
- Support of for, while and if statements
- High-speed Digital IO (max. 4 kHz)
- ARGEE libraries
- Support of bit field data structures: Decimal point notation allows the control of individual bits
- Function blocks

ARGEE features in TAS

The following features can only be run in TAS:

- Activate/deactivate write and delete protection of ARGEE projects (also protected against factory reset)
- Load an ARGEE project on selected devices
- Delete an ARGEE project on selected devices

4.3 ARGEE PRO and ARGEE Flow operating modes

ARGEE has two operating modes: ARGEE PRO and ARGEE Flow.

ARGEE PRO is the default and recommended IDE mode.

ARGEE PRO

ARGEE PRO enables ARGEE-capable devices to:

- Perform arithmetic operations
- Use internal variables, timers and counters up to a total size of 6 KB
- Exchange more extensive data with a PLC
- Execute if statements and state sequences

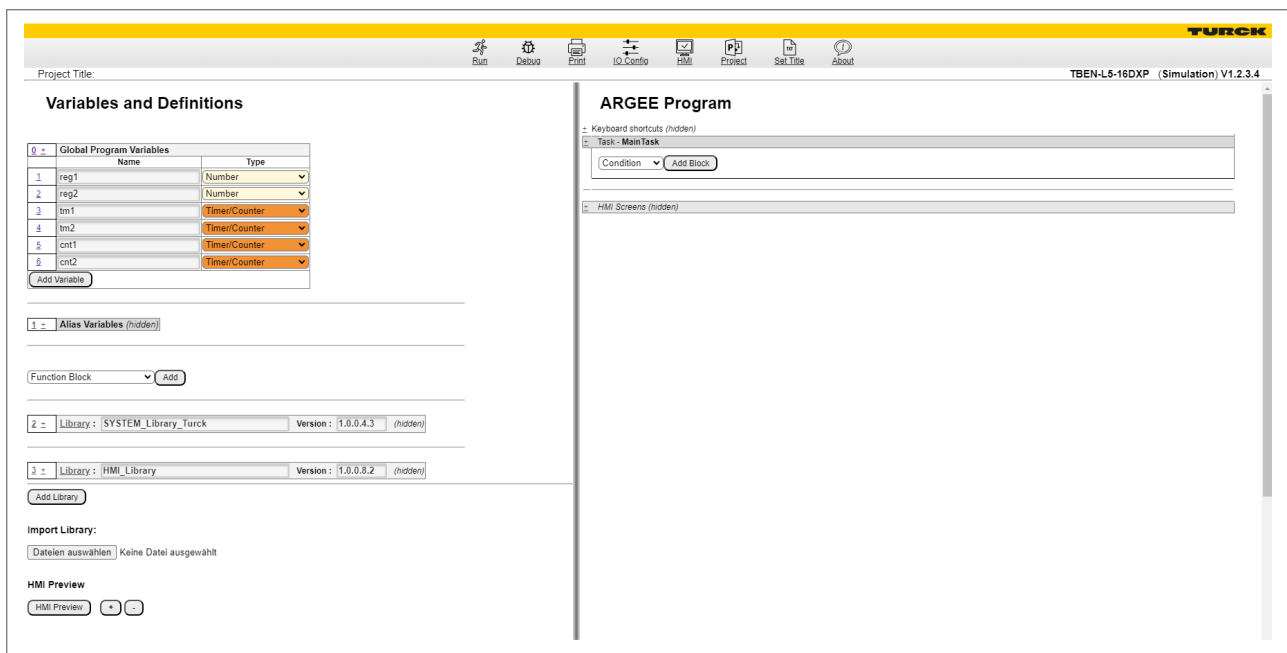


Fig. 1: ARGEE PRO interface

ARGEE Flow

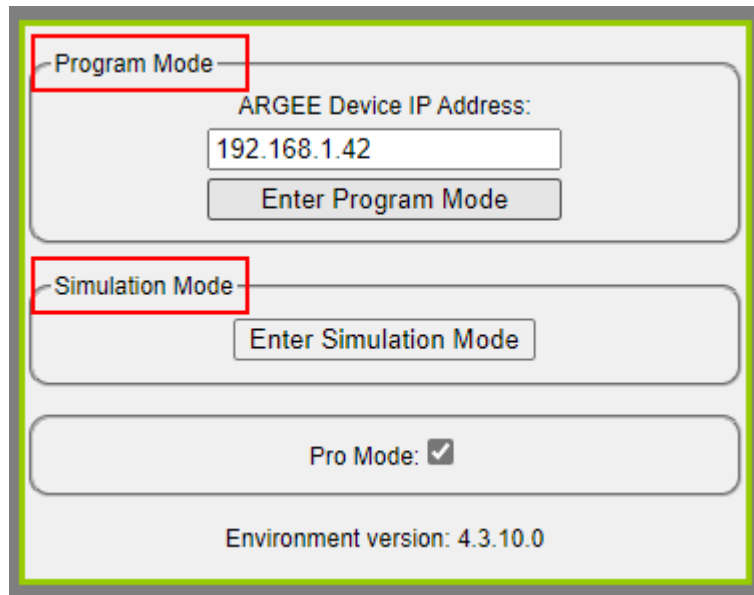
ARGEE Flow is a restricted configuration environment in which control functions can be defined and visualized via drop-down fields using if then links.

The ARGEE Flow environment consists of three areas:

1. Conditions
2. Operations
3. Actions

4.4 ARGEE mode selection

The desired mode must be selected before the programming process begins.



The screenshot shows a software interface for mode selection. It features three main sections, each with a red-bordered header: 'Program Mode', 'Simulation Mode', and 'Pro Mode'. The 'Program Mode' section includes a text field for 'ARGEE Device IP Address' containing '192.168.1.42' and a button labeled 'Enter Program Mode'. The 'Simulation Mode' section has a button labeled 'Enter Simulation Mode'. The 'Pro Mode' section has a checkbox labeled 'Pro Mode' which is checked. At the bottom of the interface, it displays 'Environment version: 4.3.10.0'.

Fig. 2: Mode selection

Program mode

At least one PC and one connected ARGEE-capable device are required to use program mode. In program mode, programs can be written for connected ARGEE-capable devices.

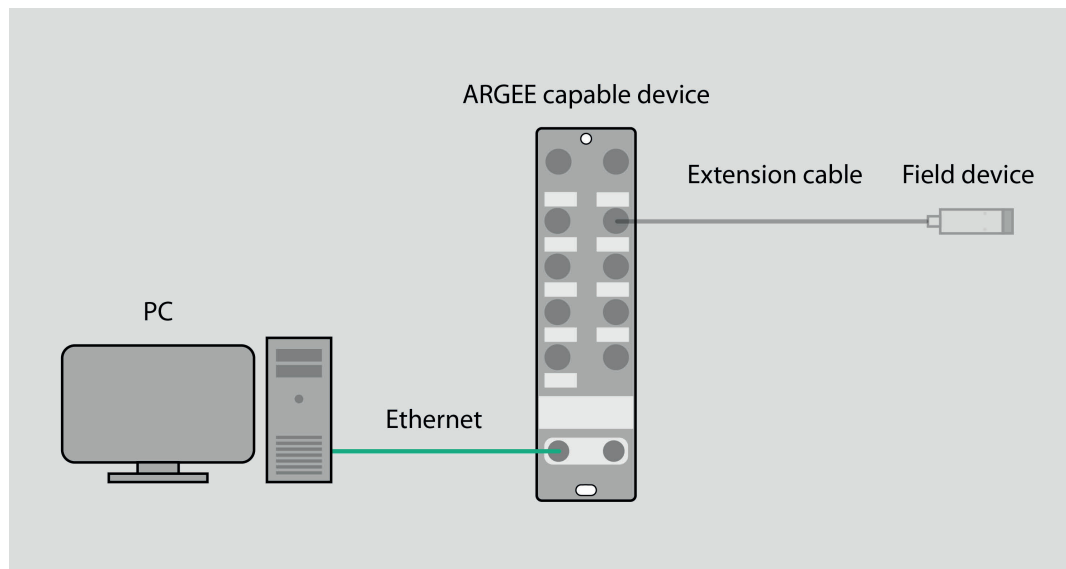


Fig. 3: Program mode application example

Simulation mode

Programs can be written in ARGEE simulation mode without the need for an ARGEE-capable device to be connected. A connected ARGEE-capable device is only simulated.

The device that is to be simulated can be selected from a list after simulation mode has started.

The ARGEE programs written in simulation mode can then be loaded onto ARGEE-capable devices.



NOTE

In simulation mode, not all ARGEE-capable devices are listed.

