Programmierleitfaden für S7-1200/1500

TIA Portal

https://support.industry.siemens.com/cs/ww/de/view/81318674

Siemens Industry Online Support

Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG ("Siemens"). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.
Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.
Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.
Ergänzend gelten die Siemens Nutzungsbedingungen (https://support.industry.siemens.com).

Securityhinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.
Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.
Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.
Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter: https://www.siemens.com/industrialsecurity.
Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter: https://www.siemens.com/industrialsecurity.

Inhaltsverzeichnis

1 Vorwort

Zielsetzung für die Entwicklung der neuen SIMATIC Steuerungsgeneration

- Ein Engineering-Framework für alle Automatisierungskomponenten (Steuerung, HMI, Antriebe, usw.)
- Einheitliche Programmierung
- Performancesteigerung
- Vollwertiger Befehlssatz für jede Sprache
- Vollständige symbolische Programmerstellung
- Datenhandling auch ohne Zeiger
- Wiederverwendbarkeit von erstellten Bausteinen

Ziel des Leitfadens

Die neue Steuerungsgeneration SIMATIC S7-1200 und S7-1500 weist eine zeitgemäße Systemarchitektur auf, und bietet zusammen mit dem TIA Portal neue und effiziente Möglichkeiten der Programmierung und Projektierung. Dabei stehen nicht mehr die Ressourcen der Steuerung (z.B. Datenablage im Speicher) im Vordergrund sondern die Automatisierungslösung selbst.

Dieses Dokument gibt Ihnen viele Empfehlungen und Hinweise zur optimalen Programmierung von S7-1200/1500 Steuerungen. Einige Unterschiede in der Systemarchitektur zu S7-300/400, sowie die damit verbundenen neuen Programmiermöglichkeiten werden verständlich erklärt. Dies hilft Ihnen, eine standardisierte und optimale Programmierung Ihrer Automatisierungslösungen zu erstellen.

Die beschriebenen Beispiele können universell auf den Steuerungen S7-1200 und S7-1500 eingesetzt werden.

Kerninhalte dieses Programmierleitfadens

Folgende Kernpunkte zum TIA Portal werden in diesem Dokument behandelt:

- S7-1200/1500 Innovationen
  - Programmiersprachen
  - Optimierte Bausteine
  - Datentypen und Anweisungen
- Empfehlungen zur allgemeinen Programmierung
  - Betriebssystem und Anwenderprogramm
  - Speicherkonzept
  - Symbolische Adressierung
  - Bibliotheken
- Empfehlungen zur hardwareunabhängigen Programmierung
- Empfehlungen zu STEP 7 Safety im TIA Portal
- Übersicht über die wichtigsten Empfehlungen

Vorteile und Nutzen

Durch die Anwendung dieser Empfehlungen und Hinweise ergibt sich eine Vielzahl an Vorteilen:

- Leistungsfähiges Anwenderprogramm
- Übersichtliche Programmstrukturen
- Intuitive und effektive Programmierlösungen

Weitere Informationen

Bei der Programmierung von SIMATIC Steuerungen hat der Programmierer die Aufgabe das Anwenderprogramm so übersichtlich und lesbar wie möglich zu gestalten. Jeder Anwender wendet eine eigene Strategie an, wie z. B. Variablen oder Bausteine benannt werden oder in welcher Art und Weise kommentiert wird. Durch die unterschiedlichen Philosophien der Programmierer entstehen sehr unterschiedliche Anwenderprogramme, die nur vom jeweiligen Ersteller interpretiert werden können.

Der Programmierstyleguide bietet Ihnen ein in sich abgestimmtes Regelwerk zur einheitlichen Programmierung. Diese Vorgaben beschreiben zum Beispiel eine einheitliche Vergabe von Variablen und Bausteinnamen bis hin zur übersichtlichen Programmierung in SCL.

Sie können diese Regeln und Empfehlungen frei verwenden und dienen Ihnen als ein Vorschlag (keine Norm oder Standard in der Programmierung) zur einheitlichen Programmierung.

Hinweis Den Programmierstyleguide für S7-1200 und S7-1500 finden Sie unter folgende Link:

https://support.industry.siemens.com/cs/ww/de/view/81318674

2 S7-1200/S7-1500 Innovationen

2.1 Einleitung

Generell ist die Programmierung der SIMATIC Steuerungen von S7-300/400 zu S7-1500 gleich geblieben. Es gibt die bekannten Programmiersprachen wie KOP, FUP; AWL, SCL oder Graph und Bausteine wie Organisationsbausteine (OBs), Funktionsbausteine (FBs), Funktionen (FCs) oder Datenbausteine (DBs). Erstellte S7-300/400 Programme können leicht auf S7-1500 umgesetzt werden und bestehende KOP, FUP und SCL Programme leicht auf S7-1200 Steuerungen.

Zusätzlich gibt es viele Innovationen, die Ihnen die Programmierung erleichtern und leistungsfähigen und speichersparenden Code ermöglichen.

Wir empfehlen Programme, die für S7-1200/1500 Steuerungen umgesetzt werden, nicht nur 1:1 umzusetzen, sondern auch auf die neuen Möglichkeiten hin zu überprüfen, und diese ggfs. einzusetzen. Oftmals hält sich der zusätzliche Aufwand in Grenzen, und Sie erhalten einen Programmcode, der z.B.

- für die neuen CPUs speicher- und laufzeitoptimal,
- leichter verständlich,
- und leichter wartbar ist.

Hinweis Informationen zur Migration von S7-300/S7-400 nach S7-1500 finden Sie unter folgenden Beitrag:

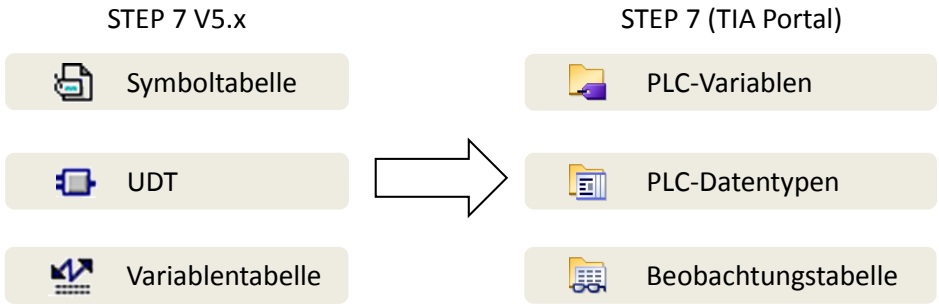
https://support.industry.siemens.com/cs/ww/de/view/109478811

2.2 Begriffe

Allgemeine Begriffe im TIA Portal

Manche Begriffe haben sich geändert, um Ihnen einen besseren Umgang mit dem TIA Portal zu ermöglichen.

Abbildung 2-1: Neue Begriffe im TIA Portal



Begriffe bei Variablen und Parametern

Wenn es um Variablen, Funktionen und Funktionsbausteine geht, gibt es viele Begriffe, die immer wieder unterschiedlich oder sogar falsch benutzt werden. Die folgende Abbildung stellt diese Begriffe klar.

