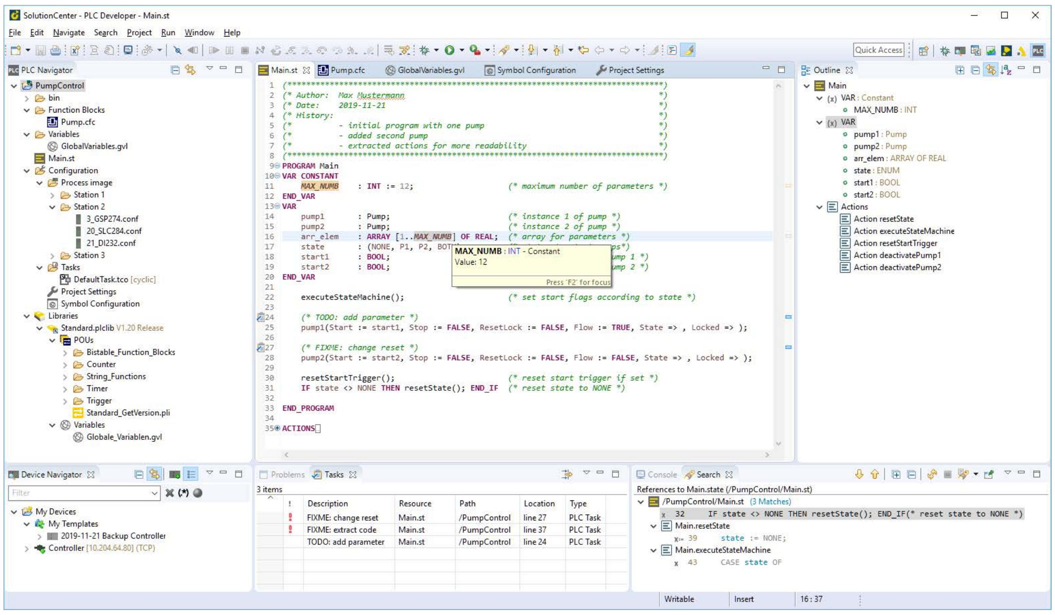
PLC Developer

Modern automisation systems solve complex tasks that go well beyond plant control. The procedural programming, in accordance with IEC 61131-3, allows multiple tasks to be solved in an efficient and structured way. This is where the PLC Developer supports the system programmer through all phases of software development.

Synergies in the SolutionCenter

The seamless integration of the PLC Developer into the SolutionCenter provides added value from the resulting synergies:

- By linking PLC projects in a Solution project with device configurations and templates, they can be combined to form an overall solution and managed as one.
- The version management function using GIT and SVN enables the user to trace code modifications easily and provides the basis for effective release management.
- The use of extensions, in the form of plugins, which can also be directly installed from the Eclipse Marketplace, simplifies engineering and programming.
- The SolutionCenter scripting API enables the use of modern software development methods such as continuous integration, model-driven development and test-driven development.



```
 1  (*************************************************************************)
 2  (* Author:  Max Mustermann                                               *)
 3  (* Date:    2019-11-21                                                   *)
 4  (* History:                                                              *)
 5  (*          - initial program with one pump                              *)
 6  (*          - added second pump                                          *)
 7  (*          - extracted actions for more readability                     *)
 8  (*************************************************************************)
 9  PROGRAM Main
10  VAR CONSTANT
11      MAX_NUMB    : INT := 12;                      (* maximum number of parameters *)
12  END_VAR
13  VAR
14      pump1       : Pump;                           (* instance 1 of pump *)
15      pump2       : Pump;                           (* instance 2 of pump *)
16      arr_elem    : ARRAY [1..MAX_NUMB] OF REAL;    (* array for parameters *)
17      state       : (NONE, P1, P2, BOT                                  ps*)
18      start1      : BOOL;                                               ump 1 *)
19      start2      : BOOL;                                               ump 2 *)
20  END_VAR
21
22      executeStateMachine();                        (* set start flags according to state *)
23
24      (* TODO: add parameter *)
25      pump1(Start := start1, Stop := FALSE, ResetLock := FALSE, Flow := TRUE, State => , Locked => );
26
27      (* FIXME: change reset *)
28      pump2(Start := start2, Stop := FALSE, ResetLock := FALSE, Flow := FALSE, State => , Locked => );
29
30      resetStartTrigger();                          (* reset start trigger if set *)
31      IF state <> NONE THEN resetState(); END_IF    (* reset state to NONE *)
32
33  END_PROGRAM
34
35  ACTIONS
```

```
MAX_NUMB : INT - Constant
Value: 12
                       Press 'F2' for focus
```