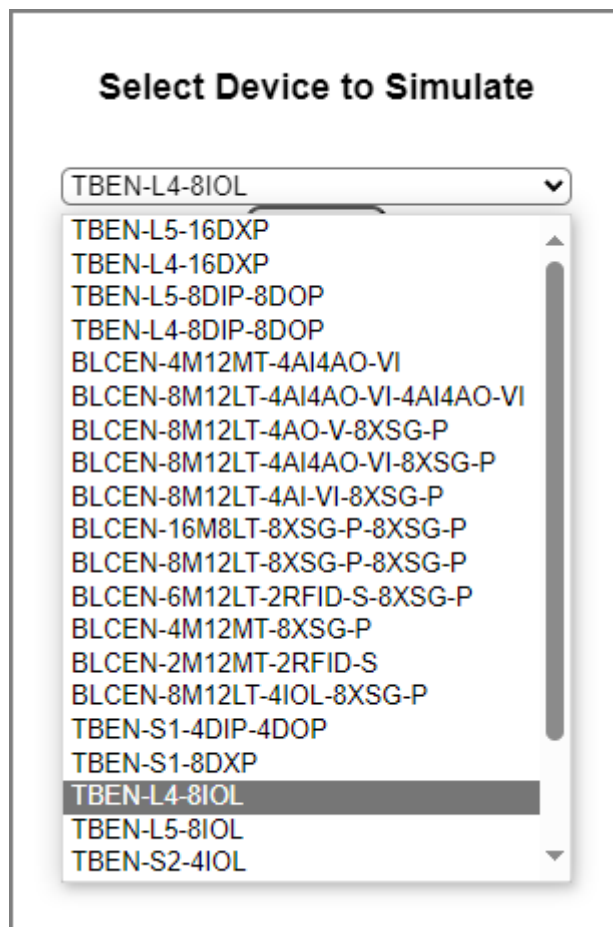


Fig. 4: Simulation mode device selection

5 Commissioning ARGEE

The ARGEE programming software can be downloaded free of charge at www.turck.com.

- ▶ Unzip the folder.
- ▶ Open **Start ARGEE Programming Environment.html** in the browser.
- ▶ Enter the IP address of the device. (Simulation mode can also be opened without entering an IP address.)

IP addresses for ARGEE-capable devices can be set via the Turck Automation Suite (TAS) or the website (web server) of the respective device. For more information on configuring the IP address, refer to the operating instructions for the respective device.

6 Overview of the ARGEE Editor

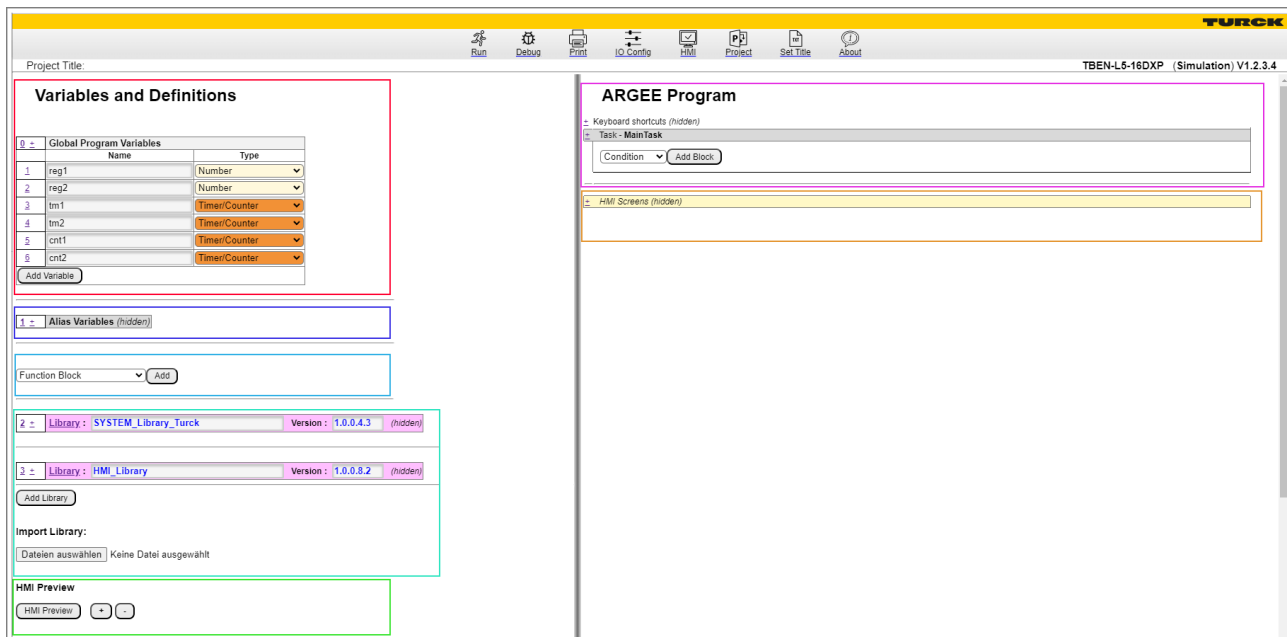


Fig. 5: ARGEE Editor main view

Function	Description
ARGEE program	The program code is written under the ARGEE Program tab.
Program variables	Program variables serve as named memory locations for changing information. The variable type can be changed by clicking the selected type.
Alias variables	Alias variables can be used to assign user-defined names to I/O points and PLC variables. Using alias variables can make code easier to understand. For an example of creating alias variables , see [► 25].
Function blocks, states, additional global variables	<p>Function blocks Function blocks are used to make the coding process faster, debug the code faster or reduce the scope of the program code. Function blocks can be used to reuse code.</p> <p>States A state is a named constant in the program.</p> <p>Additional global variables Additional global variables can be created via Additional Global Variables. For a more detailed description of function blocks, states and additional global variables, see [► 26].</p>
Libraries	Libraries are pre-built collections of function blocks, states and additional global variables that can be imported and used directly in ARGEE. You can also create libraries yourself. ARGEE 4 already includes the SYSTEM_Library_Turck and the HMI_Library . Additional libraries can be found at www.turck.com .
HMI preview	ARGEE can be used to program Human Machine Interface (HMI) screens. HMI screens act as the user interface of the program. The HMI preview can be used to display a programmed HMI screen before the program starts.
HMI screens	The code for parameterizing the HMI screens is written under the HMI Screens tab.

6.1 Setting up a project

6.1.1 Project tab

Clicking **Project** opens the Project tab.

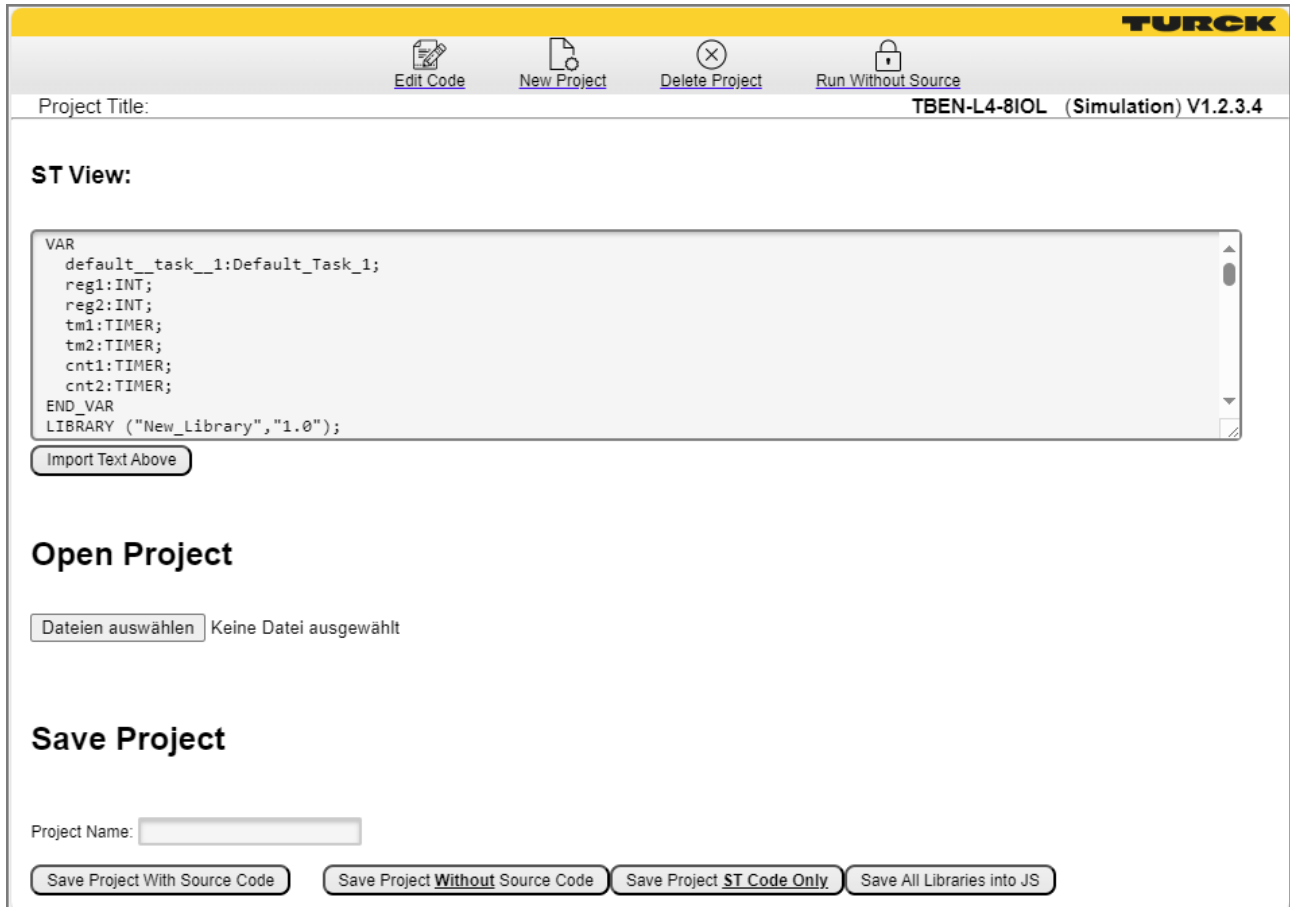


Fig. 6: Project tab

Creating a new project (option 1)

When ARGEE is opened for the first time, an empty project is automatically created.