Abbildung 2-2: Begriffe bei Variablen und Parametern

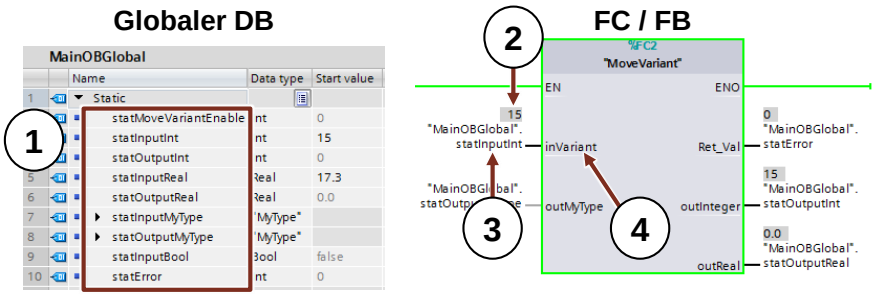


Tabelle 2-1: Begrffe bei Variablen und Parametern

	Begriff	Erklärung
1.	Variable	Variablen werden durch einen Namen/Identifier bezeichnet und belegen eine Adresse im Speicher in der Steuerung. Variablen werden immer mit einem bestimmten Datentyp (Bool, Integer, usw.) definiert: • PLC-Variablen • einzelne Variablen in Datenbausteinen • komplette Datenbausteine
2.	Variablenwert	Variablenwerte sind Werte, die in einer Variablen gespeichert sind (z.B. 15 als Wert einer Integer-Variablen)
3.	Aktualparameter	Aktualparameter sind Variablen, die an den Schnittstellen von Anweisungen, Funktionen und Funktionsbausteinen verschaltet sind.
4.	Formalparameter (Übergabeparameter, Bausteinparameter)	Formalparameter sind die Schnittstellenparameter von Anweisungen, Funktionen und Funktionsbausteinen (Input, Output, InOut und Ret_Val).

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Welche Beiträge für die Migration nach STEP 7 (TIA Portal) und WinCC (TIA Portal) stehen im Internet zur Verfügung?
https://support.industry.siemens.com/cs/ww/de/view/56314851

Welche Voraussetzungen müssen erfüllt sein, um ein STEP 7 V5.x Projekt in STEP 7 Professional (TIA Portal) zu migrieren?
https://support.industry.siemens.com/cs/ww/de/view/62100731

PLC-Migration zur S7-1500 mit STEP 7 (TIA Portal)
https://support.industry.siemens.com/cs/ww/de/view/67858106

Wie können Sie in STEP 7 (TIA Portal) für die S7-1200/S7-1500 effizient und performant programmieren?
https://support.industry.siemens.com/cs/ww/de/view/67582299

Warum ist in STEP 7 (TIA Portal) die Mischung von Registerpassing und expliziter Parameterübergabe bei der S7-1500 nicht möglich?
Unter anderem wird in diesem Beitrag die Migration von AWL Programmen auf S7-1500 beschrieben.
https://support.industry.siemens.com/cs/ww/de/view/67655405

2.3 Programmiersprachen

Zur Programmierung eines Anwenderprogramms stehen verschiedene Programmiersprachen zur Verfügung. Jede Sprache hat ihre eigenen Vorteile, die je nach Anwendung variabel eingesetzt werden können. Somit kann jeder Baustein im Anwenderprogramm in einer beliebigen Programmiersprache erstellt werden.

Tabelle 2-2: Programmiersprachen

Programmiersprache	S7-1200	S7-1500
Kontaktplan (KOP oder LAD)	ja	ja
Funktionsplan (FUP oder FBD)	ja	ja
Structured Control Language (SCL)	ja	ja
Graph	nein	ja
Anweisungsliste (AWL oder STL)	nein	ja

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

SIMATIC S7-1200 / S7-1500 Vergleichsliste für Programmiersprachen
https://support.industry.siemens.com/cs/ww/de/view/86630375

Was müssen Sie bei der Migration eines S7-SCL Programms in STEP 7 (TIA Portal) beachten?
https://support.industry.siemens.com/cs/ww/de/view/59784005

Welche Anweisungen sind in STEP 7 (TIA Portal) in einem SCL Programm nicht verwendbar?
https://support.industry.siemens.com/cs/ww/de/view/58002709

Wie können unter STEP 7 (TIA Portal) die Konstanten in einem S7-SCL-Programm definiert werden?
https://support.industry.siemens.com/cs/ww/de/view/52258437

2.4 Optimierter Maschinencode

TIA Portal und S7-1200/1500 ermöglichen eine optimierte Laufzeitperformance in jeder Programmiersprache. Alle Sprachen werden gleichermaßen direkt in Maschinencode kompiliert.

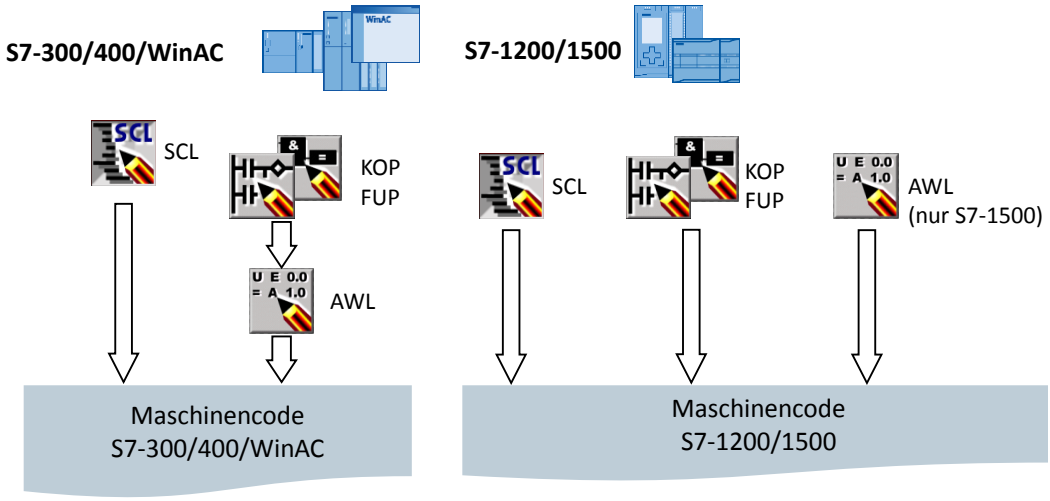
Vorteile

- Alle Programmiersprachen haben gleich hohe Leistung (bei gleichen Zugriffsarten)
- Keine Leistungsminderung durch zusätzliches Übersetzen mit Zwischenschritt über AWL

Eigenschaften

In folgender Abbildung wird der Unterschied bei der Übersetzung von S7-Programmen in Maschinencode dargestellt.

