bachmann.
SolutionCenter

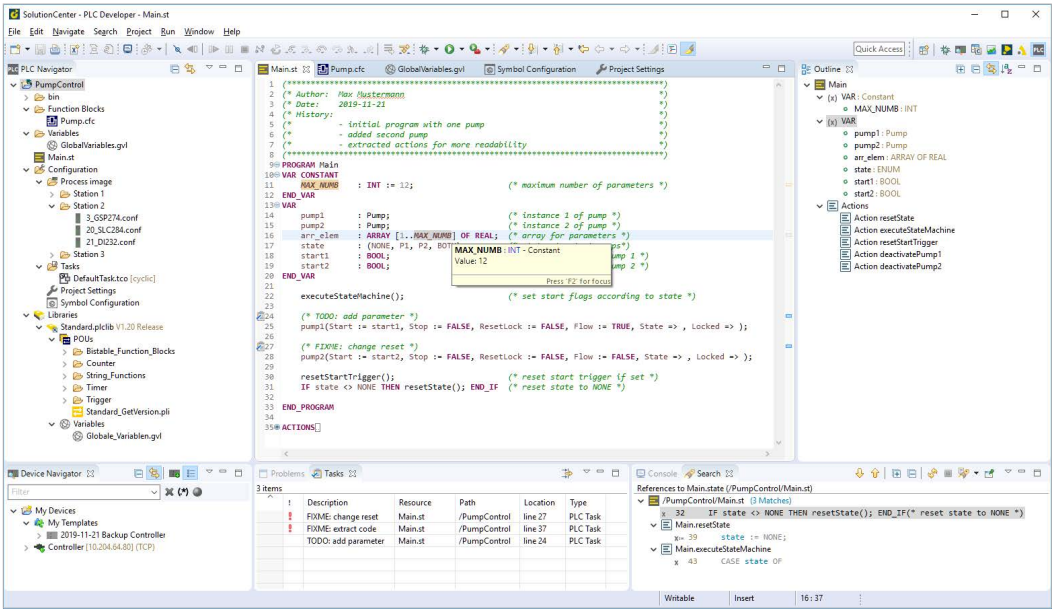
PLC Developer

Moderne Automatisierungssysteme lösen komplexe Aufgabenstellungen, die über das reine Steuern einer Anlage hinausgehen. Die prozedurale Programmierung nach IEC 61131-3 bietet die Möglichkeit verschiedenste Aufgabenstellungen effizient und strukturiert zu lösen. Dabei unterstützt der PLC Developer den Anlagenprogrammierer in allen Phasen der Software-Entwicklung.

Synergien im SolutionCenter

Durch die Integration des PLC Developer in das SolutionCenter können durch Synergien Mehrwerte geschaffen werden:

- PLC-Projekte können durch Verlinken in ein Solution-Projekt mit Device-Konfigurationen und Vorlagen zu einer Gesamtlösung verbunden und gemeinsam verwaltet werden.
- Die Versionsverwaltung mittels GIT und SVN ermöglicht dem Benutzer die einfache Nachverfolgbarkeit der Code-Änderungen und stellt die Basis für ein Release-Management dar.
- Die Verwendung von Erweiterungen in Form von Plugins, die auch direkt aus dem Eclipse Marketplace installiert werden können, erleichtern die Projektierung und Programmierung.
- Durch die Verwendung der SolutionCenter Scripting API können moderne Software-Entwicklungsmethoden wie Continuous Integration, Model-Driven-Development und Test-Driven-Development zum Einsatz gebracht werden.



Effizientes Programmieren und Verwalten
Die übersichtlichen und intuitiven Editoren erlauben eine moderne und effiziente Arbeitsweise:

