



miCon-L

First steps

ProSign GmbH

Werner-Heisenberg-Straße 1
D-39106 Magdeburg

phone: +49 (0) 391 563 068 90
fax: +49 (0) 391 563 068 99
email: support@micon-l.de
web: www.pro-sign.de

BARTH® Elektronik GmbH

Gewerbepark BW-Depot
Im Depot 1-3
D-49838 Lengerich/Emsland

phone: +49 (0) 5904 499 970
email: info@barth-elektronik.de
web: www.barth-elektronik.com

Contents

Contents.....	3
1 Introduction and goal of the document.....	4
2 Newcomer	5
2.1 What is miCon-L?	5
2.2 User interface of miCon-L	6
2.3 Software download and installation.....	7
2.4 Start software	8
2.5 Connect mini-PLC to the computer	9
2.6 Example with STG-600.....	10
3 Changer	13
3.1 What is different or new in miCon-L 7.1?	13
3.2 Project import	15
4 More information	16
4.1 Where can I find more information?	16
4.2 Example projects	17
4.3 Good to know / tips	18
4.4 How does the miCon-L technology package work?.....	20
4.5 Function block overview.....	22
4.6 Glossary and data types	24
5 Revision history	29

1 Introduction and goal of the document

This document presents essential aspects of the graphical programming system miCon-L 7.1 to enable a quick introduction. It is aimed at those switching from an older miCon-L version ($\leq V3.8.1$) as well as newcomers. At the same time, it is intended as a central information document to provide quick access to further information.

The “First steps” document is structured in such a way that it begins in chapter 2 with very basic information, which is essentially aimed at newcomers.

→ [Chapter 2 - Newcomer](#)

Chapter 3 summarizes information on new features of version 7.1 and notes on project import from older versions.

→ [Chapter 3 - Changer](#)

Further useful information follows from chapter 4 as an overview.

→ [Chapter 4 - More information](#)

2 Newcomer

2.1 What is miCon-L?

miCon-L is a graphical programming software which is based on the graphical programming system iCon-L. The software is specially adapted for the programming of the BARTH lococube® mini-PLCs from the STG series.

Block diagrams are used for the description or modeling of technical systems so that the user does not need to learn textual programming languages.

A program, or application design, is created through the graphical connection of function blocks. The block sequence determines the processing order and the connections visualize the data flow. The function blocks are represented by technical symbols that are very easy to understand.

Many function blocks can be parameterized and, for example, their data type can be adapted. Such simple adjustments are implicitly visualized via the different colors. However, parameterization can also be very extensive and is always tailored to the respective application purpose.

The BARTH lococube® mini-PLC is a group of Programmable Logic Controllers which can be used for a wide range of industrial applications, e.g. in the automotive or engineering field. It has compact dimensions and is very suitable for the usage in harsh environmental conditions. Current models offer the possibility of transferring data via CAN-Bus or a Modbus-Slave interface.

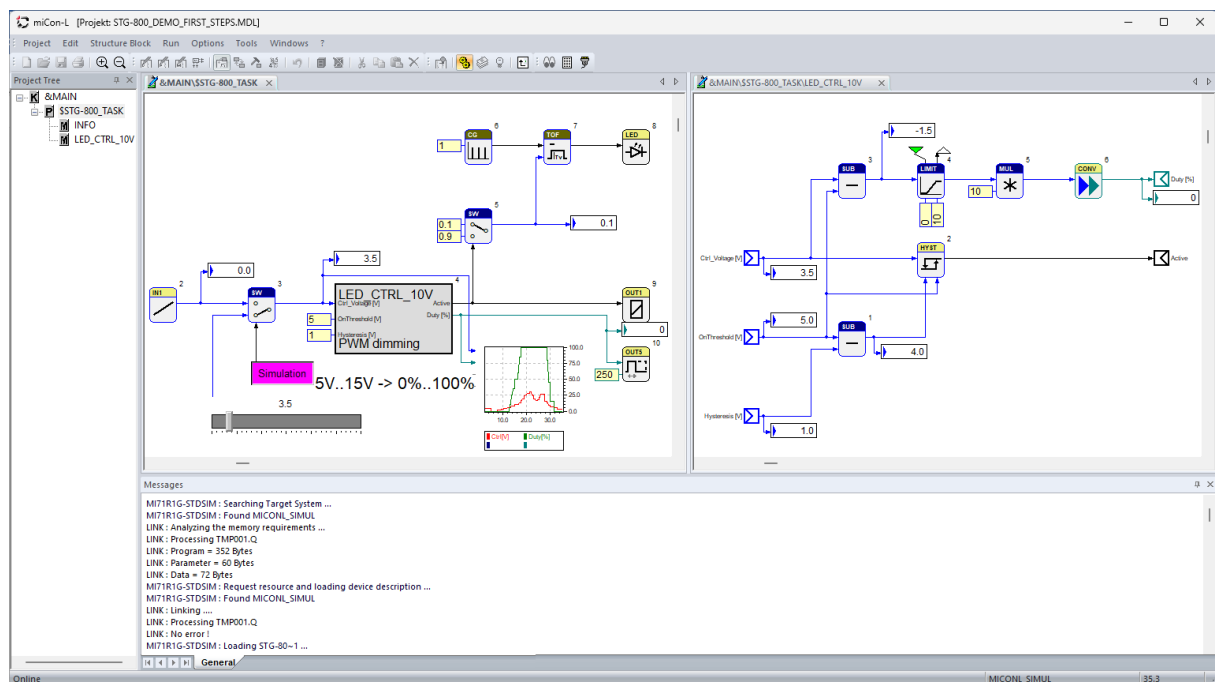


Figure 1: miCon-L in Online Mode (Example project for LED control)

2.2 User interface of miCon-L

The user interface of miCon-L has three basic modes: “Edit Mode”, “Run Mode” and “Online Mode”.

➡ Commands can be called up via the menu, the toolbar, the context menu (right-click) and key combinations.