Abbildung 2-3: Maschinencodeerstellung mit S7-300/400/WinAC und S7-1200/1500

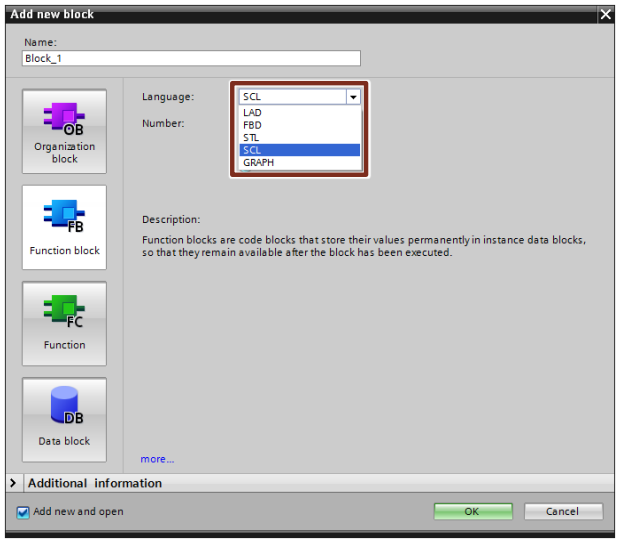


- Bei S7-300/400/WinAC Steuerungen werden KOP und FUP Programme zuerst in AWL übersetzt bevor Maschinencode erstellt wird.
- Bei S7-1200/1500 Steuerungen werden alle Programmiersprachen direkt in Maschinencode übersetzt.

2.5 Bausteinerstellung

Alle Bausteine wie OBs, FBs und FCs können direkt in der gewünschten Programmiersprache programmiert werden. Somit muss bei SCL Programmierung keine Quelle erstellt werden. Sie wählen nur den Baustein und SCL als Programmiersprache. Den Baustein können Sie dann direkt programmieren.

Abbildung 2-4: Dialog "Add new Block" ("Neuen Baustein hinzufügen)"



2.6 Optimierte Bausteine

S7-1200/1500 Steuerungen besitzen eine optimierte Datenablage. In optimierten Bausteinen sind alle Variablen gemäß ihrem Datentyp automatisch sortiert. Durch die Sortierung wird sichergestellt, dass Datenlücken zwischen den Variablen auf ein Minimum reduziert werden und die Variablen für den Prozessor zugriffsoptimiert abgelegt sind.

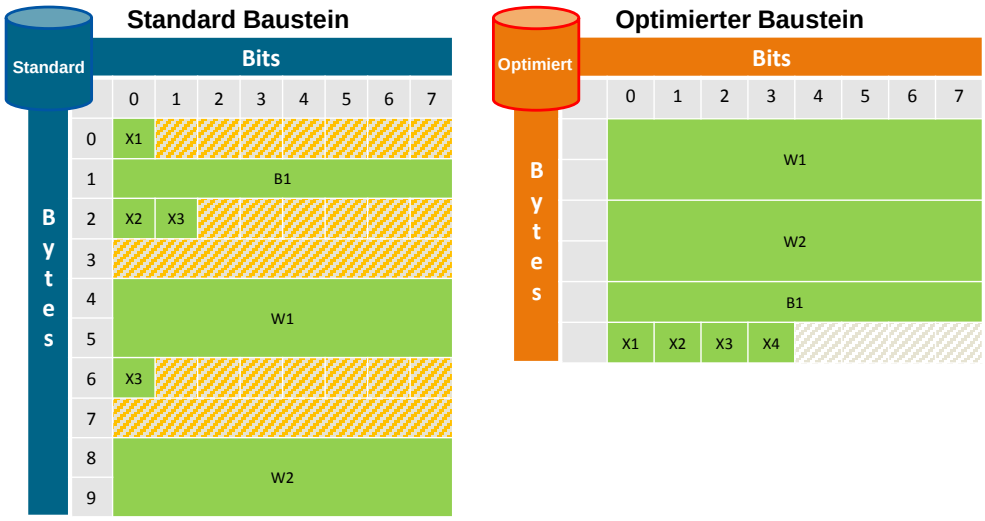
Nicht optimierte Bausteine sind in S7-1200/1500 Steuerungen nur aus Kompatibilitätsgründen vorhanden.

Vorteile

- Der Zugriff erfolgt immer schnellstmöglich, da die Dateiablage vom System optimiert wird und unabhängig von der Deklaration ist.
- Keine Gefahr von Inkonsistenzen durch fehlerhafte, absolute Zugriffe, da generell symbolisch zugegriffen wird.
- Deklarationsänderungen führen nicht zu Zugriffsfehlern, da z.B. HMI-Zugriffe symbolisch erfolgen.
- Einzelne Variablen können gezielt als remanent definiert werden.
- Keine Einstellungen im Instanzdatenbaustein notwendig. Es wird alles im zugeordneten FB eingestellt (z.B. Remanenz).
- Speicherreserven im Datenbaustein ermöglichen das Ändern ohne Verlust der Aktual Werte (siehe Kapitel 3.2.8 Laden ohne Reinitialisierung).

2.6.1 S7-1200: Aufbau von optimierten Bausteinen

Abbildung 2-5: Optimierte Bausteine bei S7-1200



Eigenschaften

- Es entstehen keine Datenlücken, da größere Variablen am Anfang des Bausteins und kleinere am Ende stehen.
- Es gibt ausschließlich den symbolischen Zugriff bei optimierten Bausteinen.