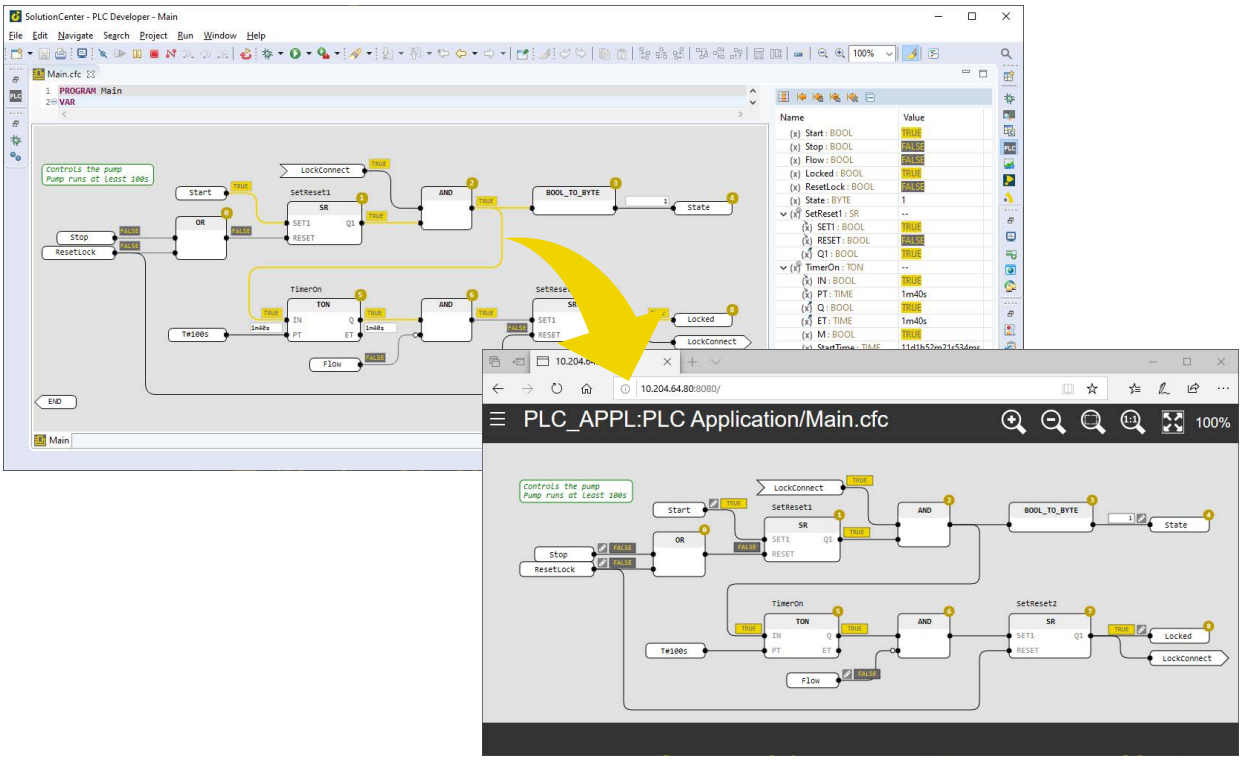
- Dabei erhöht das benutzerdefinierte Syntax-Highlighting die Lesbarkeit von Code der Sprache »Strukturierter Text« (ST).
- Das moderne und klare Design der Oberfläche für die Programmierung in »Continuous Function Chart« (CFC) erlaubt auch in großen Programmen eine gute Übersicht.
- Durch die durchgängig verfügbare Code-Navigation und Aufrufhierarchie können auch in komplexen Anwendungen Zusammenhänge leicht erkannt werden.
- Eine effiziente Programmierung wird speziell durch die kontextabhängige Autovervollständigung und die vordefinierten und erweiterbaren Code- und Dateivorlagen begünstigt.
- Durch die Verwaltung der Projekte in einem Arbeitsbereich werden alle Änderungen an den Source-Dateien in einem lokalen Protokoll gespeichert. Diese können zu einem späteren Zeitpunkt wieder abgerufen werden unabhängig davon ob das Projekt versionsverwaltet ist.
- Direkt bei der Eingabe wird der Programmierer durch eine umfangreiche Validierung auf Programmierfehler sofort hingewiesen. Über Schnellkorrekturen können fehlende Programmteile wie Variablendeklarationen automatisch erzeugt werden.
- Informationen zu Schnittstellen und Dokumentation von Funktionsblöcken oder Variablen können, ohne den Editor zu verlassen über Tooltips abgerufen und angezeigt werden.
- Das projektweite Ändern von Bezeichnern ist einfach und durchgängig möglich.
- Bei der Eingabe und beim Speichern wird die Schreibweise von Schlüsselworten und Bezeichnern automatisch korrigiert.
- Die Bedienung der CFC-Editoren ist durchgängig mittels Tastatur möglich. Beim Erstellen von CFC-Diagrammen kann so die Effizienz erhöht werden.

Moderne Arbeitsweise durch modularen Aufbau
Der PLC Developer ist offen, flexibel und transparent gestaltet, um das Engineering auf eine neue Ebene zu bringen:

- Alle Teile des Source-Codes werden in einem definierten Dateiformat in Textform im Projekt gespeichert, was dem Benutzer die Möglichkeit der Generierung von Bausteinen und Konfigurationen eröffnet. Wiederkehrende Arbeitsschritte können so automatisiert und Copy/Paste-Fehler vermieden werden.
- Die Strukturierung der Programmteile ist frei wählbar und ermöglicht somit einen modularen Aufbau der Applikation.
- Um einen reproduzierbaren Build zu gewährleisten, werden die verwendeten Bibliotheken im Projekt gespeichert. Beim Aktualisieren wird übersichtlich dargestellt, welche Bibliotheken in einer neueren Version vorliegen.
- Das Erstellen und Debuggen von benutzerdefinierten Bibliotheken in den Sprachen ST, CFC, C und C++ ist durchgängig möglich und dient vor allem der Wiederverwendbarkeit von Code in anderen Projekten.
- Die Programmierung erfolgt unabhängig von der Zielplattform, welche erst zum Ausführungszeitpunkt ausgewählt werden muss. Nach dem Kompilieren kann die erstellte Applikation direkt auf eine M1-Steuerung übertragen und ausgeführt werden.
- Die Konfiguration der nach außen sichtbaren Variablen ist übersichtlich in der Symbolkonfiguration möglich.