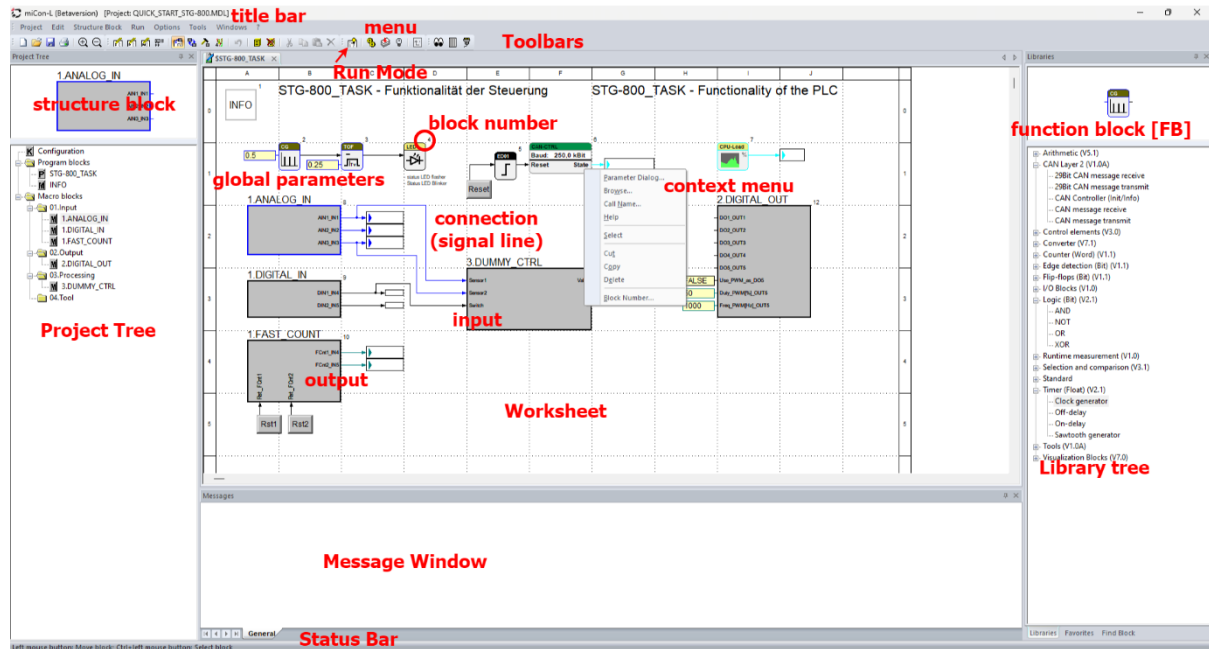


Figure 2: User interface in Edit Mode

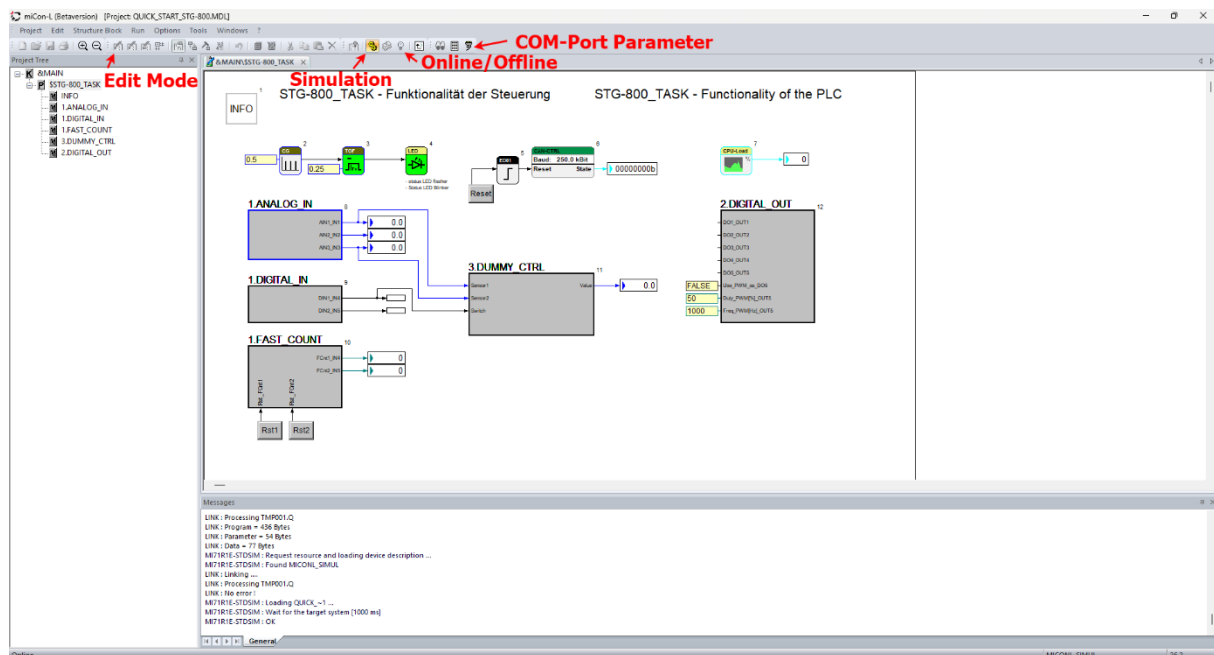


Figure 3: User interface in Online Mode

No function blocks or structure blocks can be inserted in Run Mode or Online Mode. However, the created program can be parameterized and/or loaded into a target system (e.g. “Simulation”) and observed.

FirstSteps_EN_V2.0.0

2025-06-10

<https://micon-l.de>

2.3 Software download and installation

The software download of the current miCon-L version is offered for the English and the German version via a separate subpage of the miCon-L website.

German: <https://barth-elektronik.com/miCon-L/>

English: <https://barth-elektronik.com/en/miCon-L/>

After downloading the current version, it can be installed by executing the corresponding setup file (e.g. miConL71R1EN_Setup.exe).

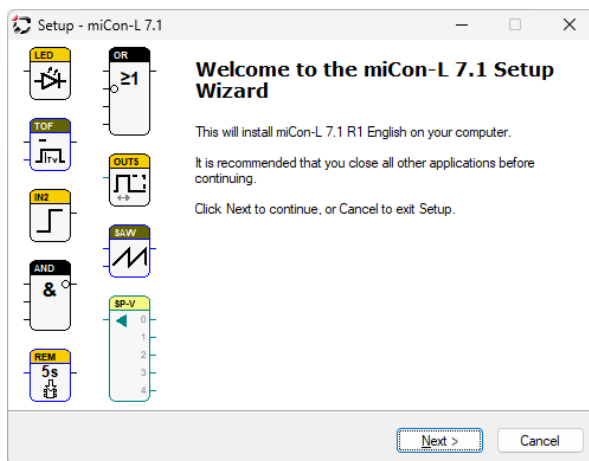


Figure 4: Start page of the installation

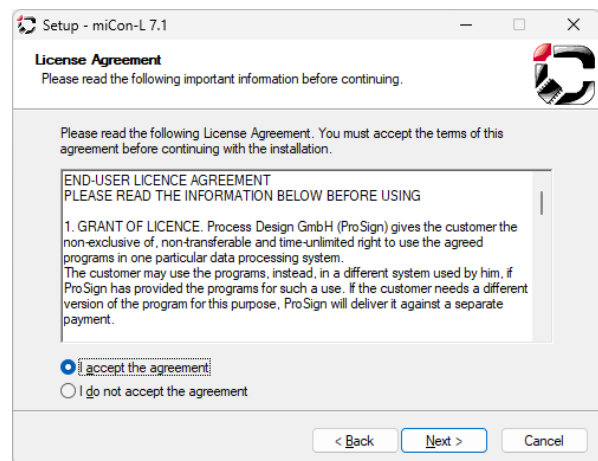


Figure 5: License agreement

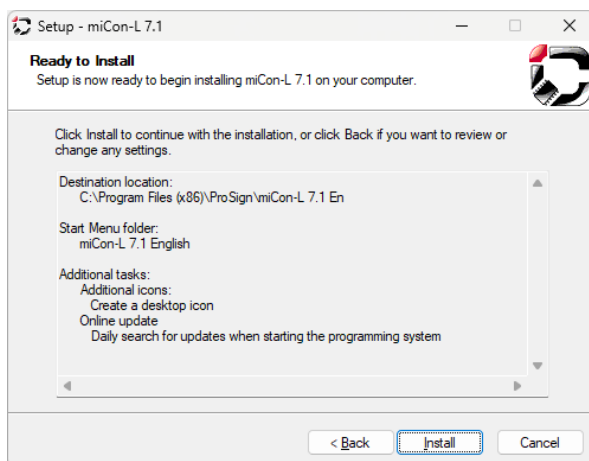


Figure 6: Summary

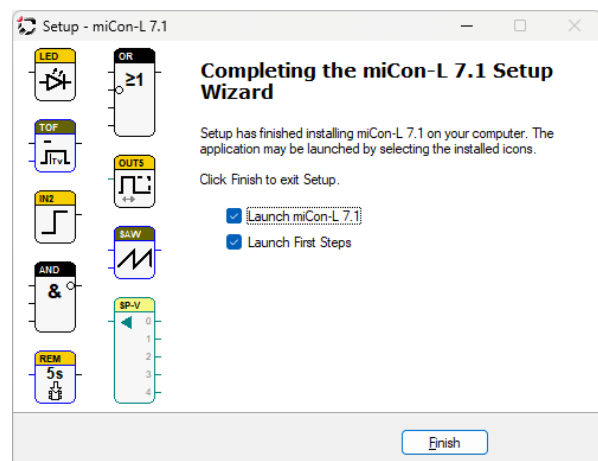


Figure 7: Completion of the installation

2.4 Start software

The software miCon-L can be opened via the Start menu entry.