2.6.2 S7-1500: Aufbau von optimierten Bausteinen

Abbildung 2-6: Optimierte Bausteine bei S7-1500

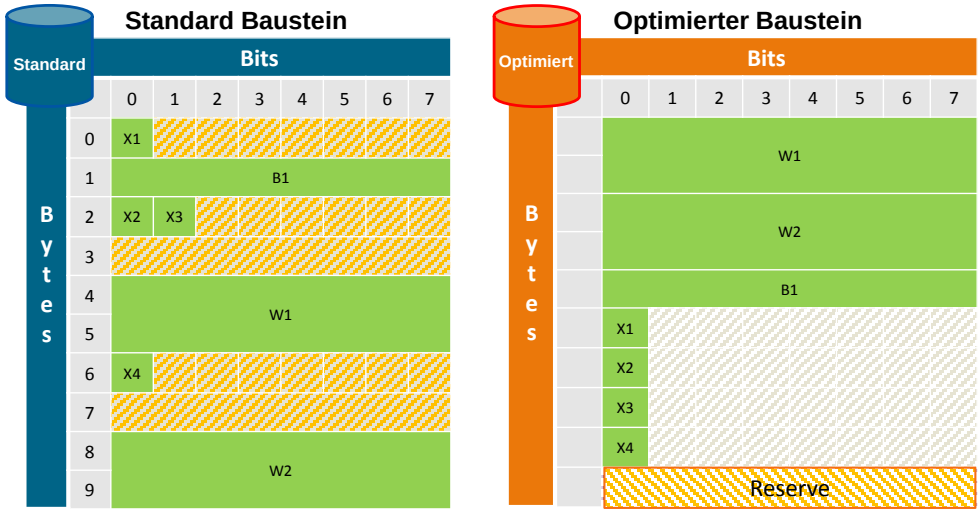
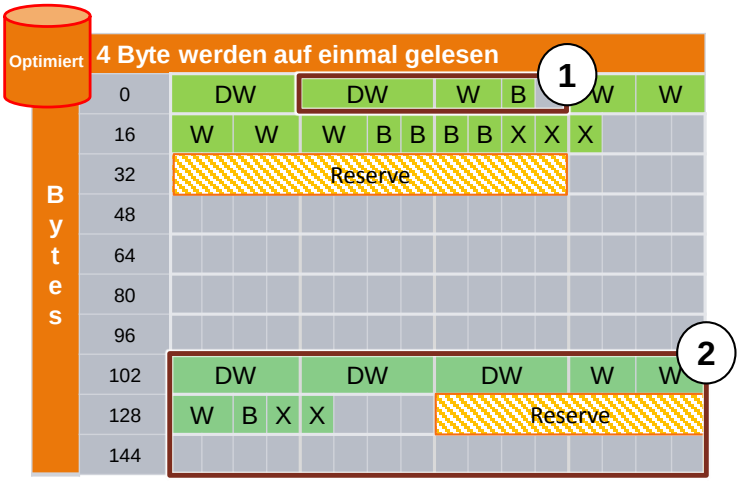


Abbildung 2-7: Speicherbelegung bei optimierten Bausteinen



1. Strukturen liegen separat und können damit als Block kopiert werden.
2. Remanente Daten liegen in einem separaten Bereich und können als Block kopiert werden.
   Bei Spannungsausfall werden diese Daten CPU-intern gespeichert. "MRES" setzt diese Daten auf die im Ladespeicher liegenden Startwerte zurück.

Eigenschaften

- Es entstehen keine Datenlücken, da größere Variablen am Anfang des Bausteins und kleinere am Ende stehen.
- Schnellerer Zugriff durch prozessoroptimale Ablage (Alle Variablen werden so abgelegt, dass der Prozessor der S7-1500 sie mit nur einem Maschinenbefehl direkt lesen oder schreiben kann).
- Boolesche Variablen werden zum schnellen Zugriff als Byte abgelegt. Somit muss die Steuerung den Zugriff nicht maskieren.

- Optimierte Bausteine haben eine Speicherreserve zum Nachladen im laufenden Betrieb (Siehe Kapitel 3.2.8 Laden ohne Reinitialisierung).
- Es gibt ausschließlich den symbolischen Zugriff bei optimierten Bausteinen.

2.6.3 Prozessoroptimale Datenablage bei S7-1500

Aus Kompatibilitätsgründen zu den ersten SIMATIC Steuerungen wurde das Prinzip der Datenablage "Big-Endian" in den S7-300/400 Steuerungen übernommen.

Die neue Steuerungsgeneration S7-1500 greift aufgrund der geänderten Prozessorarchitektur immer auf 4 Byte (32 Bit) in "Little-Endian"-Reihenfolge zu. Dadurch ergeben sich folgende systemseitige Eigenschaften.

Abbildung 2-8: Datenzugriff einer S7-1500 Steuerung

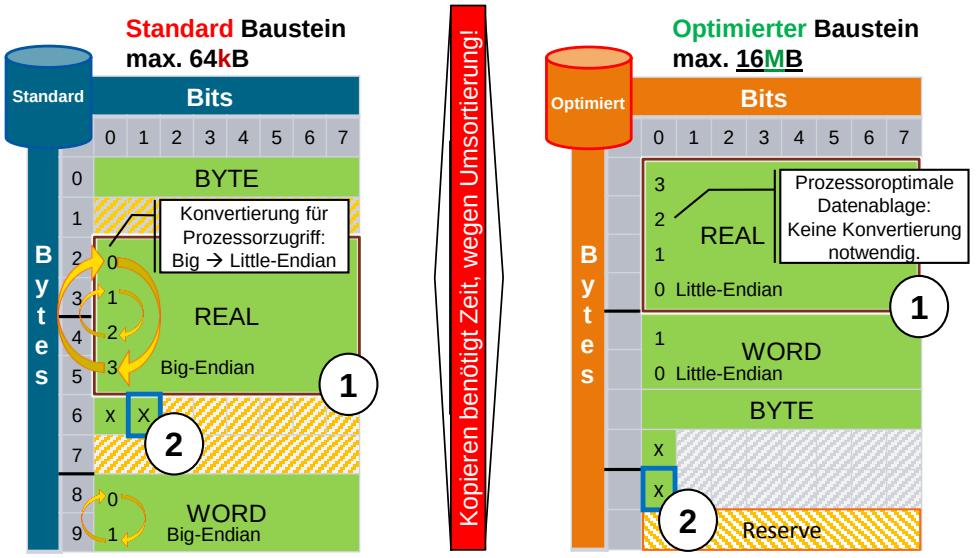


Tabelle 2-3: Datenzugriff einer S7-1500 Steuerung

	Standard Baustein	Optimierter Baustein
1.	Bei ungünstigem Offset benötigt die Steuerung 2x16 Bit-Zugriffe, um einen 4-Bytewert (z.B. REAL-Wert) zu lesen. Zusätzlich müssen die Bytes gedreht werden.	Die Steuerung legt die Variablen zugriffsoptimiert ab. Ein Zugriff erfolgt mit 32 Bit (REAL). Eine Drehung der Bytes ist nicht notwendig.
2.	Pro Bitzugriff wird das komplette Byte gelesen und maskiert. Das komplette Byte wird für jeden anderen Zugriff gesperrt.	Jedes Bit belegt ein Byte. Die Steuerung muss beim Zugriff das Byte nicht maskieren.
3.	Maximale Bausteingröße beträgt 64kB.	Maximale Bausteingröße, kann bis zu 16MB betragen.

Empfehlung

- Verwenden Sie generell nur optimierte Bausteine.
  - Sie benötigen keine absolute Adressierung und können immer mit symbolischen Daten (objektbezogen) adressieren. Indirekte Adressierung ist auch mit symbolischen Daten möglich (siehe Kapitel 3.6.2 Datentyp ARRAY und indirekte Feldzugriffe).
  - Die Abarbeitung von optimierten Bausteinen in der Steuerung ist wesentlich schneller als bei Standardbausteinen.
- Vermeiden Sie das Kopieren/Zuweisen von Daten zwischen optimierten und nicht optimierten Bausteinen. Die dafür erforderliche Umwandlung der Daten zwischen Quell- und Zielformat benötigt eine hohe Abarbeitungszeit.