Bessere Qualität durch schnelle Fehlersuche
Gerade bei der Entwicklung neuer Applikationen und bei einer Fehlersuche ist das Debug-Framework des PLC Developer äußerst hilfreich:

- Nach dem Starten einer Debug-Session können die aktuellen Variablenwerte überwacht werden, ohne die Applikation anzuhalten. Dies ist über die übersichtliche Monitoring-Tabelle in ST oder das Inline-Monitoring in CFC möglich. Zusätzlich können einzelne Variablen und Strukturen der Applikation in Variablenlisten zusammengefasst und gemeinsam überwacht, in einem Diagramm aufgezeichnet und gesetzt werden.
- Durch Aktivierung der Ablaufkontrolle kann der Programmfluss verfolgt werden, ohne die Abarbeitung zu beeinflussen.
- Mittels Unterbrechungspunkt kann eine Applikation an einer beliebigen Stelle angehalten und in der Folge schrittweise ausgeführt werden. Unterbrechungspunkte können gemeinsam verwaltet und gesamthaft aktiviert bzw. deaktiviert werden.
- Bei angehaltener Applikation wird der Stack Frame angezeigt, über welchen auch zu den aufrufenden Funktionen navigiert werden kann.
- Durch die Hervorhebung von Variablenwerten bei Änderung kann die Dynamik von Variablen besser beurteilt werden.
- Werte von booleschen Variablen werden farblich hervorgehoben, um den aktuellen Status einer Applikation schneller erfassen zu können.
- Code-Änderungen können online übernommen werden, ohne die Applikation anzuhalten. Der PLC Developer erkennt bei der Übertragung, ob der Online-Change durchgeführt werden kann.
- Das gleichzeitige Debuggen mehrerer Applikationen ist möglich. Beim Debuggen einer Applikation mit mehreren Tasks kann der aktive Debug-Task beim Verbinden ausgewählt und während der Debug-Session gewechselt werden.
- Durch die Verwendung von PLC Insight, der automatisch generierten webMI-Visualisierung aus ausgewählten CFC-Bausteinen, kann auch ohne Engineering-Tool ein Blick auf die Applikationslogik geworfen werden. Die Visualisierung wird gemeinsam mit der Applikation auf der Zielsteuerung installiert. Neben dem Signalfluss können auch Werte exportierter Variablen überwacht und verändert werden.



PLC Developer	
Allgemein	
Integration	• Verwaltung mehrerer PLC-Projekte in einem Arbeitsbereich • Verwaltung aller Devices und PLC-Projekte in einem Arbeitsbereich • Verlinkung von PLC-Projekten in Solution-Projekte für die gemeinsame Verwaltung
Darstellung	• Vordefinierte Perspektive mit Ansichten zur Bearbeitung von PLC-Projekten • Übersichtliche, frei wählbare Anordnung der Ansichten und Editoren
Versionsverwaltung	• GIT (lokal und remote) • ZVN (lokal und remote)
Bibliotheken	• Einbindung der Bibliotheken als Kopie mit Versionsvergleich • Darstellung der Version • Einbinden von Standard-Bibliotheken • Einbinden interner PLC-Bibliotheken • Einbinden externe PLC-Bibliotheken (C/C++, MATLAB®/Simulink®)
Automatisierung	• Ausführen von Skripten im SolutionCenter (JavaScript, Python) • Vordefinierte Module für das Erstellen und Übertragen von Applikationen
Erweiterbarkeit	• Erweiterungen (Plugins) über Eclipse Marketplace installierbar • Verwendung selbst erstellter Eclipse Plugins
Skalierbarkeit	• Erstellen der Applikation erfolgt unabhängig von der Zielplattform • Übertragung und Ausführung auf allen M200-CPUs
Struktur	• Zusammenfassen von Projekten in Arbeitssets • Ordnerstruktur im Projekt frei wählbar • Voller Zugriff auf alle Quellcode-Dateien
Kompatibilität	• Kompatibel mit M-PLC (Kompiler und Laufzeit) • Import von M-PLC-Projekten möglich
Editoren	
Sprachen	• Strukturierter Text (ST) • Continuous Function Chart (CFC) • Beliebig kombinierbar
Code-Navigation	• Referenzen finden (im Projekt) • Zufruf-Hierarchie (bis zum Task) • Aufrufer-Hierarchie (Hierarchie-Ebenen erweiterbar) • Deklaration öffnen (lokal und global)
Autovervollständigung	• Kontextabhängig (Deklaration, Aufruf, Programmteil) • Vordefinierte Quellcode-Vorlagen • Benutzerdefinierte Quellcode-Vorlagen
Autokorrektur	• Korrektur der Schreibweise von Bezeichnern • Korrektur der Schreibweise von Schlüsselwörtern
Syntax Highlighting	• Frei konfigurierbar • Separat für Strukturierten Text, globale Variablenlisten, Datentypen und Hardware-Konfiguration
Schnellkorrekturen	• Deklaration von Variablen und Funktionsblock-Instanzen • Erstellen fehlender Elemente (Verbindungsmarken, Sprünge und Sprungmarken) • Aktualisierung der Funktionsblock-Schnittstelle
Änderungsverfolgung	• Vergleich mit lokal gespeicherten Versionen (lokales Protokoll) • Vergleich mit Quellcode aus SVN oder GIT
Änderungen	• Projektweites Ändern von Bezeichnern
Informationen	• Kontextabhängige Informationen im Tooltip auf Bezeichner (ST) • Kontextabhängige Informationen auf Funktionsblock und Pin (CFC)
Tastaturbedienbarkeit	• Ausführung der häufigsten Aktionen durch Tastenkürzel • Programmierung auch in CFC mittels Tastatur möglich
Validierung	• Zyntaktische und semantische Prüfung des Quellcodes bei Eingabe • Darstellung der Probleme direkt im Quellcode und in der Fehler-Ansicht