3 items

!	Description	Resource	Path	Location	Type
!	FIXME: change reset	Main.st	/PumpControl	line 27	PLC Task
!	FIXME: extract code	Main.st	/PumpControl	line 37	PLC Task
	TODO: add parameter	Main.st	/PumpControl	line 24	PLC Task

```
References to Main.state (/PumpControl/Main.st)
/PumpControl/Main.st (3 Matches)
    32      IF state <> NONE THEN resetState(); END_IF(* reset state to NONE *)
  Main.resetState
    39      state := NONE;
  Main.executeStateMachine
    43      CASE state OF
```

Writable | Insert | 16 : 37

Efficient programming and management

The clear and intuitive editors allow modern and efficient operation:

- The user-defined syntax highlighting increases the code readability in »Structured Text« (ST).
- The modern and clear design of the user interface for programming in »Continuous Function Chart« (CFC) also allows a good overview in large programs.
- Thanks to the fully available code navigation and call hierarchy, it is also possible to easily identify interrelationships in complex applications.
- Efficient programming is particularly enhanced by the context-sensitive autocomplete function and the predefined and extendable code and file templates.
- Project management in one workspace enables all changes to the source files to be stored in a local history. These can be called again at a later time irrespective of whether the project is version managed.
- An extensive validation function immediately notifies the programmer of programming errors when the program is edited. Quick fixes correct missing program sections such as variable declarations by adding them automatically.
- Information on interfaces and documentation of function blocks or variables can be called and displayed via tooltips without leaving the editor.
- The project-wide changing of identifiers can be carried out easily and thoroughly.
- When entering and saving the program, the notation of keywords and identifiers is corrected automatically.
- The CFC editors can also be fully operated via the keyboard. This increases efficiency when creating CFC diagrams.

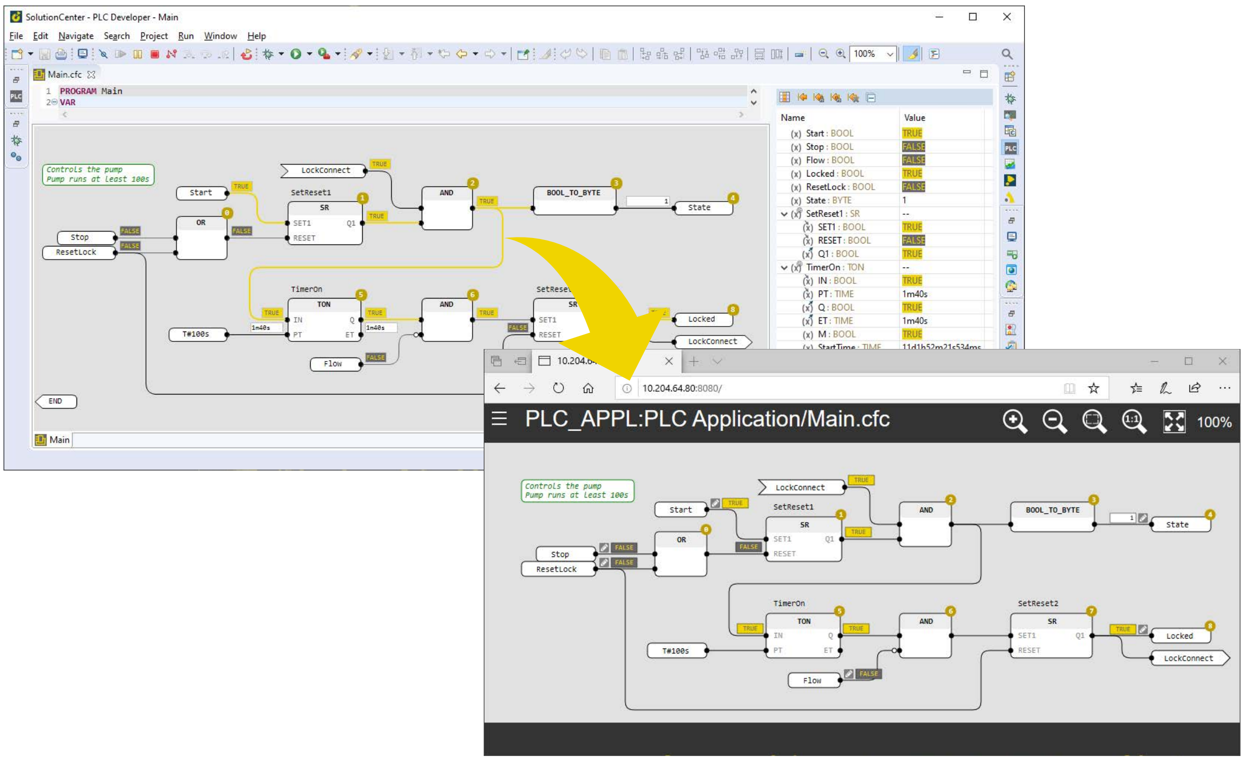
Modern operation through modular structure