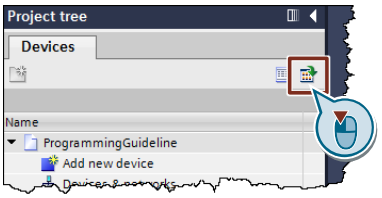
Beispiel: Optimierten Bausteinzugriff einstellen

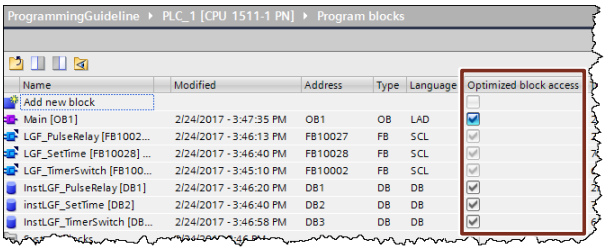
Standardmäßig ist der optimierte Bausteinzugriff bei allen neuerstellten Bausteinen bei S7-1200/1500 aktiviert. Der Bausteinzugriff kann bei OBs, FBs und Global-DBs eingestellt werden. Bei Instanz-DBs leitet sich die Einstellung vom jeweiligen FB ab.

Der Bausteinzugriff wird nicht automatisch umgestellt, wenn ein Baustein von einer S7-300/400 Steuerung auf eine S7-1200/1500 migriert wird. Sie können den Bausteinzugriff nachträglich auf "Optimierter Bausteinzugriff" ändern. Nach der Umstellung des Bausteinzugriffs, müssen Sie das Programm neu übersetzen. Falls Sie FBs auf "Optimierter Bausteinzugriff" ändern, werden die zugeordneten Instanzdatenbausteine automatisch aktualisiert.

Um den optimierten Bausteinzugriff einzustellen, folgen Sie folgenden Anweisungen.

Tabelle 2-4: Optimierten Bausteinzugriff einstellen

Schritt	Anweisung
1.	Klicken Sie in der Projektnavigation auf die Schaltfläche "Maximiert/minimiert die Übersicht" ("Maximizes/minimizes the Overview"). 
2.	Navigieren Sie zu "Programmbausteine" ("Program blocks").

Schritt	Anweisung
3.	Hier sehen Sie alle Bausteine im Programm und ob sie optimiert sind oder nicht. In dieser Übersicht kann der Status "Optimierter Bausteinzugriff" ("Optimized block access") komfortabel geändert werden.  Hinweis: Instanzdatenbausteine (hier "Function_block_1_DB") erben den Status "Optimiert " vom zugehörigen FB. Daher kann die Einstellung "Optimiert" nur am FB geändert werden. Nach dem Übersetzen des Projekts übernimmt der DB den Status abhängig vom zugehörigen FB.

Darstellung von optimierten und nichtoptimierten Bausteinen im TIA Portal

In den beiden folgenden Abbildungen sind die Unterschiede zwischen einem optimierten und einem nicht optimierten Instanz-DB erkennbar.

Bei einem Global-DB gibt es die gleichen Unterschiede.

Abbildung 2-9: optimierter Datenbaustein (ohne Offset)

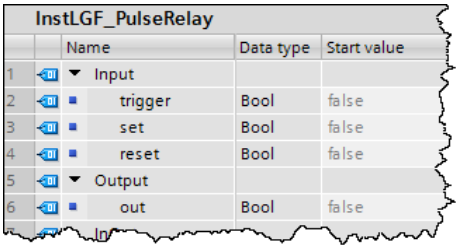


Abbildung 2-10: nicht optimierter Datenbaustein (mit Offset)

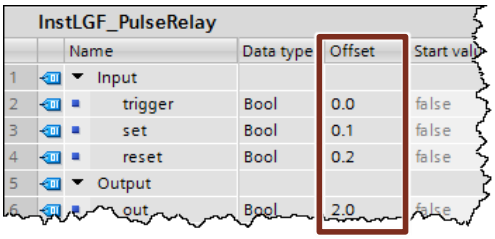


Tabelle 2-5: Unterschied: Optimierter und nicht optimierter Datenbaustein

Optimierter Datenbaustein	Nicht optimierter Datenbaustein
Optimierte Datenbausteine werden symbolisch adressiert. Deshalb wird **kein** "Offset" angezeigt.	Bei nicht optimierten Bausteinen wird der "Offset" angezeigt und kann zur Adressierung genutzt werden.
Im optimierten Baustein können Sie **jede** Variable einzeln mit "Remanenz" ("Retain") deklarieren.	Im nichtoptimierten Baustein können nur **alle oder keine** Variable mit "Remanenz" ("Retain") deklariert werden.

Die Remanenz von Variablen eines Global-DB definieren Sie direkt im Global-DB. Standardmäßig ist Nichtremanenz voreingestellt.

Die Remanenz von Variablen einer Instanz definieren Sie im Funktionsbaustein (nicht im Instanz-DB). Diese Einstellungen gelten damit für alle Instanzen dieses FBs.

Zugriffsarten bei optimierten und nicht optimierten Bausteinen

In folgender Tabelle sind alle Zugriffsarten auf Bausteine dargestellt.

Tabelle 2-6: Zugriffsarten

Zugriffsart	Optimierter Baustein	Nicht optimierter Baustein
Symbolisch	ja	ja
Indizierte (Felder)	ja	ja
Slice-Zugriffe	ja	ja
AT-Anweisung	nein (Alternative: Slice-Zugriff)	ja
Direkt Absolut	nein (Alternative: ARRAY mit Index)	ja
Indirekt Absolut (Zeiger)	nein (Alternative: VARIANT / ARRAY mit Index)	ja
Laden ohne Reinitialisierung	ja	nein

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Welche Unterschiede müssen Sie zwischen der optimierten Datenablage und dem Standard-Bausteinzugriff in STEP 7 (TIA Portal) beachten?
https://support.industry.siemens.com/cs/ww/de/view/67655611

Welche Eigenschaften müssen Sie in STEP 7 (TIA Portal) bei den Anweisungen "READ_DBL" und "WRIT_DBL" beachten, wenn Sie DBs mit optimiertem Zugriff verwenden?
https://support.industry.siemens.com/cs/ww/de/view/51434747

2.6.4 Konvertierung zwischen optimierten und nicht optimierten Variablen

Grundsätzlich wird empfohlen mit optimierten Variablen zu arbeiten. Falls man aber in Einzelfällen seine bisherige Programmierung beibehalten möchte, ergibt sich eine Mischung von optimierter und nicht optimierter Datenablage im Programm.

Das System kennt die interne Ablage jeder Variablen, egal ob strukturiert (von einem selbst definierten Datentyp abgeleitet) oder elementar (INT, LREAL, …).

Bei typgleichen Zuweisungen zwischen zwei Variablen mit unterschiedlicher Speicherablage konvertiert das System automatisch. Diese Konvertierung benötigt bei strukturierten Variablen Performance und sollte deshalb möglichst vermieden werden.

2.6.5 Parameterübergabe zwischen Bausteinen mit optimiertem und nicht optimierten Zugriff

Wenn bei einem Bausteinaufruf Strukturen als Durchgangsparameter (InOut) an den aufgerufenen Baustein übergeben werden, werden diese standardmäßig als Referenz übergeben (siehe Kapitel 3.3.2 Übergabe per Referenz (Call-by-reference)).