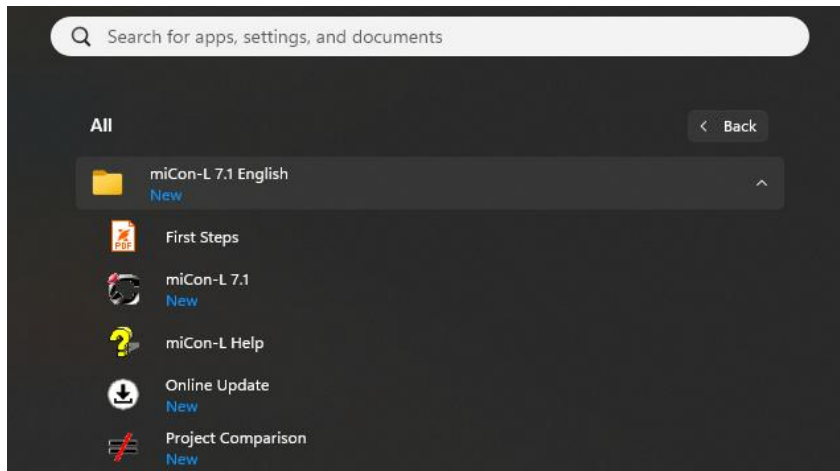


Figure 8: miCon-L in the Start menu entry

You can also simply start miCon-L via the desktop icon.



Figure 9: miCon-L on the desktop

If neither a Start menu entry nor a desktop icon was created during the installation, miCon-L must be opened directly via the file system.



.\miCon-L\BIN\miCon-L.exe

After starting miCon-L, an existing (sample) project can be opened or a new project can be created.

The creation of a new project is supported by a dialog in which project templates for the supported controller types are offered. For some controllers, there are also quick start templates that already contain I/O function blocks.

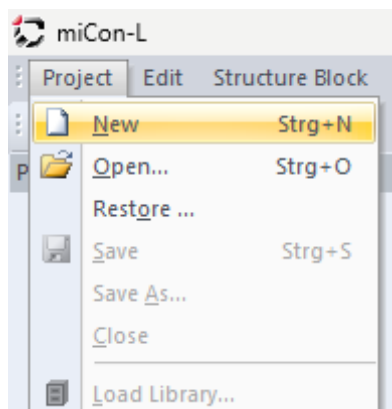


Figure 10: Menu: Project → New

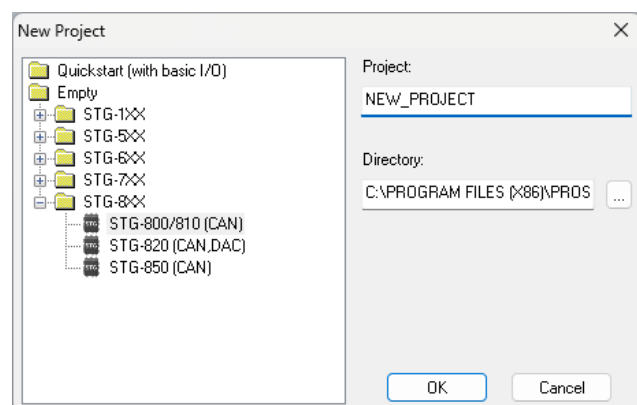


Figure 11: Dialog: New project

2.5 Connect mini-PLC to the computer

In order for the mini-PLC to be found by the computer, it must be correctly supplied with power and the appropriate communication cable must be correctly connected.

The power supply for the mini-PLC is designed for 7V - 32V DC voltage to support 12V/24V systems.

The following communication cables are used depending on the controller:

VK-10 (USB/RS232)	FTDI chip	STG-500
VK-12 (USB)	FTDI chip	STG-115/600
VK-16 (USB/TTL232)	FTDI chip	STG-550/570/580/650/680/700/8XX
VK-46 (USB/(TTL/RS)232)	ST Chip	STG-500/550/570/580/650/680/700/8XX

The VK-46 is a new cable that combines the function of VK-10 and VK-16.


Communication always takes place via a serial interface, which is implemented as a native RS232 or TTL232 and is usually made available as a virtual COM port.

The driver installation for the virtual COM port is carried out automatically by Windows.

In exceptional cases, the installation must be carried out manually.


Setup for FTDI chip: .\miCon-L\SETUP\USBdriver

Setup for ST chip: [STSW-LINK009](#) (Download from ST-Website, registration required)

After setting up the communication link, the correct COM port parameter must be set in the miCon-L project ().

→ **COM64 is the maximum COM port!**

The set COM port is automatically used when an attempt is made to connect to the control unit, e.g.

to perform a download ().

If the project has already been loaded into the controller, online observation can be started or ended directly via the Online/Offline symbol

( / ).

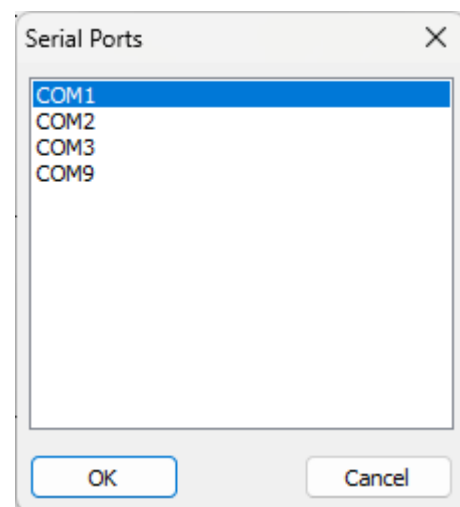




Figure 12: COM port configuration

Projects can also be simulated independently of the hardware.

In this case, miCon-L provides a PC target system to run the program even without real hardware. The simulation can be started () or stopped () via an icon in the toolbar.

2.6 Example with STG-600

First follow the instructions in chapter [2.4 Start software](#) and create a new project for the STG-600. The controller must then be correctly connected to the computer. To do this, follow the instructions in chapter [2.5 Connect mini-PLC to the computer](#).

In this example, the mini-PLC is connected to a potentiometer, a button and 3 LEDs.

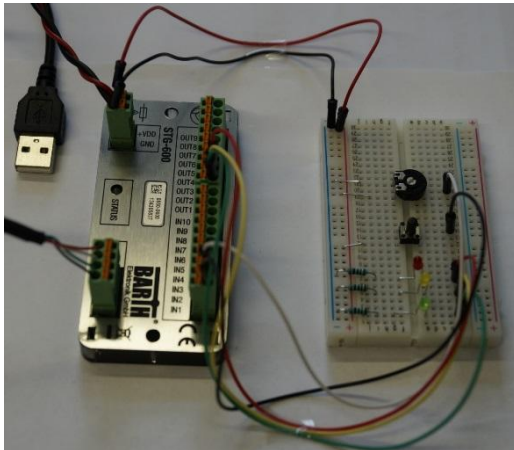


Figure 13: Experimental setup (STG-600)

The inputs and outputs on the hardware are assigned as follows:

IN1	Button (push-button)
IN3	Potentiometer
OUT1	Green LED
OUT2	Yellow LED
OUT3	Red LED

When the button (IN1) is pressed, all 3 LEDs should be switched on. The voltage at the middle tap of the potentiometer should simply be read in and visualized.