PLC Developer	
Konfiguration	
Symbolkonfiguration	• Sichtbarkeit der lokalen und globalen Variablen • Zugriff auf lokale und globale Variablen • Export der Sammel-, Struktur- und Arrayeinträge • Vererbbare Konfigurationen
Globale Variablenlisten	• Deklaration globaler Variablen • Deklaration globaler Konstanten • Deklaration globaler Retain-Variablen
Variablenkonfiguration	• Zuordnung von Variablen zu physikalischen Hardware-Adressen
Taskkonfiguration	• Bis zu 16 Tasks konfigurierbar • Konfiguration aufzurufender Programme • Konfiguration Task-Typ (Zyklisch, Sync, Event/Error Interrupt, freilaufend) • ZTP Synchronisation • Watchdog Timeout
Prozessabbild	• Hardware-Import von Online- und Offline-Device (selektiv) • Manuelles Hinzufügen von Hardware-Modulen
Projekteinstellungen	• Versionsnummer • Name des Moduls (m-Datei) • Fehlertoleranz • Multicore-Tauglichkeit • Konfliktbehandlung bei Retain-Variablen • Speicher-Layout • Merker-Einstellungen
Online gehen	
Installation	• Online-Change bei Code-Änderungen möglich • Zielsteuerung permanent oder dynamisch wählbar • Installation auf Zielsteuerung temporär oder permanent • Name der Modulinstanz frei wählbar
Initialisierung	• Debug-Task beim Start wählbar (bei Multitasking-Projekten)
Deploy-Konfigurationen	• Mehrere Konfigurationen pro Projekt möglich • Kombinierte Ausführung von mehreren Konfigurationen möglich (Launch Groups) • Verwaltung der Favoriten
Debuggen – Fehlersuche	
Schrittweise Abarbeitung	• Anhalten der Applikation mittels Breakpoint • Schritt hinein/über/zurück
Überwachen	• Variablenwerte in Monitoring-Tabelle im Editor (ST/CFC) • Variablenwerte inline (nur CFC) • Variablenliste für ausgewählte Variablen • Darstellung des ausgeführten Codes mittels Ablaufkontrolle
Darstellung	• Umschaltung des Formats von INT-Werten (dec, hex, bin) • Aktueller Stack-Frame bei angehaltener Applikation
Bibliotheken	• Interne Bibliotheken durch Einbinden des Quellcodes (verlinkte Ressourcen) • Externe Bibliotheken durch Verbinden mit den Quellcodes (C/C++)
Manipulieren	• Werte von Variablen können gesetzt werden (einmaliges Schreiben) • Werte von Variablen können forciert werden (Schreiben vor jeder Ausführung)
Visualisierung	• Generierung einer vollständigen webMI-Visualisierung aus CFC-Bausteinen (PLC Insight) • Übertragung der Visualisierung auf die Zielsteuerung zusammen mit der Applikation • Überwachen und Verändern der exportierten Variablen über die Visualisierung • Nachverfolgen des Signalflusses anhand der Verbindungen zwischen den Funktionsblöcken