Dies gilt jedoch nicht, wenn einer der Bausteine die Eigenschaft "Optimierter Zugriff" und der andere Baustein die Eigenschaft "Standardzugriff" hat. Dann werden grundsätzlich alle Parameter als Kopie übergeben (siehe Kapitel 3.3.1 Übergabe per Wert (Call-by-value)).

Der aufgerufene Baustein arbeitet in diesem Fall immer mit den kopierten Werten. Während der Bausteinbearbeitung werden diese Werte möglicherweise verändert und nach Abarbeitung des Bausteinaufrufs wieder auf den ursprünglichen Operanden zurückkopiert.

Dies kann zu Problemen führen, wenn die ursprünglichen Operanden durch asynchrone Prozesse verändert werden, z.B. durch HMI-Zugriffe oder durch Alarm-OBs. Wenn nach der Bausteinbearbeitung die Kopien wieder auf die ursprünglichen Operanden zurückkopiert werden, werden dabei die asynchron durchgeführten Änderungen an den ursprünglichen Operanden überschrieben.

Hinweis

Weitere Informationen finden Sie in folgenden Beitrag:

Warum kann es zum Überschreiben von Daten des HMI-Systems oder des Webservers in der S7-1500 kommen?
https://support.industry.siemens.com/cs/ww/de/view/109478253

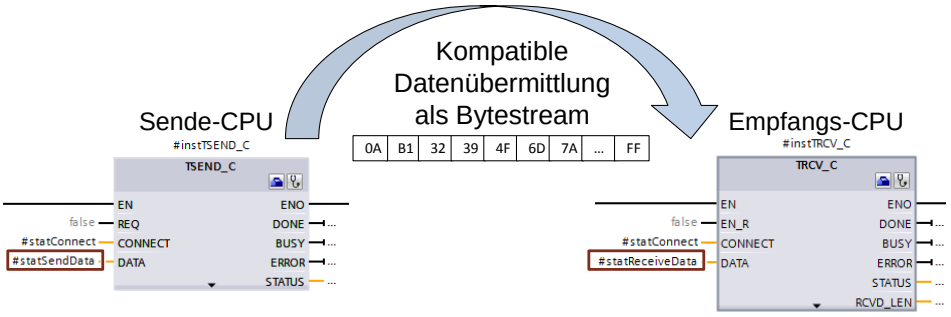
Empfehlung

- Stellen Sie immer in beiden Bausteinen, die miteinander kommunizieren dieselbe Zugriffsart ein.

2.6.6 Kommunikation mit optimierten Daten

Die Schnittstelle (CPU, CM) überträgt die Daten so, wie sie angeordnet sind (unabhängig ob optimiert oder nicht optimiert).

Abbildung 2-11: CPU-CPU Kommunikation



Sendedaten können sein:

- optimiert
- nicht optimiert
- Variable beliebigen Typs
- Buffer (Bytearray)

Empfangsdaten können sein:

- optimiert
- nicht optimiert
- Variable beliebigen Typs
- Buffer (Bytearray)

Beispiel

- Eine Variable mit PLC Datentyp (Datensatz) soll an eine CPU weitergegeben werden.
- In der Sende CPU wird die Variable als Aktualparameter mit dem Kommunikationsbaustein (TSEND_C) verschaltet.
- In der Empfangs CPU werden die Empfangsdaten einer Variablen gleichen Typs zugewiesen.
- In diesem Fall kann direkt mit den erhaltenen Daten symbolisch weitergearbeitet werden.

Hinweis Als Datensätze können jegliche Variablen oder Datenbausteine (abgeleitet von PLC Datentypen) verwendet werden.

Hinweis Es ist auch möglich, die Sende- und Empfangsdaten unterschiedlich zu definieren:

Sendedaten		**Empfangsdaten**
optimiert	-->	nicht optimiert
nicht optimiert	-->	optimiert

Die Steuerung sorgt automatisch für die richtige Datenübertragung und –ablage.

2.7 Bausteineigenschaften

2.7.1 Bausteingrößen

Bei S7-1200/1500 Steuerungen wurde die maximale Größe von Bausteinen im Arbeitsspeicher deutlich vergrößert.

Tabelle 2-7: Bausteingrößen

Max. Größe und Anzahl (ohne Berücksichtigung der Arbeitsspeichergröße)		**S7-300/400**	**S7-1200**	**S7-1500**
DB	Max. Größe	64 kB	64 kB	64 kB 16 MB (optimiert CPU1518)
	Max. Nummer	16.000	65.535	65.535
FC / FB	Max. Größe	64 kB	64 kB	512 kB
	Max. Nummer	7.999	65.535	65.535
FC / FB / DB	Max. Anzahl	4.096 (CPU319) 6.000 (CPU412)	1.024	10.000 (CPU1518)

Empfehlung

- Nutzen Sie DBs bei S7-1500 Steuerungen als Datencontainer von sehr großen Datenmengen.
- Datenmengen von > 64 kB können Sie mit S7-1500 Steuerungen in einem optimierten DB (max. Größe 16 MB) ablegen.

2.7.2 Anzahl der Organisationsbausteine (OB)

Mit OBs kann eine hierarchische Struktur des Anwenderprogramms erstellt werden. Hierfür stehen unterschiedliche OBs zur Verfügung.

Tabelle 2-8: Anzahl der Organisationsbausteine

Organisationsbautein Typ	**S7-1200**	**S7-1500**	**Nutzen**
Zyklische und Anlauf OBs	100	100	Modularisierung des Anwenderprogramms
Prozessalarme	50	50	Separater OB je Ereignis möglich
Verzögerungsalarme	4 *	20	Modularisierung des Anwenderprogramms
Weckalarme		20	Modularisierung des Anwenderprogramms
Uhrzeitalarme	nein	20	Modularisierung des Anwenderprogramms

* Ab Firmware V4 sind jeweils 4 Verzögerungsalarme und 4 Weckalarme möglich.

Empfehlung

- Nutzen Sie OBs, um das Anwenderprogramm hierarchisch zu strukturieren.
- Weitere Empfehlungen zur Verwendung von OBs finden Sie im Kapitel 3.2.1 Organisationsbausteine (OB).

2.7.3 Bausteinschnittstelle – Bausteinparameter ausblenden (ab V14)

Bausteinparameter können beim Aufruf des Bausteins definiert aus- oder eingeblendet werden. Hier haben Sie drei Möglichkeiten, die für jeden Formalparameter einzeln parametriert werden können.