Step 1: Programming access to the potentiometer:

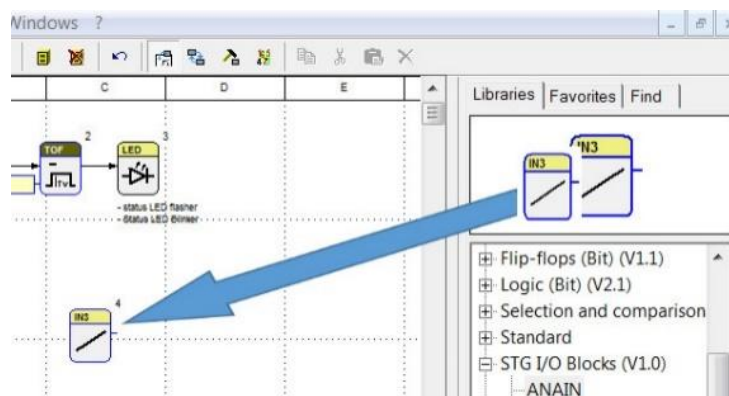


Figure 14: Insert block "Analog input" (ANAIN)

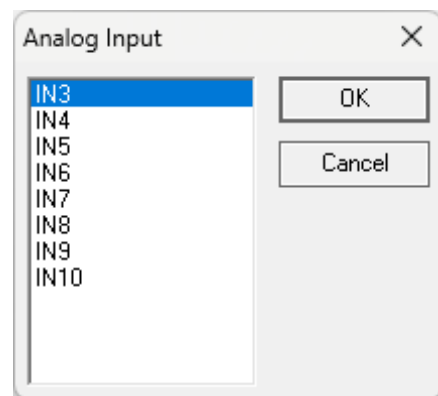


Figure 15: Parameterization to IN3

To do this, the "Analog input" function block is dragged and dropped directly from the library tree (or from the block window) onto the worksheet. The correct analog input is selected in the parameter dialog that opens.

The “Display numeric” visualization block is then added, which enables online monitoring of the connected analog signal.

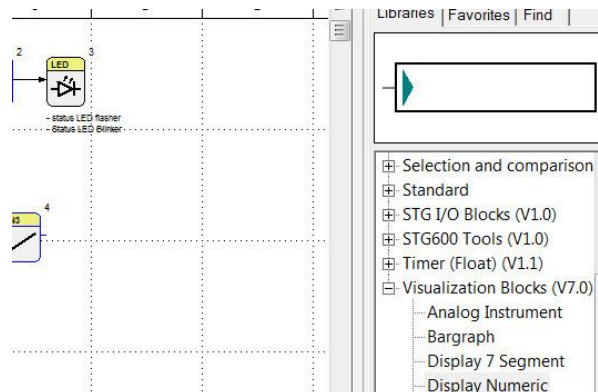


Figure 16: Insert block “Display numeric”

The connection between input and output will be generated by left-click on the output of the analog input, then click again on the input of the numeric display.

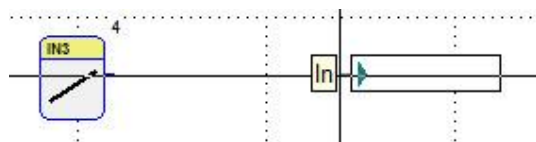


Figure 17: Connect display with signal

Step 2: Programming the LEDs:

In the following, 4 more function blocks must be added to the worksheet, parameterized and connected to each other.

First the digital input IN1, to which the push-button is connected:

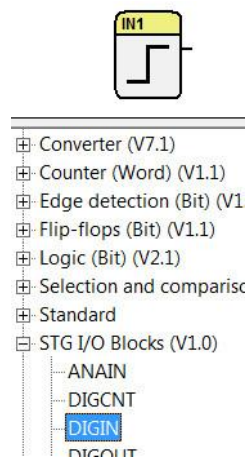


Figure 18: Select block “Digital input” (DIGIN)

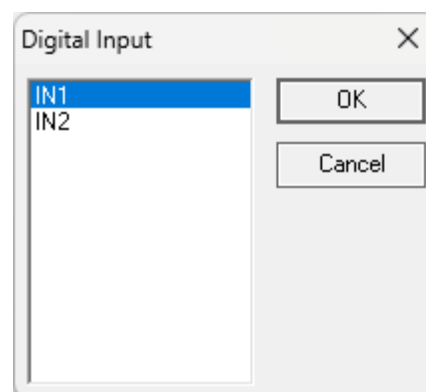


Figure 19: Parameterization to IN1

The 3 function blocks for the digital outputs (OUT1/2/3), to which the LEDs are connected, are then inserted and connected directly to IN1 (push-button).

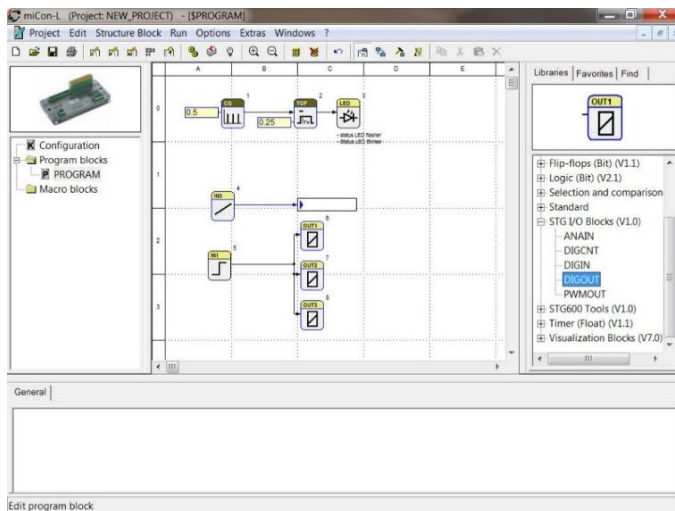


Figure 20: Final program

The final program is transferred to the mini-PLC via Download (📶).

Pressing the “Download” symbol automatically exits Edit Mode, switches to Run Mode, establishes a connection to the target system and transfers the project to the target system.

In Online Mode, the two flashing function blocks switch-off delay and status LED can be recognized. A green symbol indicates the HIGH (TRUE) status. When turning the potentiometer, the changing voltage values can be seen in the numerical display.

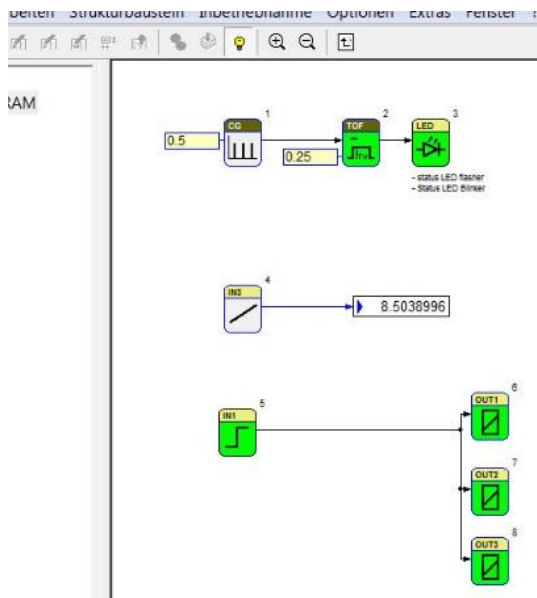


Figure 21: miCon-L in Online Mode

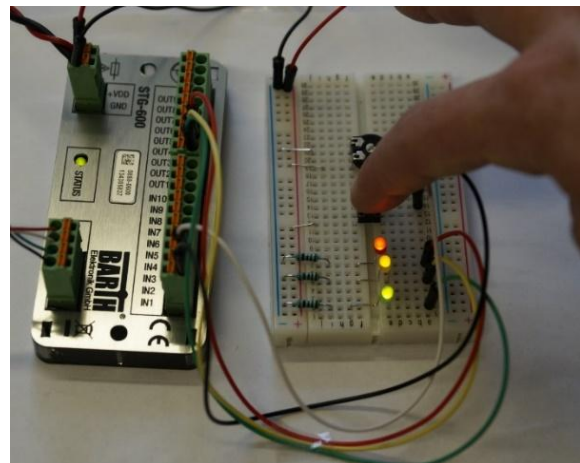


Figure 22: Experimental setup (STG-600)

When the button is pressed, the three physical LEDs and the corresponding blocks in the miCon-L project light up.