The PLC Developer has an open, flexible and transparent design, and thus brings the engineering to a new level:

- All sections of the source code are saved in text form in the project in a defined file format, which opens the possibility for the user to generate modules and configurations. Recurring operating steps can thus be automated and copy/paste errors prevented.
- The structuring of the program sections can be selected as required and thus enables the application to have a modular design.
- In order to ensure a reproducible build, the libraries used are saved in the project. An update produces a clear display of the libraries present in a newer version.
- Creation and debugging of user-defined libraries in ST, CFC, C and C++ is possible throughout and particularly supports the reusability of code in other projects.
- Programming is carried out irrespective of the target platform, which only has to be selected at the time of execution. After compilation, the created application can be transferred directly to an M1 controller and run.
- Externally visible variables can be configured clearly in the symbol configuration.

Better quality through rapid troubleshooting
The debug framework of the PLC Developer is extremely useful, particularly with the development of new applications and with troubleshooting:

- After a debug session is started, the actual variable values can be monitored without stopping the application. This can be carried out via the clearly designed monitoring table in ST or the inline monitoring in CFC. The individual variables and structures of the application can also be combined in variable lists and monitored jointly, plotted in a graph and set.
- Activating the flow control makes it possible to follow the program flow without stopping the application.
- An application can be stopped at any position via a break point and then executed in steps. Break points can be managed jointly and activated or deactivated altogether.
- When the application is stopped, the stack frame is displayed by which it is also possible to navigate to the functions to be called.
- The dynamics of variables can be better assessed by highlighting variable values when they change.
- Values of Boolean variables are highlighted in color in order to better identify the current status of an application.
- Code modifications can be applied online without stopping the application. The PLC Developer detects during the transfer whether the online change can be carried out.
- It is also possible to debug several applications simultaneously. When an application with several tasks is debugged, the active debug task can be selected when connecting and changed during the debug session.
- PLC Insight is an automatically generated webMI visualization based on selected CFC blocks. It provides users with access to the application logic without the need for an engineering tool. The visualization is installed on the target controller together with the application. Along with the signal flow, values of exported variables can also be monitored and changed.


SolutionCenter - PLC Developer - Main
File Edit Navigate Search Project Run Window Help
Main.cfc
Controls the pump Pump runs at least 100s
Start
Stop
ResetLock
OR
LockConnect
SetReset1
SR
SET1
Q1
RESET
AND
BOOL_TO_BYTE
State
TimerOn
TON
IN
Q
PT
ET
T#100s
Flow
SetReset2
Locked
END
Main
10.204.64.80:8080/
PLC_APPL:PLC Application/Main.cfc
100%

```
PROGRAM Main
VAR
```

Name	Value
(x) Start : BOOL	TRUE
(x) Stop : BOOL	FALSE
(x) Flow : BOOL	FALSE
(x) Locked : BOOL	TRUE
(x) ResetLock : BOOL	FALSE
(x) State : BYTE	1
(x) SetReset1 : SR	--
(x) SET1 : BOOL	TRUE
(x) RESET : BOOL	FALSE
(x) Q1 : BOOL	TRUE
(x) TimerOn : TON	--
(x) IN : BOOL	TRUE
(x) PT : TIME	1m40s
(x) Q : BOOL	TRUE
(x) ET : TIME	1m40s
(x) M : BOOL	TRUE
(x) StartTime : TIME	11d1h52m21s534ms