Creating a new project (option 2)

- ▶ Click **Project** in the header.
- ▶ Click **New Project** in the header.
- ▶ Click **OK**.

Deleting a project

- ▶ Click **Project** in the header.
- ▶ Click **Delete Project** in the header.
- ▶ Click **OK**.

6.1.2 Opening projects

Importing a file

- ▶ Click **Project** in the header.
- ▶ Under the **Open Project** heading, click **Select Files**.
- ▶ Select the desired file.
- ▶ Click **OK**.

Importing ST text

This function is used to import text that was previously created in a text editor or was copied from another ARGEE project.

- ▶ Copy the ST text to be imported.
- ▶ Click **Project** in the header.
- ▶ Paste the ST text in the **ST View** window.
- ▶ Under the **ST View** window, click **Import Text Above**.

Note on the program version and device type

When importing an already created program into ARGEE, the following notification appears:

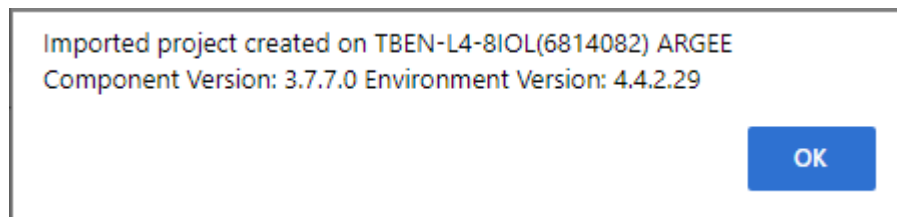


Fig. 7: Note on the program version and device type

The notification displays the device on which the program was created, the ARGEE component version and the environment version.

This information can be helpful in the event of an error or complications when running a program.

This information can be used to determine whether ARGEE engineering (environment version) and ARGEE runtime (component version) are compatible with each other.

6.1.3 Saving projects

Projects and libraries can be exported in various formats.

- Click **Project** in the header to access the storage functions.

Storage option	Description
Save Project With Source Code	The project is saved with the source code and can be opened and edited again. The project can only be loaded on ARGEE-capable devices. The project must be executed once.
Save Project Without Source Code	The project is saved without the source code and can no longer be edited. The project can only be loaded on ARGEE-capable devices. The project must be executed once.
Save Project ST Code Only	The project is saved as ST code.
Save All Libraries into JS	All libraries are saved as a JavaScript file. The file can be used so that ARGEE always starts with the saved libraries without the need to re-import the libraries each time. To do this, the glob_libs.js file in the ARGEE download directory must be replaced: Replacing the file: <ul style="list-style-type: none"> ► Rename the JavaScript file glob_libs.js and save it. ► Open the argee directory (...\\ARGEE V3\\ARGEE IDE v4.3.10.0\\internal). ► Replace the existing glob_libs.js in the directory with the newly created JavaScript file. ► Start ARGEE via the Start ARGEE Programming Environment.html file.

Storage space for ARGEE projects and ARGEE programs

The amount of space provided for ARGEE programs depends on the device type. The available memory is specified by the compiler when the project is created.

Device type	Storage space for ARGEE programs (program memory)
ARGEE-capable devices with IO-Link and RFID	32 KB
FEN20-16DXP	25 KB
All other ARGEE-enabled devices	42 KB

The storage space provided for ARGEE projects (project memory) is 260 KB for all ARGEE-capable devices.

6.1.4 Displaying or printing the program code as a PDF

A PDF of the program code is created by clicking **Print** in the header.

6.1.5 Naming the project

The project can be named or renamed by clicking **Set Title** in the header.

The title assigned here is displayed in the web server and in TAS.



NOTE

The assigned title is not the title of the project file.

6.1.6 Configuring the IO block

The IO-parameters (IO-Link, RFID, analog) of the connected or simulated IO block can be defined by clicking **IO Config** in the header.

The parameter data set in IO Config is always used as the first reference address.

The parameter configuration is part of the ARGEE project. Parameters that are changed during operation (in the webserver or in the ARGEE program code) are reset to the value defined in the ARGEE project when the ARGEE-capable device is restarted.

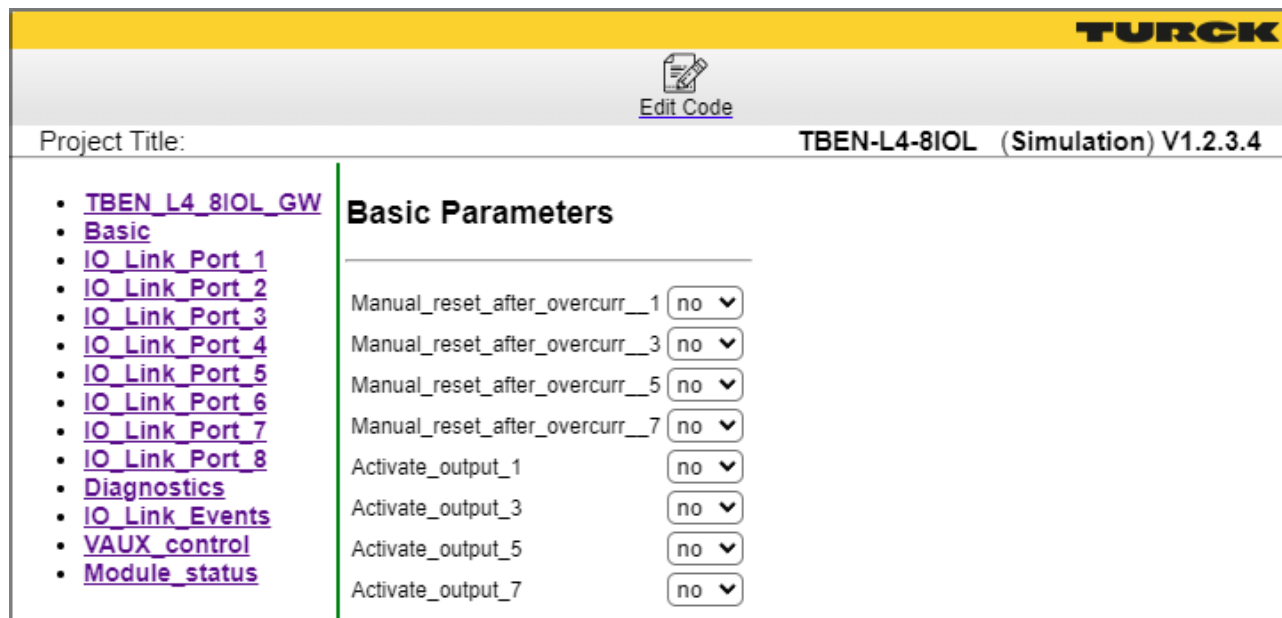


Fig. 8: IO Config menu

6.1.7 Information on the ARGEE version

The current IDE version and the current ARGEE runtime version can be viewed by clicking **About** in the header.

6.2 Navigating the project environment

6.2.1 Context menus in ARGEE

In ARGEE, it is possible to display context menus that contain additional functions in the respective menu sections.

Opening the context menu

- Click the number to the left of the respective text input field.

Context menus

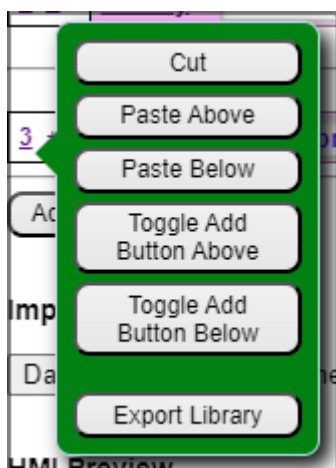


Fig. 9: Libraries context menu

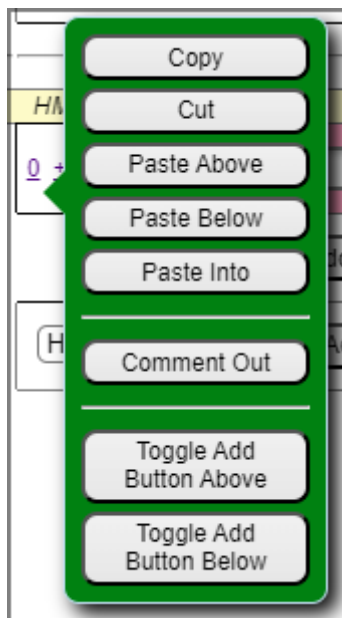


Fig. 10: ARGEE program context menu

Function	Description
Copy	Copies the variable to the clipboard.
Cut	Cuts the variable.
Paste Above	Pastes the copied or cut variable above the selected position.
Paste Below	Pastes the copied or cut variable below the selected position.
Toggle Add Button Above	Adds the Add Variable button above the selected position if no Add Variable button was previously present. Removes the Add Variable button above the selected position if an Add Variable button was previously present.
Toggle Add Button Below	Adds the Add Variable button below the selected position if no Add Variable button was previously present. Removes the Add Variable button below the selected position if an Add Variable button was previously present.
Comment	Inserts a comment field above the program variable.
Export Library	Exports the library.
Comment Out	Converts the selected statement into a comment field. Comments are not compiled when the program starts.