3 Changer

3.1 What is different or new in miCon-L 7.1?

a) Installation

miCon-L 7.1 has a full installation, in contrast to the self-extracting archive at miCon-L 3.x. The installation registers the access rights required by miCon-L with the operating system (Windows), can create a Start menu entry and a desktop icon, but requires the basic right to install software.

b) Communication settings

miCon-L 7.1 no longer has StartMe.exe, which changes the COM port setting for some controllers. This has been standardized and now behaves in the same way as the STG-8XX, i.e. the COM port is set individually for each project. This can be done by the toolbar () or the Tools menu (see chapter [2.5 Connect mini-PLC to the computer](#)).

c) FirstSteps (this document)

The “First steps” document is no longer called up via StartMe.exe because it no longer exists, but directly via the help menu in miCon-L. The First Steps now contain significantly more information and are also intended as a central document to link different sources of information and provide a better overall view.

d) Supported controllers

miCon-L 7.1 supports fewer controllers than miCon-L 3.8.1 directly. This means that new projects can no longer be created for some controllers and the corresponding example projects no longer exist. This applies to STG-32, STG-606, STG-860 and all WCU-XXX. – However, the import is supported for all old projects, and therefore indirectly also for every old controller (see next chapter [3.2 Project import](#)).

e) Improvements and innovations in miCon-L 7.1

- During online parameterization, the number of control elements no longer has any influence on the reaction speed, e.g. when pressing a button.
- In Edit Mode, blocks (function blocks and structure blocks) can now be inserted directly from the respective tree using drag-and-drop.
- The positioning of the individual windows (e.g. project tree, library tree, messages) can be done much more freely and offers significantly more flexibility.
- Possible block sequence problems are much easier to find by highlighting the corresponding block number with a red circle. This function is deactivated by default and can be activated manually. To do this, the leading semicolon must be removed in line 36 in the file `.\BIN\icon.ini`:
`;checkblocksequence=display|dontprint`
- There are some new browsers to simplify the search for signals, structure block connections, errors, texts, blocks or entire libraries and macros.
- There is also a search field that can be activated individually for the project tree and library tree.
- Handling of structure blocks (macros and program blocks)
 - The connections of structure blocks (Input/Output) are automatically placed in the macro design.
 - The macro design can now also be scrolled (important for high zoom levels).
 - The data types of Inputs/Outputs can now be easily changed via the parameter dialog (even after insertion on the worksheet).
 - More file formats for structure block images: BMP, EMF, PNG, JPG, TIF and GIF
 - There is an additional info text with up to 200 characters on the symbol or in the project tree.
- Variable management has been improved so that variables can be grouped in folders. Another new feature is that variables can also be instantiated.
- There is an additional standard function block "Rich text", with which formatted texts (color, font, ...) can be created (max. 30kBytes in RTF format). Text written in Microsoft Word, for example, can also be copied directly from the clipboard. (Images are not supported. Tables are only supported to a limited extent).
- There are 2 new control elements:
 - Link: Open a document or a web page
 - TODO mark: Generating a message, a warning or an error during the download
- Online update: Since miCon-L 7.1 the IDE can search for updates independently.
- Project comparison: Expert tool to find project differences.

3.2 Project import

Projects created with older miCon-L versions can simply be imported with the new version 7.1 and then used.

When opening each project, the IDE checks whether a project import is necessary and, if so, displays the dialog shown below.

During the import, a copy of the old project is automatically created. The copy is then adapted to the new miCon-L.

The directory for the copy can be changed in the import dialog. Furthermore, the details should be checked by type to ensure that the correct controller family has been recognized.

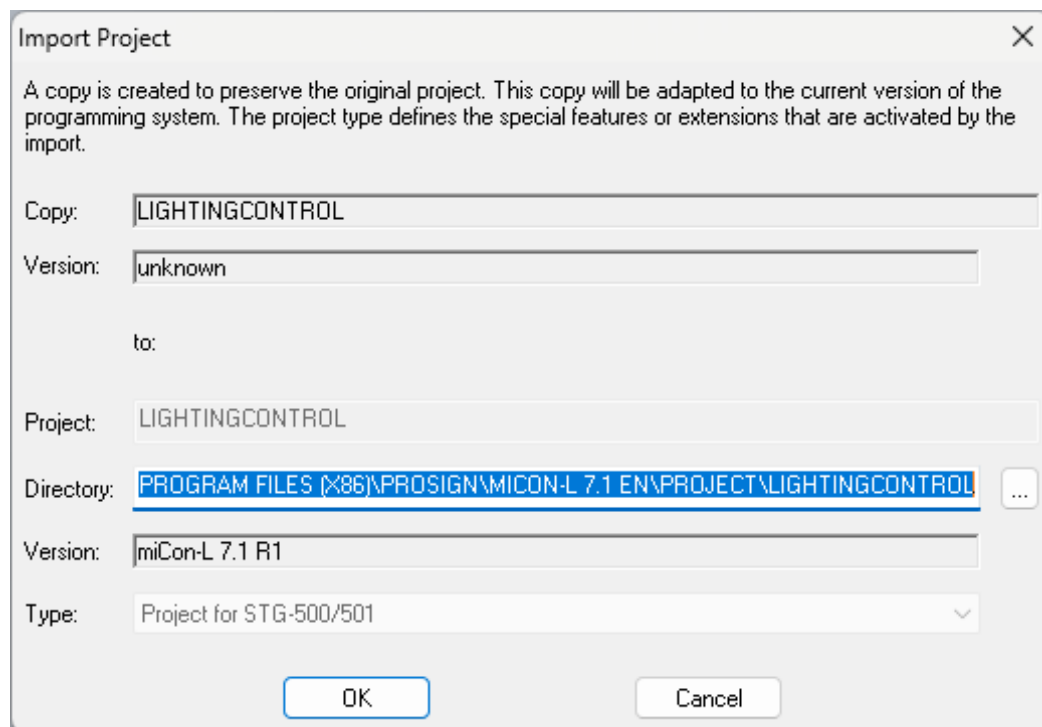


Figure 23: Dialog project import

Due to considerable differences between the various controllers, the project import is configured so that it automatically recognizes the project type and only creates projects of the same or compatible type, so that the user does not have to do much.

For projects whose project type cannot be determined automatically, STG-850 is used as the default project type.

4 More information

4.1 Where can I find more information?

Further information is available on the following websites:

- Company BARTH® Elektronik GmbH: www.barth-elektronik.com
- miCon-L product website: <https://barth-elektronik.com/en/miCon-L/>

Explanations of miCon-L are available in the “**Online help**” of the IDE. (use F1 key)

Explanations of the function blocks are available for each library (and each block) in a separate “**Block help**”. (call via the context menu of the function block)

This document (**FirstSteps**) is linked in the Start menu during installation and can be called up at any time via the “?” symbol in the miCon-L menu. It is also available online.

Sales Support: BARTH® Elektronik GmbH → office@barth-elektronik.de

Technical Support: BARTH® Elektronik GmbH → support@barth-elektronik.de

Customer feedback: BARTH® Elektronik GmbH → info@barth-elektronik.de

The topic of **examples** is dealt with in the **next chapter**.