PLC Developer	
General	
Integration	• Management of several PLC projects in one workspace • Management of all devices and PLC projects in one workspace • Linking of PLC projects in the Solution projects for joint management
Display	• Predefined perspectives with views for processing PLC projects • Clear, freely selectable arrangement of views and editors
Version management	• ZIT (local and remote) • SVN (local and remote)
Libraries	• Integration of libraries as copies with version comparison • Version display • Incorporation of standard libraries • Incorporation of PLC libraries • Incorporation of external PLC libraries (C/C++, MATLAB® /Simulink®)
Automation	• Execution of scripts in the SolutionCenter (JavaScript, Python) • Predefined modules for creating and transferring applications
Expandability	• Extensions (plugins) installable via Eclipse Marketplace • Use of user-created Eclipse plugins
Scalability	• Creation of the application independent of the target platform • Transfer and execution on all M200-CPUs
Structure	• Combination of projects in work sets • Folder structure freely selectable in the project • Full access to all source code files
Compatibility	• Compatible with M-PLC (compiler and runtime) • Import of M-PLC projects possible
Editors	
Languages	• Structured text (ST) • Continuous function chart (CFC) • Can be combined as required
Code navigation	• Finding references (in the project) • Call hierarchy (up to the task) • Caller hierarchy (hierarchy levels expandable) • Opening declaration (local and global)
Auto-complete	• Context-sensitive (declaration, call, program section) • Predefined source code templates • User-defined source code templates
Autocorrection	• Correction of identifier notation • Correction of keyword notation
Syntax highlighting	• Freely configurable • Separate for structured text, global variable lists, data types and hardware configuration
Quick fixes	• Declaration of variables and function block instances • Creation of missing elements (connection marks, jumps and jump labels) • Updating of the function block interface
Version control	• Comparison with locally stored versions (local history) • Comparison with source code from SVN or GIT
Changes	• Project-wide rename of identifiers
Information	• Context-sensitive information in the tooltip on identifier (ST) • Context-sensitive information on function block and pin (CFC)
Keyboard operation	• Execution of most frequently used actions with keyboard shortcuts • Programming possible also in CFC via keyboard
Validation	• Syntactical and semantic checking of the source code on typing • Display of problems directly in the source code and in the error view

PLC Developer	
Configuration	
Symbol configuration	• Visibility of the local and global variables • Access to local and global variables • Export of collective, structure and array entries • Inheritable configurations
Global variable lists	• Declaration of global variables • Declaration of global constants • Declaration of global retain variables
Variable configuration	• Assignment of variables to physical hardware addresses
Task configuration	• Up to 16 tasks configurable • Configuration of called programs • Configuration of task type (cyclic, sync, event/error interrupt, free wheeling) • PTP synchronization • Watchdog timeout
Process image	• Hardware import of online and offline device (selective) • Manual addition of hardware modules
Project settings	• Version number • Name of the module (m-file) • Error tolerance • Multicore suitability • Conflict handling with retain variables • Memory layout • Marker settings
Online Operation	
Installation	• Online change for code changes • Target controller permanently or dynamically selectable • Installation on target controller temporarily or permanently • Name of the module instance freely selectable
Initialization	• Selectable debug task on start (with multitasking projects)
Deploy configurations	• Several configurations per project possible • Combined execution of several configurations possible (launch groups) • Management of favorites
Debugging – Troubleshooting	
Stepwise processing	• Stopping of the application via breakpoint • Step into/over/return
Monitoring	• Variable values in the monitoring table in the editor (ST/CFC) • Variable values inline (only CFC) • Variable list for selected variables • Display of executed code via flow control
Display	• Changing the format of INT values (dec, hex, bin) • Current stack frame when the application is stopped
Libraries	• Internal libraries by incorporating the source codes (linked resources) • External libraries by connecting with the source codes (C/C++)
Manipulation	• Variable values can be set (one-off write) • Variable values can be forced (writing before each execution)
Visualization	• Generation of a complete webMI visualization from CFC blocks (PLC Insight) • Transfer of the visualization to the target together with the application • Monitoring and changing the exported variables via the visualization • Tracking the signal flow based on the connections between the function blocks