6.2.2 Operating aids

ARGEЕ provides the following operating aids to facilitate the operation of the software:

- Keyboard shortcuts
- Context menus in ARGEЕ
- Drag-and-select features

Keyboard shortcuts

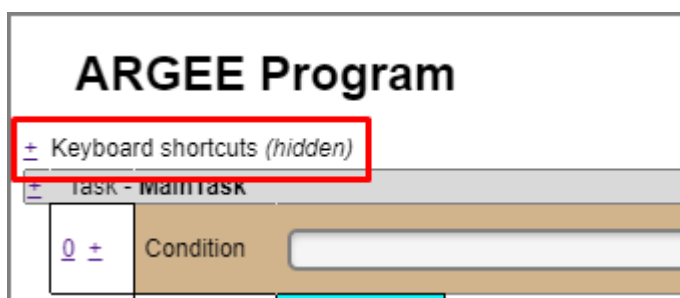


Fig. 11: Shortcuts (hidden)

ARGEЕ has a list of keyboard shortcuts.

Keyboard shortcut	Description	Requirements
Ctrl + Q	Opens a list of all project variables.	Click into the edit field of any program code.
Ctrl + L	Opens a list of local variables for functions, tasks, function blocks and procedures.	Click into the edit field of the function block.
Ctrl + I	Opens a list of the I/O variables.	Click into a line of the program code.
Ctrl + F	Opens a list of the built-in functions of the field previously clicked with the cursor.	
Ctrl + S	Opens a list of the state names.	
Ctrl + down arrow key	Opens the Add Below menu. In the Add Below menu, Click into a line of the program code. you can select different tasks to be inserted above the selected program line.	
Ctrl + up arrow key	Opens the Add Above menu. In the Add Above menu, you can select different tasks to be inserted above the selected program line.	Click into a line of the program code.
Ctrl + X	Cuts the selected statement.	
Ctrl + C	Copies the selected statement to the clipboard.	
Ctrl + Z	Undoes the last change (up to 32 actions).	
Ctrl + Y	Redoes the last undone change (up to 32 actions).	
Ctrl + D	Converts statements into comments. Comments are not compiled when the program starts.	Click in the numbered field to the left of the program block.
Ctrl + Shift + D	Converts comments into statements. Statements are compiled when the program starts.	Click in the numbered field to the left of the program block.
F1	Opens an overview of the keyboard shortcuts.	
F2	Opens a read-only view of the project.	
F3	Opens the IO description . The IO data of the connected devices is displayed in a table in the IO description .	

Keyboard shortcut	Description	Requirements
Double-click	Toggles the Add Block button above the line in the program code. Used to make the code clearer. See double-click example.	The double-click must be performed in the white box to the left of the line in the program code.
Ctrl + double-click	Toggles the Add Block button below the line in the program code. Used to make the code clearer. See double-click example.	The double-click must be performed in the white box to the left of the line in the program code.

Double-click example

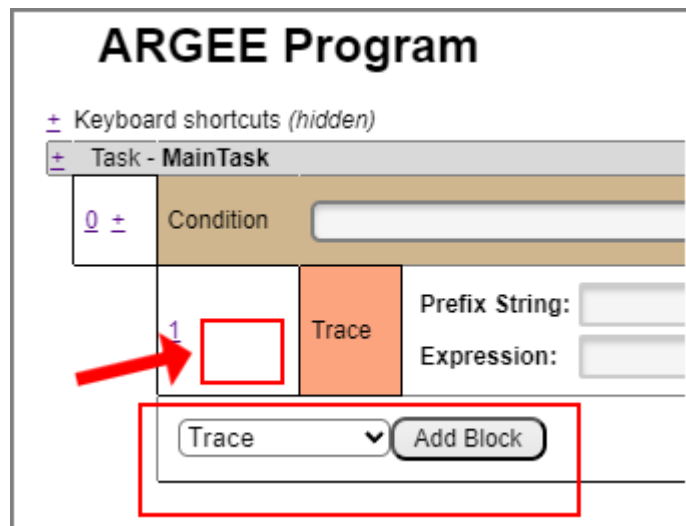


Fig. 12: Ctrl + double-click example before

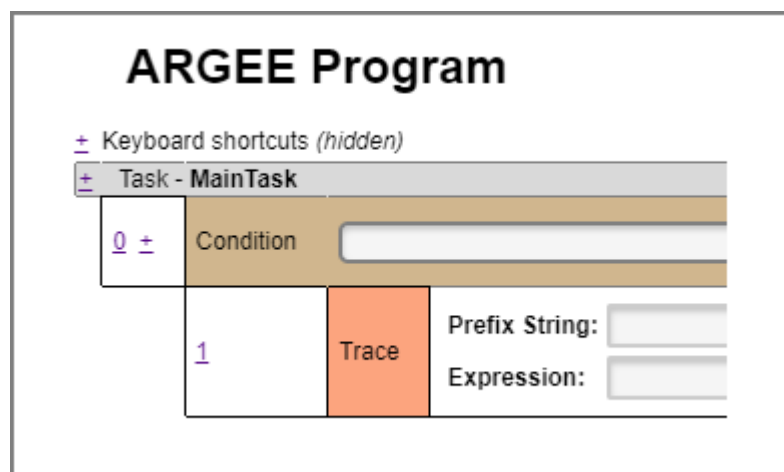


Fig. 13: Ctrl + double-click example after

Drag-and-select features

In ARGEE, it is possible to select several program blocks or variables using drag and select, and then copy, cut and paste them.

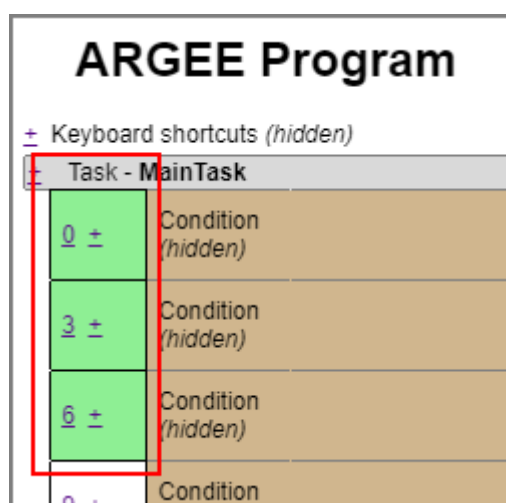
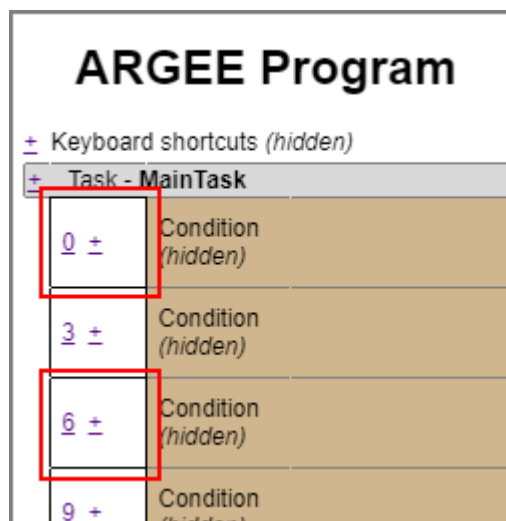


NOTE

The actions (copy, cut and paste) cannot be performed via the context menus. The selected code sections can only be copied, cut or pasted using the keyboard shortcuts Ctrl + C = copy, Ctrl + X = cut, Ctrl + V = paste.

Selecting rows:

- ▶ Use the mouse pointer to left-click in the white space of the first condition block to be selected, keeping the mouse button pressed.
- ▶ Drag the mouse pointer down to the last condition block to be selected and release the left mouse button.
- ▶ Use the keyboard shortcut to perform the desired action (Ctrl + C = copy, Ctrl + X = cut, Ctrl + V = paste).



6.3 Setting up the program

6.3.1 Program variables

Program variables serve as named memory locations for changing information.

The name of a global program variable cannot be duplicated anywhere else in the project.

The variable type can be changed by clicking the selected type.

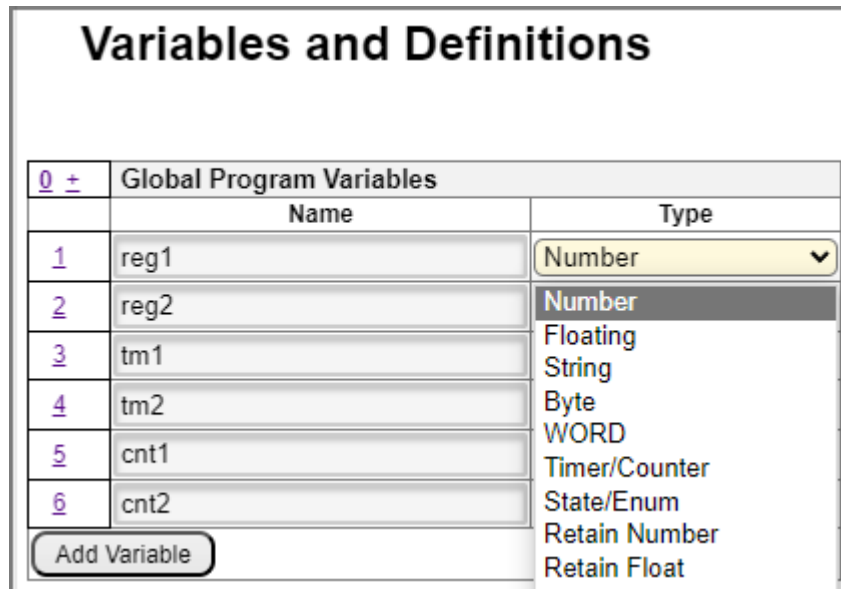


Fig. 14: ARGEE variable type



NOTE

The number of retain variables (retain number and retain float) is limited. The lifetime of the retain memory is limited to 100,000 write operations.