4.2 Example projects

Example projects can be found in the miCon-L installation folder:

.\\miCon-L\\PROJECT\\SAMPLE_PROGS\\

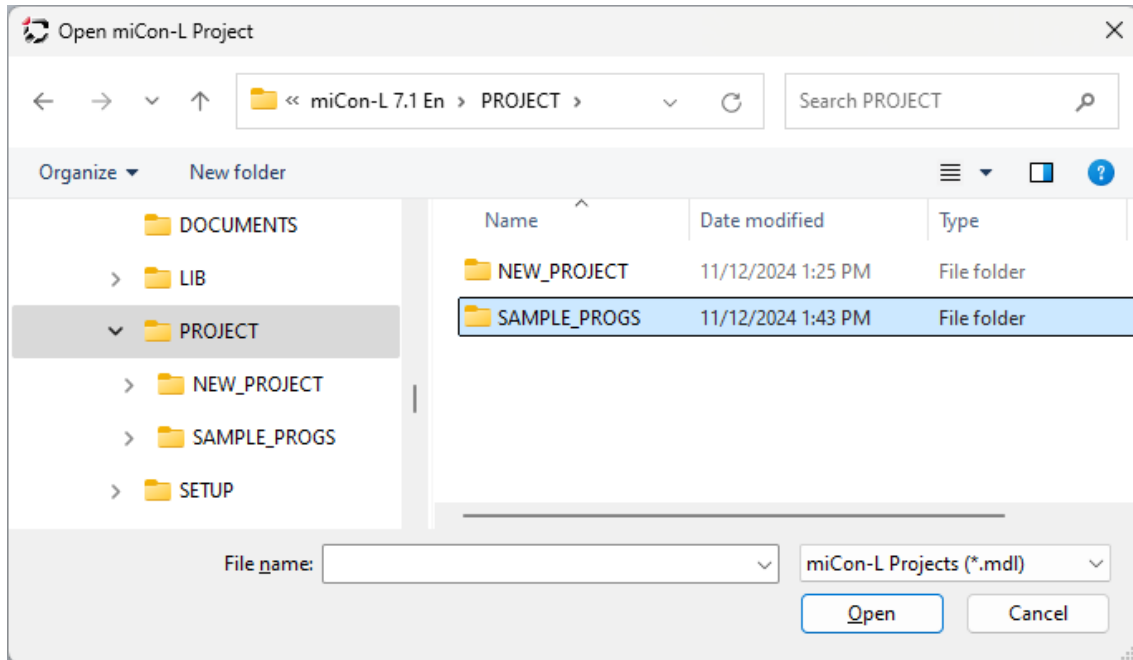


Figure 24: Dialog open project (SAMPLE_PROGS)

There are **general examples** that are mainly intended for the simulation as a target system, but can also be downloaded to all STG-8XX. – These general examples show basic principles and solutions for various small tasks.

There are also **examples** for each control system that only relate to the **specific blocks**.

4.3 Good to know / tips

a) Motivation

A deep understanding is not necessary for simple tasks and miCon-L offers a very quick and easy introduction to the PLC world.

If tasks become more complex or the requirements for real time become more stringent, and the development time is shorter, then a much deeper understanding at all levels is often necessary for the solution!

The necessary understanding concerns several areas:

- Functionality and operation of miCon-L (editor)
- Functionality and application of the various function blocks
- Properties of the runtime environment (range of functions, performance, behavior, ...)
- Properties of the hardware (interface behavior, memory properties, ...)
- Field buses, sensors, actuators and other connected devices
- Signal processing and control technology

The list can be continued depending on the task area, but should nevertheless illustrate that complex solutions can require considerable effort despite good tools.

ProSign would like to recommend some best practices:

- Personal basic attitude: Learning by doing → Use of the Simulation!
- Development in small steps → Breakdown into manageable subtasks
- Pay attention to the block order
- Separate input, processing and output, e.g. by encapsulating the processing as far as possible in macros with clear interfaces. (Variables as hidden interfaces should only be used in exceptional cases!)
- Self-explanatory programs:
 - Limiting the number of blocks on a worksheet (e.g. 30)
 - Useful names for macros, interfaces (Input/Output) and variables
 - Use text comments and rich text for documentation

The Simulation together with online observability offers very good conditions for developing and verifying partial programs at an early stage and in small steps.

b) Concepts of the function blocks and libraries

The blocks and libraries have **grown historically** and have been implemented by various developers. An attempt was always made to maintain a high level of stability.

Libraries group blocks of similar function classes. miCon-L also supports other groupings via the concept of favorites. **Favorites** can be created, edited, exported and imported in any project.

Arrangement of block inputs: Inputs are normally only located on the left-hand side, but can also be arranged at the bottom, e.g. as control signals.

Arrangement of block outputs: Outputs are normally only located on the right-hand side, but can also be located at the top, e.g. as visualization signals.

Logical processing for simple blocks:

[INPUT1] [OPERATION] [INPUT2] → [OUTPUT]

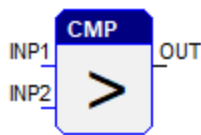


Figure 25: Comparator: greater

INP1 > INP2 → OUT

INPUT1	... INP1	(top left)
OPERATION	... >	(functions block symbol)
INPUT2	... INP2	(bottom left)
OUTPUT	... OUT	(top right)

Color of block border and pin indicates the datatype
(**blue**: FLOAT, **black**: BIT/BYTE)

c) Instantiation

For most applications, only global parameters and global variables are required. Only in exceptional cases are macro blocks used several times with different parameters in the project. Then the boundary conditions of the chapter “Classes and instances” in the miCon-L help must be observed.

There are a few function blocks that use instantiable (local) block parameters rather than global ones. This applies to the Control elements Switch / Button and Slider and also the I/O blocks with parameter dialog. In these cases, it is recommended that parameter changes are always made in the class (Edit Mode) AND the instance (Run Mode).

d) What makes everyday life with miCon-L easier?

- On questions, first look in the block help, the miCon-L help or the FirstSteps.
- Use copy and paste (Ctrl + C, Ctrl + V)
- Use export and import → Don't invent the wheel twice!
- The clearest possible requirements are particularly important in support cases.
- If projects are passed on (the entire project directory, please!), then packaged as a zip file. The RESTORE directory can be deleted from the archive.

4.4 How does the miCon-L technology package work?

The miCon-L technology package consists of the block diagram editor (miCon-L IDE on the PC) and various target systems (STG-XXX with miCon-L runtime environment).

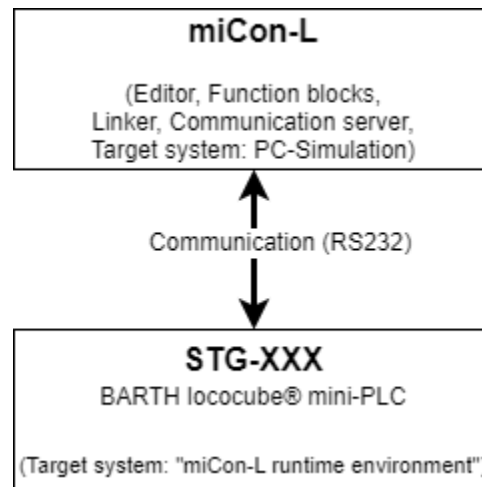


Figure 26: miCon-L context

The block diagram editor can be used in Edit Mode to create a graphical program (application design) that is organized hierarchically. Among other things, function blocks can be inserted, structure blocks (macros) can be created and inserted and connecting lines can be drawn.

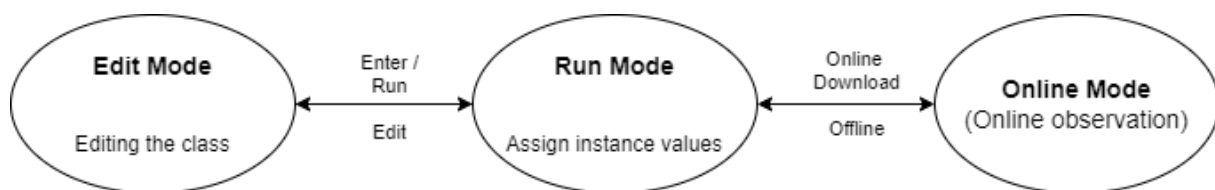


Figure 27: Modes of the editor

Instantiation and address assignment take place when switching to Run Mode. This means that memory is allocated for all data and parameters of the blocks and macros used.