- "Anzeigen" ("Show")
- "Verstecken" ("Hide")
- "Verstecken, wenn Parameter nicht zugewiesen" ("Hide if no parameter is assigned")

Vorteile

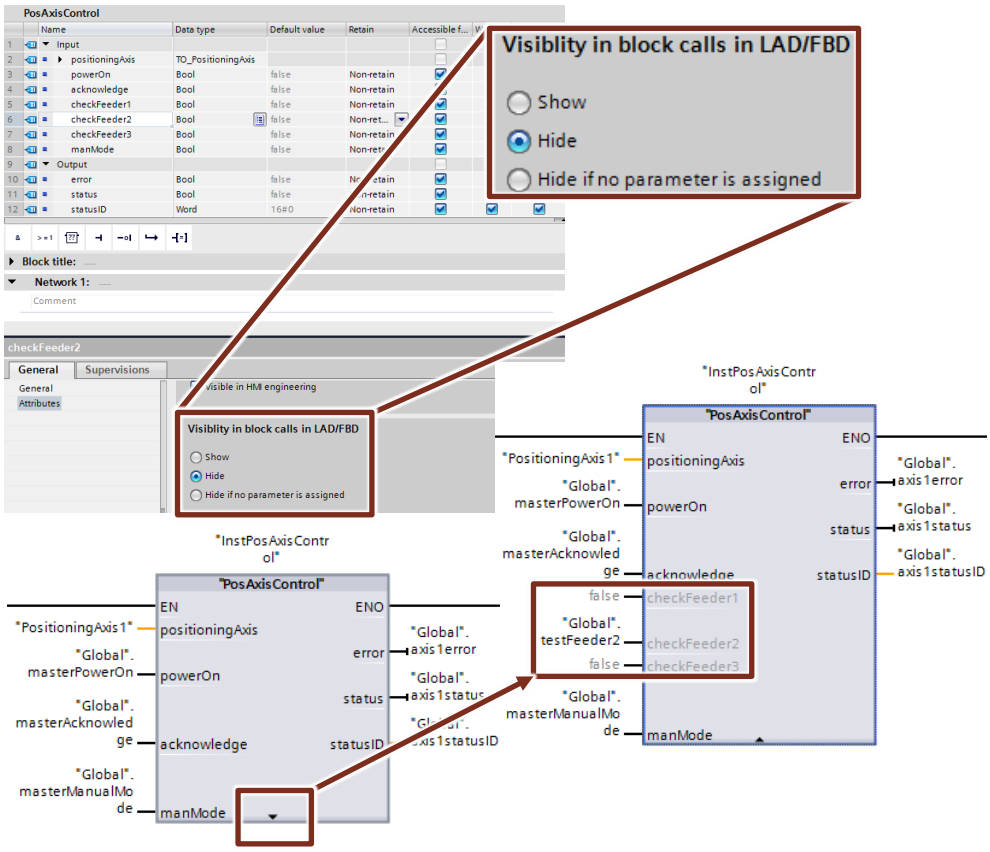
- Bessere Übersichtlichkeit bei Bausteinen mit vielen optionalen Parametern

Eigenschaften

- Nutzbar für:
  - FCs, FBs
  - In, Out, InOut

Beispiel

Abbildung 2-12: Bausteinparameter ausblenden



2.8 Neue Datentypen bei S7-1200/1500

S7-1200/1500 Steuerungen unterstützen neue Datentypen, um die Programmierung komfortabler zu ermöglichen. Mit den neuen 64 Bit Datentypen können wesentlich größere und genauere Werte genutzt werden.

Hinweis Weitere Informationen finden Sie in folgendem Beitrag:

Wie erfolgt im TIA Portal die Umwandlung von Datentypen für die S7-1200/1500?
https://support.industry.siemens.com/cs/ww/de/view/48711306

2.8.1 Elementare Datentypen

Tabelle 2-9: Ganzzahlige Datentypen

Typ	Größe	Wertebereich
USint	8 Bit	0 .. 255
SInt	8 Bit	-128 .. 127
UInt	16 Bit	0 .. 65535
UDInt	32 Bit	0 .. 4,3 Mio
ULInt*	64 Bit	0 .. 18,4 Trio (10^{18})
LInt*	64 Bit	-9,2 Trio .. 9,2 Trio
LWord	64 Bit	16#0000 0000 0000 0000 bis 16# FFFF FFFF FFFF FFFF

* nur bei S7-1500

Tabelle 2-10: Gleitkommadatentypen

Typ	Größe	Wertebereich
Real	32 bit (1 Bit Vorzeichen, 8 Bit Exponent, 23 Bit Mantisse), Genauigkeit 7 Stellen nach dem Komma	-3.40e+38 .. 3.40e+38
LReal	64 bit (1 Bit Vorzeichen, 11 Bit Exponent, 52 Bit Mantisse), Genauigkeit 15 Stellen nach dem Komma	-1.79e+308 .. 1.79e+308

Hinweis Weitere Informationen finden Sie in folgenden Beitrag:

Warum wird in STEP 7 (TIA Portal) das Ergebnis der DInt-Addition in SCL nicht richtig angezeigt?
https://support.industry.siemens.com/cs/ww/de/view/98278626

2.8.2 Datentyp Date_Time_Long

Tabelle 2-11: Aufbau von DTL (Date_Time_Long)

Year	Month	Day	Weekday	Hour	Minute	Second	Nanosecond

DTL liest immer die aktuelle Systemzeit. Der Zugriff auf die einzelnen Werte erfolgt durch die symbolischen Namen (z.B. `My_Timestamp.Hour`)

Vorteile

- Alle Teilbereiche (z.B. Year, Month, …) können symbolisch adressiert werden.

Empfehlung

Nutzen Sie den neuen Datentyp DTL statt LDT und adressieren Sie symbolisch (z.B. `My_Timestamp.Hour`).

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Wie können Sie in STEP 7 (TIA Portal) das Datum und die Uhrzeit für die CPU-Baugruppen der S7-300/S7-400/S7-1200/S7-1500 eingeben, lesen und weiter verarbeiten?
https://support.industry.siemens.com/cs/ww/de/view/43566349

Welche Funktionen stehen jeweils in STEP 7 V5.5 und im TIA Portal für die Bearbeitung der Datentypen DT und DTL zur Verfügung?
https://support.industry.siemens.com/cs/ww/de/view/63900229

2.8.3 Weitere Zeitdatentypen

Tabelle 2-12: Zeitdatentypen (nur S7-1500)

Typ	Größe	Wertebereich
LTime	64 Bit	LT#-106751d23h47m16s854ms775us808ns bis LT#+106751d23h47m16s854ms775us807ns
LTIME_OF_DAY	64 Bit	LTOD#00:00:00.000000000 bis LTOD#23:59:59.999999999

2.8.4 Unicode-Datentypen

Mit Hilfe der Datentypen WCHAR und WSTRING können Unicode-Zeichen verarbeitet werden.