Variable type	Description
Number	Stores an integer value between -2,147,483,658 and 2,147,483,657 (4-byte signed integer).
Floating	Stores a floating-point number.
String	Stores integer values and/or letters.
Byte	A byte without a sign. Stores integers from 0 to 255 or hex values from 0x00 to 0xFF.
Word	Two bytes without a sign. Stores integers from 0 to 65,535 or hex values from 0x0000 to 0xFFFF.
Timer/Counter	Timer/Counter registers can store a value between -2,147,483,658 and 2,147,483,657.
State/Enum	State/Enum (enumeration) is used to create a state variable. State variables are used in state machines.
Retain Number	Stores integers between -2,147,483,658 and 2,147,483,657 during a power cycle. The value is synchronized approximately every 2 minutes.
Retain Float	Stores a floating-point number over a current cycle. The value is synchronized approximately every 2 minutes.

Additional predefined variables can be imported using libraries. See [▶ 32].

6.3.2 Initializing variables

Clicking the number next to the global variable to be initialized opens a context menu. Clicking **Init** opens a dialog box in which the value for the register of a program variable can be preset.

6.3.3 Creating an array

Clicking the number next to the global variable opens a context menu. Clicking **Make it Array** converts the variable into an array.

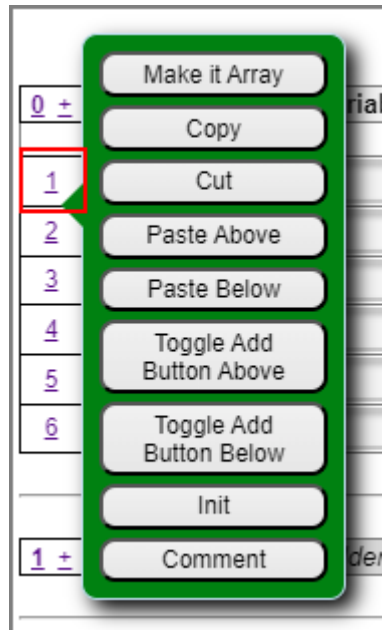


Fig. 15: Initializing variables

Initializing an array

Clicking the number next to the array to be initialized opens a context menu. Clicking **Init** opens another line of text in which the array can be initialized.

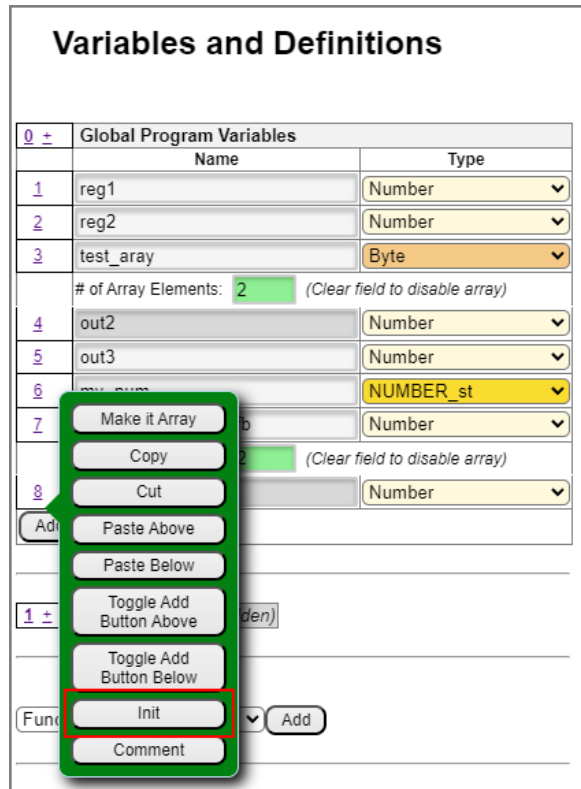


Fig. 16: Array Init

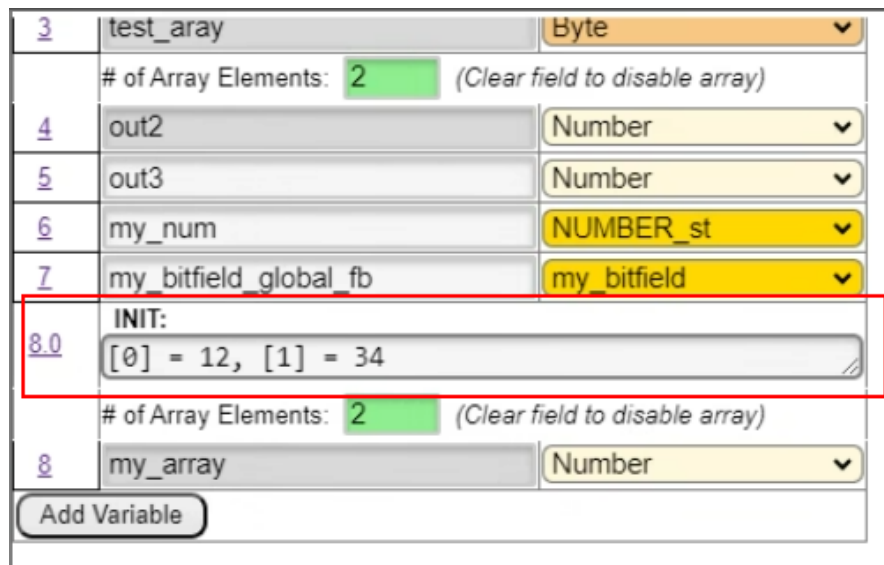


Fig. 17: Array Init text box

6.3.4 Creating alias variables

Alias variables can be used to assign user-defined names to I/O points and global variables.

Using alias variables can make code easier to understand.

For example, the I/O point `IO_Basic_Input_Input_value_4` becomes `car_sensor`.

Alias Variables		
	Name	IO Point
0	Friendly_IO_Point_Name	IO_Point
1	car_sensor	IO_Basic_Input_Input_value_4
2	greenlight	IO_Basic_Output_Output_value_6
3	PLC_in	IO_PLC_TO_ARGEE_Word4
Add Variable		

Fig. 18: Alias variable example



NOTE

I/O points referenced via alias variables are not recognized in the HMI view (I/O points can not be duplicated anywhere else in the project). This can lead to incorrect compilation/display of the programmed HMI.

As a workaround:

- ▶ Create a global variable.
- ▶ Create an alias variable with an I/O point.
- ▶ Create an assignment in the ARGEE program (destination: "name of the global program variable", expression: "name of the alias variable").
- ▶ Use the name of the global program variable in the HMI view.

6.3.5 Creating function blocks, states and additional global variables

A drop-down menu can be used to select whether a function block, a state or an additional global variable is to be created.

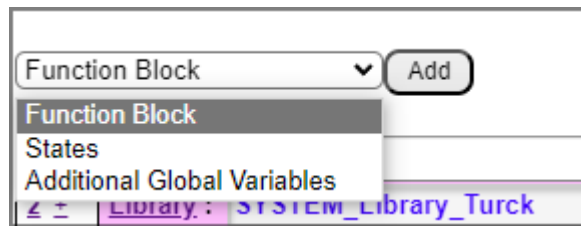


Fig. 19: Creating function blocks, states or additional global variables

Function Block : Function_Block_Name			Procedure
	Name	Type	Segment
0		Number	VARIABLE
Add Element			
States			
	Name		
0			
Add Element			
Additional Global Variables			
	Name	Type	
0		Number	
Add Variable			
Additional Global Variables			Add

Fig. 20: Function block, state and additional global variable in ARGEE

Function block

Function blocks are used to speed up the coding process and to make the code clearer. A detailed description of the function block types can be found in the table on the next page.

State

A state is a named constant. States can enforce a named value.

Additional global variables

Additional global variables can be used to create further variables in addition to the global variables. Additional global variables make it possible to create standalone libraries.