The addresses of the respective memory are managed via logical names. Memories of the same data type are grouped into blocks, which are called resources in the context of miCon-L. (A resource is therefore a named memory block with a certain number of elements of the same size). Typical examples of logical addresses within resources are M1.0, M2.7, DF1, DF2, DW0, DW1, PF1, PF2.

The zero element (M0.0, DF0, DW0, PF0, ...) is always present and initialized with 0 and is used automatically if, for example, inputs of macros or blocks are not connected. All open inputs of a data type use the same zero element, i.e. changes to one input affect all of them. When downloading to the target system, the graphical program runs through two transformation steps.

In the first step, the hierarchical structure is broken up and converted into a large linear list and output in text form (Q file in the TEMP directory). This intermediate code is normal ASCII text and contains all block calls and all logical addresses used. – Based on this intermediate code, the IDE determines the total memory required for each resource.

A resource request list is sent to the target system, which responds with a valid device description file if enough memory (RAM, flash) is available, otherwise with an error message. – The device description file contains the physical addresses of the device functions and the physical addresses for the various memory areas.

In the second step, the intermediate code (Q file) is translated using the linker and the device description file. The logical addresses are exchanged for physical addresses. The result of the linking is a module connection list (MVL), which is available in binary form.

This binary MVL and the parameters from the project are loaded into the controller as so-called restart data and stored there in non-volatile memory.

The runtime environment can process this configuration data set (restart data). (The block diagram editor has no information about the content of the function blocks, as these are already programmed and compiled in the runtime environment. The editor therefore only generates a call list of the target functions and provides the respective parameters).

Processing in the target system is controlled by the Dispatcher, which reads the MVL and calls the corresponding target functions. After a target function has been processed, the next target function (the next function block) is called.

The MVL is processed cyclically with the set task cycle time. – The last function block in the MVL informs the Dispatcher that a complete run has been completed. The dispatcher then ends its processing loop. A new Dispatcher call is started via a timer module in time with the task cycle time.

miCon-L has read and write access to the memory in the controller via the communication connection.

Read access is required for online observation of the target system and write access for online parameterization.

Values that were changed in the volatile RAM during online parameterization are no longer available after the next start of the target system. A new download must be carried out to update the restart data.

4.5 Function block overview

miCon-L contains many different function block libraries, each of which contains function blocks of a specific type. – Due to their diversity, no controller supports all blocks, as there are many specific blocks (e.g. I/O blocks and interface blocks).

However, there are also “basic libraries”, without dependencies on physical interfaces, which are available for all controllers (Logic, Arithmetic, Converter, Timer, ...).

Many blocks are static in their application, i.e. the inputs and outputs have fixed data types and functions. However, there are also many blocks where, for example, the data type (Arithmetic, Converter, ...) or the number of connections (Logic, CAN send/receive) is only defined during use.

In principle, each function block has its own help, which can be opened in the library tree or on the worksheet via the context menu.

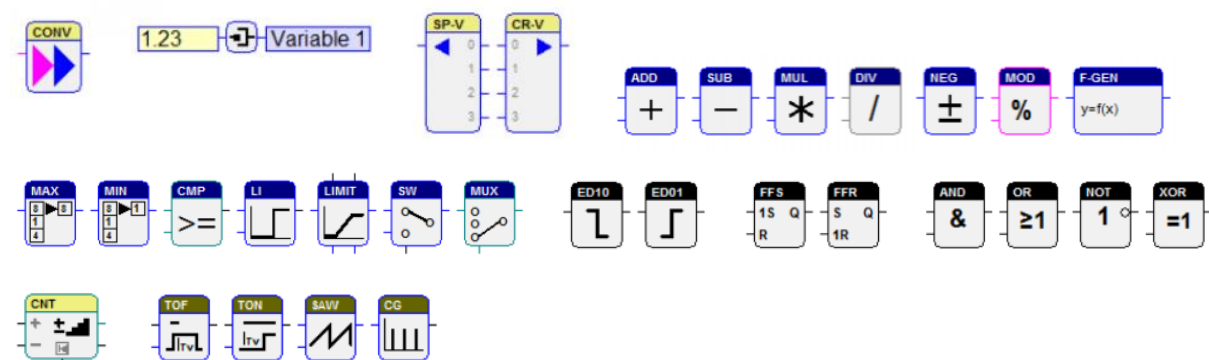
The following section provides a brief overview of the various block groups.

Standard blocks (Input, Output, Enable, Text Comment, Rich Text)



- Input and Output are needed to define the interfaces of macros
- Enable is needed, to control the call of macros at runtime
- Text Comment and Rich Text are used for better documentation of the program

Basic blocks (Converter, Arithmetic, Selection and comparison, Edge detection, Flip-flops, Logic, Counter, Timer)

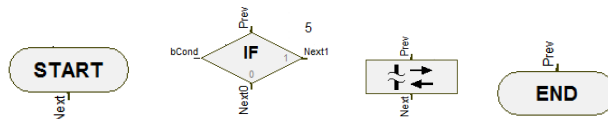


- The basic blocks are required in a wide range even for simple tasks.

Extended basic blocks (Numeric, Controller, Shift and rotate, Signal generators, Signal processing, Standard transmission terms)

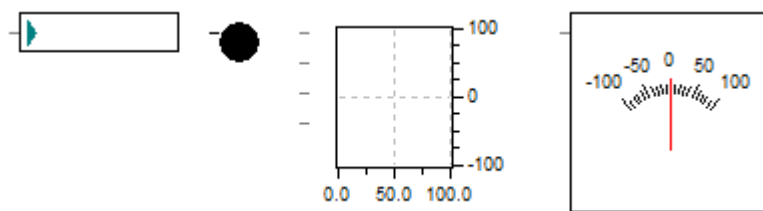
- These blocks provide additional math functions and signal processing functions and can be used for the STG-8XX.

Flow-Chart-Blocks (START, IF, STEP, END)



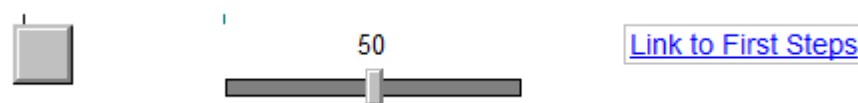
- The blocks can be used to implement flow-oriented programs.
→ Programs without a flow chart are data flow-oriented (i.e. strict block sequence).
- **Note:** Flow chart blocks cannot be easily combined with others.
→ See: Chapter **Introduction** in the block help for the library Flow-Chart-Blocks

Visualization Blocks (Display Numeric, LED, Trend, Analog Instrument, ...)



- The visualization blocks are used for online observation of the target system.
- The blocks do not have their own target function in the control system and only place a load on the system during active online communication.

Control elements (Switch / Button, Slider, ...)



- The control elements are used for online parameterization of the target system.
- The blocks do not have their own target function in the control system and only load the system when values are changed during active online communication.
- The Link block and the TODO-Mark block are not intended for online parameterization, but provide support for documentation, Run Mode and downloading.

Hardware specific blocks (I/O-Blocks, Communication interfaces, ...)

- Each controller has specific hardware interfaces that are directly supported by one or more dedicated libraries.

4.6 Glossary and data types

Table 1: Glossary - General

Term	Explanation
application / application design	Graphical program (user program) or simply “program”
block	Synonym for function block and structure block
block diagram	Graphical representation of the automation solution using function blocks and macros
communication server / online server	Software module that provides the communication driver on the PC side and connects the graphical editor to the target system so that data communication can take place.
computation time	Time required to process the entire program block (depending on the application and target system, in the order of a few microseconds to many (100) milliseconds)
connection line / (signal line)	Connection lines are used to exchange data between function blocks and structure blocks. Normally, only outputs can be connected to inputs. A connection line can also be replaced by a variable.
function block	In the miCon-L programming software, the function block is the central programming tool and represents the smallest usable unit of a user program.
function block library / library	Structuring elements for function blocks Libraries can be loaded, removed and replaced.
I/O	Inputs and outputs
IDE	Integrated Development Environment (is used synonymously with block diagram editor (i.e. miCon-L))
intermediate code (Q-file)	The intermediate code is a translation of the graphical hierarchical program into a linear representation in the form of an ASCII text file. e.g. .\miCon-L\TEMP\TMP001.Q
linker	Software module that translates the graphical program with logical addresses into physical addresses of the target system. The direct input for the linker is the intermediate code generated by the graphical editor.
macro block (macro)	Macro blocks contain sub-functions and are used for modularization. They are not to be understood as a classic subroutine, but generate the full code each time they are used.
PLC	Programmable Logic Controller → BARTH lococube® mini-PLC Is used to control or regulate a machine or system and is programmed on a digital basis (with miCon-L).
processing sequence / block sequence	Sequence in which blocks are processed.

	Normal programming is data flow-oriented and the sequence is specified via the block number (start at block number 1).
program block	A program block is inserted in the configuration level. miCon-L only supports a program block that has already been inserted. The actual functionality is programmed within the program block.
project	A (miCon-L) project is a collection of all “configurable” files that are necessary to fully support a solution for a control system. The actual application design (model) is stored in the MDL file. However, there are other files that contain, for example, the variables or the workspace settings (IWS file, iCon-L workspace settings).
real time error	Occurs when the processing time is greater than the task cycle time of the program. Short-term real time errors do not usually lead to permanent timing problems. - Frequent real time errors should be avoided in any case, as some blocks are dependent on an equidistant call. A reduction or avoidance is possible through a longer task cycle time or more efficient programs.
Simulation ("Software-PLC")	(Standard) simulation that provides a PC target system to run the application design even without a real hardware PLC. The simulation is ideal for quick and uncomplicated testing of the actual program logic.
structure block	generic term for macro and program blocks
target system (target) / runtime environment	In the context of miCon-L, a target system is a runtime environment that can process a miCon-L program. The target system contains all supported target functions.
target function	A target function is a programmed function in the target system that implements a function block functionality.
task cycle time (cycle time)	Time after which the program block is processed again. (The program is processed cyclically, with the task cycle time as the period duration).







Table 2: Glossary - Operating and data concept

Term	Explanation
block parameter	Parameters that are stored within the block and cannot be changed via connection attributes. These can usually be instantiated, e.g. selection of the “channel” for inputs and outputs or the display text for the control element switch/button.
class	A class of an attribute, block or structure block is created in Edit Mode and is a kind of template for the derived instances.
configuration	In the context of a miCon-L application, the configuration level is the top level. It is used for graphical configuration. In Run Mode, the task cycle time can be set here. – miCon-L only supports one task (program block).
connector attribute (attribute)	Designates connection parameters at signal inputs and variables at signal inputs and signal outputs.
context menu	Context menus are called up in the worksheets or in the project or library tree using the right mouse button. The commands contained refer to the object under the cursor.
data flow-oriented	Contrary to sequence-oriented The processing sequence is strictly defined by the block number and cannot be changed. A maximum of partial functions can be activated or deactivated (→ Macro with Enable block).
download	Download refers to the process of transferring the transformed program to the target system.
Edit Mode	Edit Mode is the basic state of the programming system. The main editing of the project takes place in this state.
flow chart / program flow chart	Flow charts enable sequence and control flow programming.
global (non instantiable)	Contrary to local or instantiable Attributes and function block parameters that are global are always the same in Edit Mode and Run Mode.
GVR	Abbreviation for Global Variables and References GVRs are configured via an extra dialog and can provide variables with various properties. This information is saved in a separate project file and can be exported and imported.
import project	Process supported by the IDE for opening a project that was created with a different (older) version of the programming system.
instance	An instance is a copy/derivation of the class of a “block”. The instances are created when you first switch to Run Mode.

	The instance values (attributes and internal block parameters) are initialized with the class values.
local (instantiable)	Contrary to global or not instantiable An instantiable signal (attribute or block parameter) is not necessarily the same in all system states of the Block Diagram Editor. (The instance value may differ from the class value).
MCL (Module Connection List)	Module connection list Linear representation of the program as a data set
Offline	General description of the fact that there is no active communication with the target system.
Online (Mode)	System status in which miCon-L is actively communicating with the target system and exchanging data (at least the cycle counter, which is shown by the editor as a time value).
Online communication	Data exchange that takes place in Online Mode between miCon-L and the target system. A high communication load can have a strong influence on the real time behavior of "weak" target systems.
Online Observation	Online Observation is only possible in Online Mode. Data is read from the target system. This includes the current cycle counter and the data for all visualization blocks present on the worksheet. Some normal function blocks (flip-flops, logic blocks, ...) also visualize their output status.
Online parameterization	Online parameterization is only possible in Online Mode. Data is written to the target system (e.g. values on signal lines and parameters).
parameter	Parameters are block parameters or connection parameters. These can be either global or local (instantiable). They can be understood as classic constants.
parameter memory	Parameter memory is a separately treated RAM memory that is pre-initialized with the parameters from the program. During download, the values are stored in a non-volatile memory and restored from it when the system is started. Parameter memory is required for parameter attributes. Parameters should only be read from.
process-oriented / control flow-oriented	Contrary to data flow-oriented Programming is carried out with flow chart blocks. The control flow or sequence can be manipulated at runtime.
program memory	The program memory contains the Module Connection List and is placed directly in the flash on small microcontrollers. The size of the available program memory largely determines the maximum size of the

	program (in addition to data memory and processing time (CPU speed)).
Run Mode	Run Mode is a special mode for instantiating and generating the program. All memories (logical addresses) are provided in this mode. Instantiable or local attributes and block parameters can be assigned for each instance.
variable	Variables are connection attributes that can be either global or local (instantiable). Data variables can be read and written. Parameter variables should not be used.

Table 3: Basic data types

Basic data types	Value range	Memory [Byte] + Color	Derived resource	Example
BIT bool	TRUE / FALSE	1 	M .. Data B .. Parameter	M0.0, M1.0, M1.1 B1.0
BYTE bool	8x TRUE / FALSE (8x BIT)	8 	M .. Data B .. Parameter	M0, M1, M2 B1
UCHAR positive integer	0 ... 255	1 	UCHARDATA .. Data UCHARPARA .. Parameter	UCHARDATA0, UCHARDATA1 UCHARPARA1
WORD integer	$-2^{15} \dots 2^{15} - 1$ -32768 ... 32767	2 	DW .. Data PW .. Parameter	DW0, DW1 PW1
LONG integer	$-2^{31} \dots 2^{31} - 1$ -2.147.483.648 ... 2.147.483.647	4 	DL .. Data PL .. Parameter	DL0, DL1 PL1
FLOAT (IEEE754) floating point num.	$\pm 1.175494\text{e-}38 \dots$ $\pm 3.402823\text{e+}38$	4 	DF .. Data PF .. Parameter	DF0, DF1 PF1

5 Revision history

Table 4: Document history

Date	Version	Changes
02/2012	-	Creation (joint German and English version)
07/2015	-	Complete revision within the scope of miCon-L 3.3 Separation into independent German and English versions
05/2016	-	Minor adjustments within the scope of miCon-L 3.7.0
03/2018	-	Partial revision within the scope of miCon-L 3.8.0 Addition of the chapters “What is miCon-L?” and “Additional information, download sources and example projects”
10.06.2025	2.0.0	Complete revision within the scope of miCon-L 7.1 References (Deactivation of the miCon-L website and the forum)