Function block type	Description
Procedure	<ul style="list-style-type: none"> ■ A procedure is similar to a function block but does not need to be instantiated. ■ Complex data structures and arrays can be passed to procedures as arguments. ■ Standard data types numbered by byte/num can be passed to procedures as input only arguments.
Structure	<ul style="list-style-type: none"> ■ A structure is a user-defined complex data type that consists of one or more data types (variables). ■ Structures can be used to combine related variables. ■ Structures are declared as standard variables. <p>Example: The structure of a cube could contain elements of width, height and length. The elements can be accessed directly using the decimal point notation ".", e.g. cube.length = 10.</p>
Function	<ul style="list-style-type: none"> ■ Functions are used to reduce the written code and to improve the readability of the code, especially if they are used in expressions, since the result of the function can be evaluated by the expression. ■ A function is limited to accessing complex data types as read only. ■ Functions must contain a single return variable called "Result". ■ Functions cannot write to global variables.
Task	<ul style="list-style-type: none"> ■ Tasks can be used to create multiple program sections within the project that run in parallel. ■ A task must be instantiated but does not need to be called (independent program section). ■ Tasks can be used to separate program blocks by using blocking statements. ■ Blocking statements are used by a task to wait for a specific condition to occur without blocking the execution of other tasks.
Bit field	<ul style="list-style-type: none"> ■ A bit field is a data structure that can be used to create individual named elements of ≤ 32 bits. ■ Bit fields can span more than 32 bits, but there must be no variable between two consecutive 32-bit boundaries within the bit field. ■ Individual elements (variables) of the bit field can be accessed via bitfieldname.elementname. ■ Bit fields can be used to organize contiguous data elements (e.g. IO-Link process data with multiple elements encoded in raw bytes). ■ It is possible to create structures that contain multiple bit fields (e.g. sensor with complex input and output process data).
Function block	<ul style="list-style-type: none"> ■ Function block can have the same arguments as a procedure. ■ A function block must be instantiated, which is why several instances of the same function block can exist in a project.
HMI block	<ul style="list-style-type: none"> ■ HMI blocks are used only on the HMI block screen. ■ An HMI block formats the layout of the screen but does not output any data on the screen. <p>Examples of HMI blocks are the HMI tables, row blocks and column blocks.</p>
HMI control	<ul style="list-style-type: none"> ■ HMI controls are used only on the HMI block screen. ■ An HMI control is an element that displays data or receives an input from the HMI. <p>Examples of HMI controls are HMI_Button, HMI_Display_Value and HMI_Dropdown.</p>

6.3.6 Bitfields and I/O mapping

Bitfields

ARGEE supports bitfields. Using bitfields, individual bits in the process data of connected devices can be read out in ARGEE.

The bits can be addressed directly in the program code using decimal point notation.

Bitfields are created as function blocks.

1 ±	Function Block : Inclinometer_IN	Bitfield ▼
	Name	size
0	Vendor_Specific	8 bit ▼
1	Scale	8 bit ▼
2	Winkel	16 bit ▼
3	Winkel_Invertiert	16 bit ▼

Fig. 21: Bitfields in ARGEE

In this example, the process data of an inclinometer was transferred to ARGEE. After transfer, the data can be accessed within the ARGEE programming environment.

I/O mapping

The process data created in the bitfields can be accessed via I/O mapping. The advantage of I/O mapping is that the combination of code and bitfields only needs to be set up once.

After setup, I/O mapping runs in the background.

Additional properties of I/O mapping:

- A lower error rate when writing the code because the data only needs to be set up once and does not need to be copied or referenced again in the code.
- The code becomes clearer, since fewer lines of code have to be written.

Setting up bitfields

The following section uses a connected inclinometer to explain how bitfields and I/O mapping work.

- ▶ Connect the IO-Link device to the ARGEE-capable device.
- ▶ Call up process data via the web server or the Turck Automation Suite (TAS).
- ▶ Transfer process data bit by bit in function blocks.

2 ±	Function Block : Inclinometer_IN		Bitfield ▼
	Name		size
0	Vendor_Specific		8 bit ▼
1	Scale		8 bit ▼
2	Angle		16 bit ▼
3	Angle_Inverted		16 bit ▼
Add Element			
3 ±	Function Block : Inclinometer_Out		Bitfield ▼
	Name		size
0	Output		8 bit ▼
Add Element			
4 ±	Function Block : Inclinometer		Structure ▼
	Name	Type	Segment
0	Input	Inclinometer_IN ▼	VARIABLE ▼
1	Output	Inclinometer_Out ▼	VARIABLE ▼
Add Element			
5 ±	Additional Global Variables		
	Name	Type	
0	Inclinometer_1	Inclinometer ▼	
Add Variable			
Additional Global Variables ▼ Add			

Fig. 22: Bitfields example

Setting up I/O mapping

In I/O mapping, the process data stored in the bitfields is accessed via an if function.

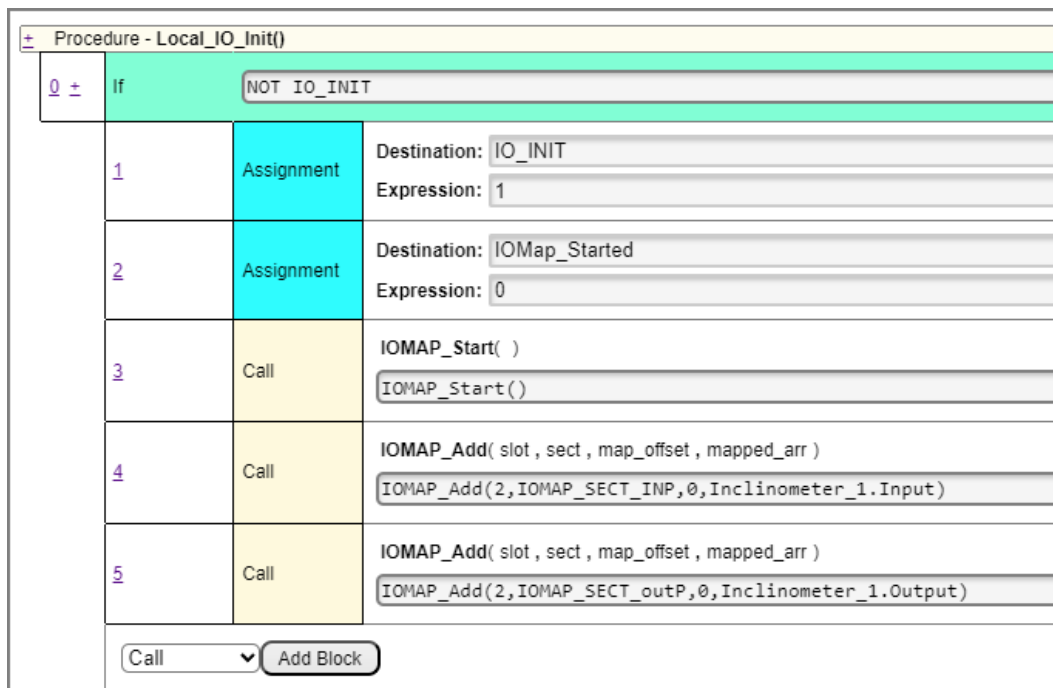


Fig. 23: I/O mapping example

To illustrate the code, the I/O description of the connected device can be opened via **F3**.

Parameters, diagnostics, inputs and outputs can be viewed here. In this example, the first IO-Link port is located on slot 2.

The map offset is the bit offset within the slot. The first data word at the input of port 1 has offset 0, the next data word has offset 16.

IO Description - Google Chrome

about:blank

Activate_output_ / 25 1 1 - yes

Slot 2 - IO_Link_Port_1

Input

Name	Bit Offset	Bit Length	Enum Values
Input_data_word_0	0	16	
Input_data_word_1	16	16	
Input_data_word_2	32	16	
Input_data_word_3	48	16	
Input_data_word_4	64	16	
Input_data_word_5	80	16	
Input_data_word_6	96	16	
Input_data_word_7	112	16	
Input_data_word_8	128	16	
Input_data_word_9	144	16	
Input_data_word_10	160	16	
Input_data_word_11	176	16	
Input_data_word_12	192	16	
Input_data_word_13	208	16	
Input_data_word_14	224	16	
Input_data_word_15	240	16	

Output

Name	Bit Offset	Bit Length	Enum Values
Output_data_word_0	0	16	
Output_data_word_1	16	16	
Output_data_word_2	32	16	
Output_data_word_3	48	16	

Fig. 24: ARGEE map offset and slot number

The input data, output data or diagnostic data that can be mapped are defined as a section in the I/O description.

The states are defined in the Turck Library:

21 ±	States
	Name
0	IO_MAP_SECT_INP
1	IO_MAP_SECT_OUTP
2	IO_MAP_SECT_DIAG

Fig. 25: ARGEE section

6.3.7 Importing libraries

Libraries are collections of theme-based function blocks and states.

Libraries can be downloaded from the Turck website or you can create them yourself.

ARGEE 4 already includes the **SYSTEM_Library_Turck** and the **HMI_Library**.

Additional libraries can be found at www.turck.com.

Importing libraries

- ▶ Download the library at www.turck.com.
- ▶ Extract the library from the ZIP file.
- ▶ Import the library.

2 ±

Library : **SYSTEM_Library_Turck**

Version : **1.0.0.4.3** (hidden)

3 ±

Library : **HMI_Library**

Version : **1.0.0.8.2** (hidden)

Add Library

Import Library:

Dateien auswählen

Keine Datei ausgewählt

Fig. 26: Importing the library

6.3.8 Program blocks

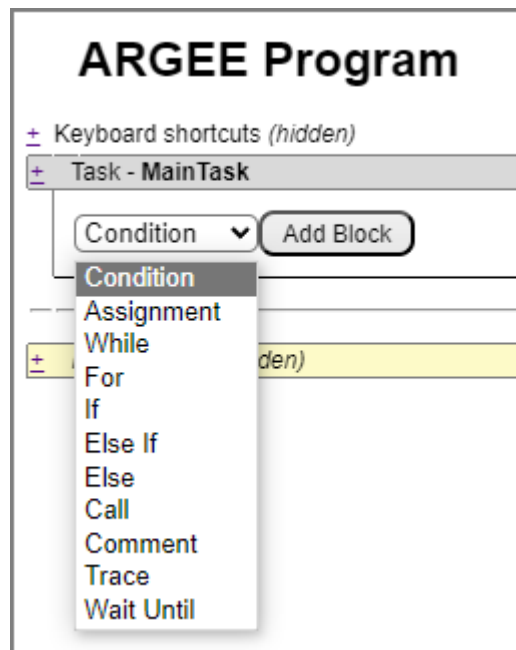


Fig. 27: Program blocks

Program block type	Description
Condition	Condition blocks are similar to an if statement and function like a ladder rung. If the expression in the condition is true, all blocks in it are executed as true. If the condition is false, all blocks in the condition are executed as false. Additional program blocks are used within the condition block. [▶ 35]
Assignment	An assignment can be used to assign a value to a variable. The value can be a constant, a complex expression or a function. The assigned value can be a variable, an output, an array or a member of a structure.
While	While loops execute the blocks within the loop for as long as the expression remains true.
For	For loops are executed until the iterator variable of the for loop corresponds to the end value. Start and end values can be constants, variables or expressions.
If	If blocks only execute the program blocks contained in the block if the expression of the if statement is true.
Else if	Else if blocks can be placed directly below if blocks. If the original if block is executed as false, additional expressions can be checked.
Else	Else blocks can be placed directly below if blocks or else if blocks. Program blocks within the else block are only executed if both the if block and all else if blocks are executed as false.
Call	Call blocks execute functions, procedures or function blocks. These can be integrated or user-defined functions. The Ctrl + Q keyboard shortcut displays the available functions. Possible arguments for a function are displayed above the expression box in a short help line.

Program block type	Description
Comment	Comment blocks can be used to add comments above or below other program blocks. Comments can be used within the program blocks below the if blocks, else if blocks and else blocks. If comment blocks are used to separate if blocks, else if blocks or else blocks, this will result in a compilation error.
Trace	Trace statements can be used for troubleshooting. When a program reaches a trace block, the expression value is appended to the prefix string stored in the trace buffer with a time stamp. The ring-shaped trace buffer can store the last 50 traces displayed on the debug screen.
Wait until	Wait until is a blocking statement that stops the task from executing until the statement in the expression becomes true. Other tasks (see task [► 27]) are executed in the project as usual.

The following program blocks can only be selected within the condition block:

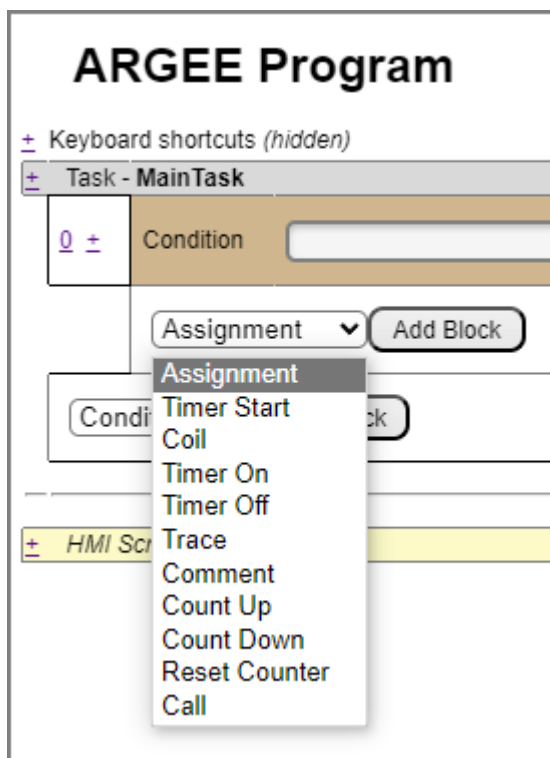


Fig. 28: Condition program blocks

Program block type	Description
Timer Start	Timer Start starts a timer if the higher-level condition block is true. As long as the condition remains true, the timer starts again at 0 in the subsequent program cycles.
Coil	Coil can be bound to a physical output or program variable. Coil remains active as long as the higher-level condition block is true. If the condition is false, the coil is deactivated.
Timer On	Timer On is an on-delay timer. The timer starts when the condition is true. The timer counts up as long as the condition remains true. If the condition becomes false, the timer is reset.
Timer Off	Timer Off is an off-delay timer. The timer starts when the condition is false. The timer counts up as long as the condition remains false. If the condition becomes true, the timer is reset.
Count Up	Count Up gradually increases the counter value when the edge of the condition rises.
Count Down	Count Down gradually decreases the counter value when the edge of the condition rises.
Reset Counter	Reset Counter resets the counter value to 0 when the edge of the condition rises.

6.4 Running the program

6.4.1 Compiling the program

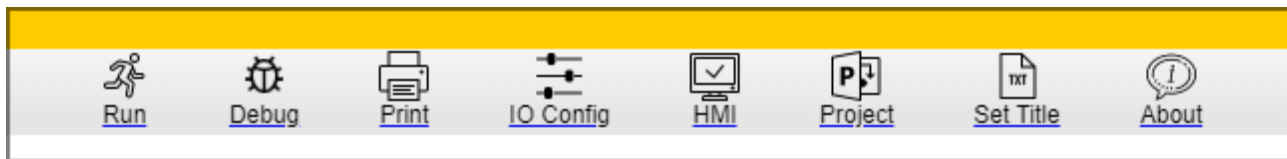


Fig. 29: ARGEE header

Clicking **Run** compiles the code.

1. When you click **Run**, the code is checked for errors.
2. If the code contains errors, a fault signal is output and the section of code containing the error is described. In addition, the line containing the error is highlighted in red.
3. If the code does not contain any errors, ARGEE loads the code onto the ARGEE-capable device.
4. ARGEE calculates the memory space required and the memory space still available for the code.
5. ARGEE switches to the **Debug** screen.

Example fault signal

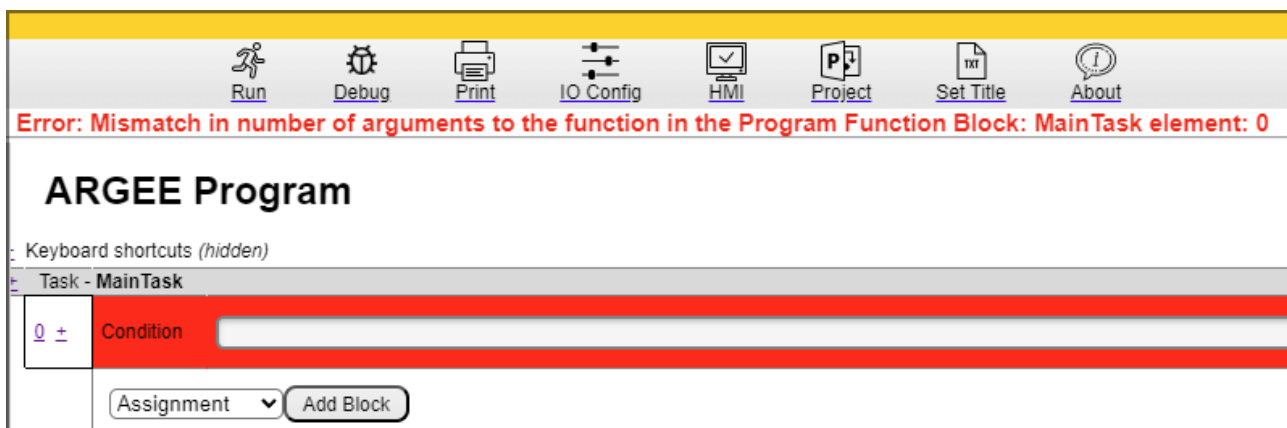


Fig. 30: ARGEE error

6.4.2 Debug options

In the debug menu, the compiled code can be checked for errors.

Functions of the debug menu:

- Breakpoints
- Trace
- Header functions

Breakpoints

Breakpoints can be used to stop the program at a desired statement. The compiled program runs from the beginning to the first breakpoint and is stopped.

Breakpoints can be switched on and off by clicking the number of the statement to be stopped.

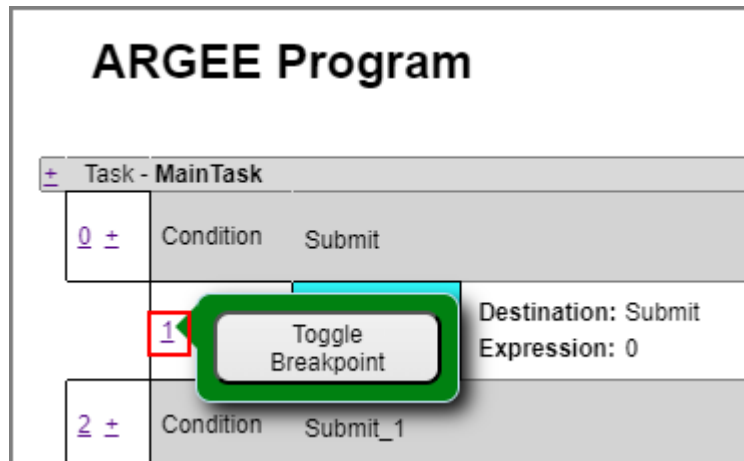


Fig. 31: Setting a breakpoint

Debug main menu



Fig. 32: Debug menu header

Function	Description
Edit Code	Edit Code opens the ARGEE Program screen.
HMI	A previously programmed HMI screen can be displayed via HMI .
Halt	Stops the program.
Step	In order to use Step , the program must have been stopped via Halt beforehand. Step can be used to go through the program code line by line. One click on Step equates to one line in the program code.
Continue	If the ARGEE program was previously stopped via Halt , it can be run normally again via Continue .
Modify Vars	Modify Vars stands for "Modify Variables." Variables in the Runtime Status window can be changed via Modify Vars . Recently changed variables are shown in yellow. Modify Vars can be used to manually control ARGEE in debug mode.
Hex	Changes the number representation to hexadecimal notation.
Dec	Changes the number representation to decimal notation.
Bin	Changes the number representation to binary notation.
Def	Changes the number representation to the default value.

Trace

Trace statements are used to measure the runtime behavior of a program.

Trace statements must be set manually in the program code.

The trace function can be used to track how long each state lasts and which statements have been visited in which order.

While the program continues, a list of the last 50 states can be generated by clicking **Pause/Resume**.

The list displays exactly to the millisecond in which line of code which data has been transferred.

The **Clear Trace** button can be used to delete the data from the list.

Trace example

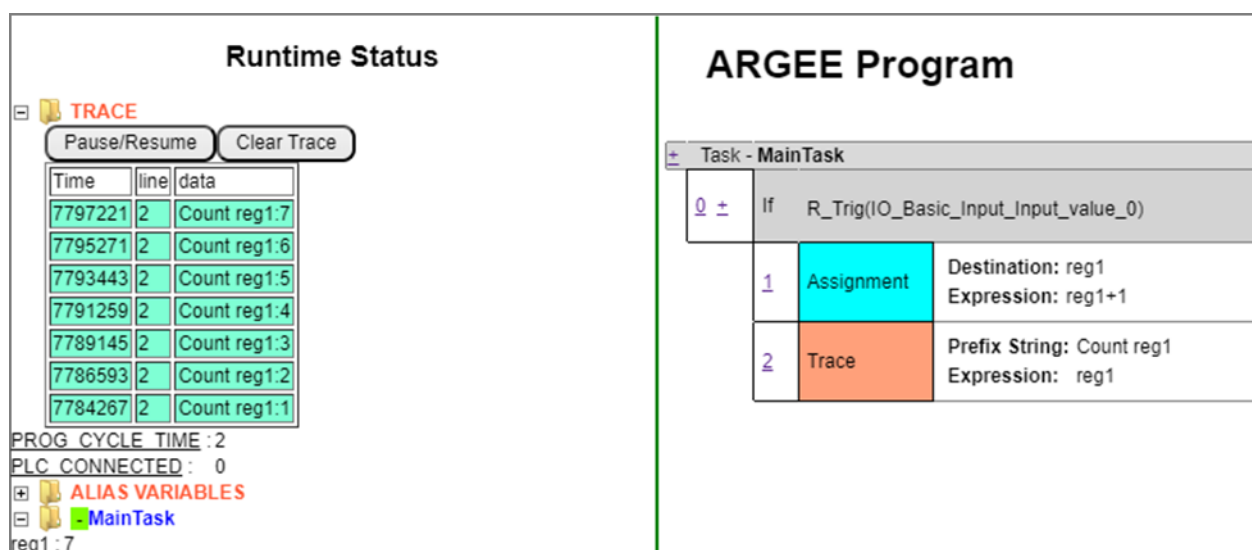


Fig. 33: Trace example

When input 0 changes from off to on, the program increases reg1 by 1.

The value of reg1 is then listed in the trace buffer. The trace displays the time stamp in ms in the left-hand column (Time) and the prefix string ("Count reg1") in the right-hand column (Data). The prefix string is appended to the value of reg1.

Between the first switch-on (7,784,267 ms) and the second switch-on (7,786,593 ms) of the input, 2326 ms (2.3 s) have elapsed.

6.4.3 HMI

In ARGEE, it is possible to program various HMI screens, which can be used to visualize and operate ARGEE programs. Four different HMI screen types and a comment function can be selected under the HMI Screens tab:

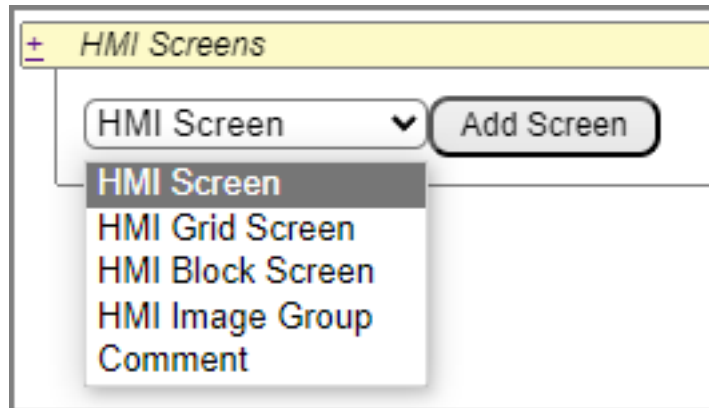


Fig. 34: Selecting ARGEE HMI screens

Type	Description
HMI screen	Data (e.g. basic operator inputs) can be displayed in one column from top to bottom.
HMI grid screen	Data can be displayed in tables, grids and columns. The HMI grid screen supports the display of images.
HMI block screen	The HMI Block Screen option allows the user to fully customize the HMI screen. The individual elements of the HMI screen are written as function blocks and saved in the HMI library. The function blocks are created in JavaScript. Additional functions for the HMI screen can be inserted using the additional blocks created.
HMI image group	Serves as a storage container for images that are used on the HMI screen.
Comment	Comment can be used to create comments in the HMI Screens section.

7 Turck branches — contact data

Germany	Hans Turck GmbH & Co. KG Witzlebenstraße 7, 45472 Mülheim an der Ruhr www.turck.de
Australia	Turck Australia Pty Ltd Building 4, 19-25 Duerdin Street, Notting Hill, 3168 Victoria www.turck.com.au
Austria	Turck GmbH Graumanngasse 7/A5-1, A-1150 Vienna www.turck.at
Belgium	TURCK MULTIPROX Lion d'Orweg 12, B-9300 Aalst www.multiprox.be
Brazil	Turck do Brasil Automação Ltda. Rua Anjo Custódio Nr. 42, Jardim Anália Franco, CEP 03358-040 São Paulo www.turck.com.br
Canada	Turck Canada Inc. 140 Duffield Drive, CDN-Markham, Ontario L6G 1B5 www.turck.ca
China	Turck (Tianjin) Sensor Co. Ltd. 18,4th Xinghuazhi Road, Xiqing Economic Development Area, 300381 Tianjin www.turck.com.cn
Czech Republic	TURCK s.r.o. Na Brně 2065, CZ-500 06 Hradec Králové www.turck.cz
France	TURCK BANNER S.A.S. 11 rue de Courtalin Bat C, Magny Le Hongre, F-77703 MARNE LA VALLEE Cedex 4 www.turckbanner.fr
Hungary	TURCK Hungary kft. Árpád fejedelem útja 26-28., Óbuda Gate, 2. em., H-1023 Budapest www.turck.hu
India	TURCK India Automation Pvt. Ltd. 401-403 Aurum Avenue, Survey. No 109 /4, Near Cummins Complex, Baner-Balewadi Link Rd., 411045 Pune - Maharashtra www.turck.co.in
Italy	TURCK BANNER S.R.L. Via San Domenico 5, IT-20008 Bareggio (MI) www.turckbanner.it
Japan	TURCK Japan Corporation ISM Akihabara 1F, 1-24-2, Taito, Taito-ku, 110-0016 Tokyo www.turck.jp

Korea	Turck Korea Co, Ltd. A605, 43, Iljik-ro, Gwangmyeong-si 14353 Gyeonggi-do www.turck.kr
Malaysia	Turck Banner Malaysia Sdn Bhd Unit A-23A-08, Tower A, Pinnacle Petaling Jaya, Jalan Utara C, 46200 Petaling Jaya Selangor www.turckbanner.my
Mexico	Turck Comercial, S. de RL de CV Blvd. Campestre No. 100, Parque Industrial SERVER, C.P. 25350 Arteaga, Coahuila www.turck.com.mx
Netherlands	Turck B. V. Ruiterlaan 7, NL-8019 BN Zwolle www.turck.nl
Poland	TURCK sp.z.o.o. Wroclawska 115, PL-45-836 Opole www.turck.pl
Romania	Turck Automation Romania SRL Str. Siriului nr. 6-8, Sector 1, RO-014354 Bucuresti www.turck.ro
Sweden	Turck AB Fabriksstråket 9, 433 76 Jonsered www.turck.se
Singapore	TURCK BANNER Singapore Pte. Ltd. 25 International Business Park, #04-75/77 (West Wing) German Centre, 609916 Singapore www.turckbanner.sg
South Africa	Turck Banner (Pty) Ltd Boeing Road East, Bedfordview, ZA-2007 Johannesburg www.turckbanner.co.za
Turkey	Turck Otomasyon Ticaret Limited Sirketi Inönü mah. Kayisdagi c., Yesil Konak Evleri No: 178, A Blok D:4, 34755 Kadiköy/ Istanbul www.turck.com.tr
United Kingdom	TURCK BANNER LIMITED Blenheim House, Hurricane Way, GB-SS11 8YT Wickford, Essex www.turckbanner.co.uk
USA	Turck Inc. 3000 Campus Drive, USA-MN 55441 Minneapolis www.turck.us

TURCK

Your Global Automation Partner

Over 30 subsidiaries and
60 representations worldwide!

100048240 | 2024/06



www.turck.com