Tabelle 2-13: Zeitdatentypen (nur S7-1500)

Typ	Größe	Wertebereich
WCHAR	2 Byte	-
WSTRING	(4 + 2*n) Byte	Voreingestellter Wert: 0 ..254 Zeichen Max. Wert: 0 ..16382

n = Länge der Zeichenkette

Eigenschaften

- Verarbeitung von Zeichen in z.B. lateinischen, chinesischen oder anderen Sprachen.
- Zeilenumbrüche, Seitenvorschub, Tabulator, Leerzeichen
- Sonderzeichen: Dollarzeichen, Anführungszeichen

Beispiel

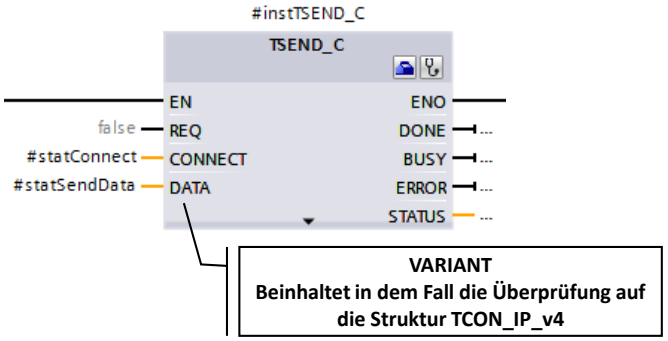
- `WCHAR#'a'`
- `WSTRING#'Hello World!'`

2.8.5 Datentyp VARIANT (S7-1500 und S7-1200 ab FW4.1)

Ein Parameter vom Typ VARIANT ist ein Zeiger, der auf Variablen verschiedener Datentypen zeigen kann. Im Gegensatz zum ANY-Pointer ist VARIANT ein Zeiger mit Typprüfung. D.h. die Zielstruktur und Quellstruktur werden zur Laufzeit geprüft und müssen identisch sein.

VARIANT wird z.B. bei Kommunikationsbausteinen (TSEND_C) als Eingang verwendet.

Abbildung 2-13: Datentyp VARIANT als Eingangsparameter bei Anweisung TSEND_C



Vorteile

- Integrierte Typprüfung verhindert fehlerhafte Zugriffe.
- Durch symbolische Adressierung der Variant-Variablen ist der Code leichter lesbar.
- Code ist effizienterer und kürzer programmierbar.
- Variant-Pointer sind deutlich intuitiver als ANY-Pointer.
- Variant-Variablen können direkt mit Hilfe von Systemfunktionen typrichtig genutzt werden.
- Flexible und performante Übergabe unterschiedlicher strukturierter Variablen ist möglich.

Eigenschaften

In einem Vergleich zwischen ANY und Variant sind die Eigenschaften erkennbar.

Tabelle 2-14: Vergleich ANY und Variant

ANY	Variant
Benötigt 10 Byte Speicher mit definierter Struktur	Benötigt für den Anwender keinen Arbeitsspeicher
Initialisierung entweder über Zuweisung des Datenbereichs oder mittels Füllen der ANY-Struktur	Initialisierung mittels Zuweisung des Datenbereichs oder Systemanweisung
Untypisiert – Typ einer verschalteten Struktur nicht erkennbar	Typisiert – Verschalteter Typ und bei Arrays auch die Länge kann ermittelt werden
Teilweise Typisiert – Bei Arrays kann auch die Länge ermittelt werden	VARIANT kann über Systemanweisungen ausgewertet und auch erzeugt werden

Empfehlung

- Prüfen Sie, wofür Sie bislang den ANY-Pointer verwendet haben. In vielen Fällen ist kein Pointer mehr notwendig (siehe folgende Tabelle).
- Verwenden Sie den Datentyp VARIANT nur bei der indirekten Adressierung, wenn die Datentypen erst zur Programmlaufzeit bestimmt werden.
  - Verwenden Sie den Datentyp VARIANT als InOut-Formalparameter zur Erstellung von generischen Bausteinen, die unabhängig vom Datentyp der Aktualparameter sind (siehe Beispiel in diesem Kapitel).
  - Nutzen Sie den Datentyp VARIANT anstatt des ANY-Zeigers. Durch die integrierte Typprüfung werden Fehler frühzeitig erkannt. Durch die symbolische Adressierung kann der Programmcode einfach interpretiert werden.
  - Verwenden Sie die Variant Anweisung z.B. zur Typerkennung (siehe folgendes Beispiel und Kapitel 2.9.2 VARIANT Anweisungen)
- Nutzen Sie den Index bei Arrays statt die Array Elemente mittels ANY zu adressieren (siehe Kapitel 3.6.2 Datentyp ARRAY und indirekte Feldzugriffe).

Tabelle 2-15: Vergleich ANY-Pointer und Vereinfachungen

Wofür werden ANY-Pointer genutzt?		Vereinfachung mit S7-1200/1500
Funktionen programmieren, die verschiedene Datentypen verarbeiten können	→	Funktionen mit Variant Pointer als InOut-Parameter bei Bausteinen (siehe folgendes Beispiel)
Verarbeitung von Arrays • z.B. Lesen, initialisieren, kopieren von Elementen gleichen Typs	→	Standard Array-Funktionen • Lesen und schreiben mit #myArray[#index] (siehe Kapitel 3.6.2 Datentyp ARRAY und indirekte Feldzugriffe) • Kopieren mit MOVE_BLK (siehe Kapitel 2.9.1 MOVE Anweisungen)
• Strukturen übergeben und performant per Absolutadressierung verarbeiten z.B. anwenderdefinierte Strukturen mittels ANY-Pointer an Funktionen übergeben	→	Strukturen als InOut-Parameter übergeben • siehe Kapitel 3.3.2 Übergabe per Referenz (Call-by-reference)

Hinweis Wenn Werte von nichtstruktureierten VARIANT Variablen kopiert werden sollen, können Sie auch VariantGet statt MOVE_BLK_VARIANT (siehe Kapitel 2.9.2 VARIANT Anweisungen) verwenden.

Beispiel

Mit dem Datentyp VARIANT ist es möglich im Anwenderprogramm Datentypen zu erkennen und entsprechend darauf zu reagieren. Der folgende Code des FCs "MoveVariant" zeigt eine mögliche Programmierung.

- Der InOut-Formalparameter "InVar" (Datentyp VARIANT) wird genutzt, um auf eine Variable unabhängig vom Datentyp zu zeigen.
- Der Datentyp des Aktualparameters wird mit der Anweisung "Type_Of" erkannt.
- Der Variablenwert wird mit der Anweisung "MOVE_BLK_VARIANT" auf die unterschiedlichen Ausgangs-Formalparameter in Abhängigkeit vom Datentyp kopiert.
- Falls der Datentyp des Aktualparameters nicht erkannt wird, gibt der Baustein einen Fehlercode zurück.

Abbildung 2-14: Formalparameter des FCs "MoveVariant"

MoveVariant

	Name	Data type	Default value
1	Input		
2	Output		
3	outInteger	Int	
4	outReal	Real	
5	outTypeCustom	"typeCustom"	
6	InOut		
7	inOutVariant	Variant	
8	Temp		
9	Constant		
10	Return		

```
CASE TypeOf(#inOutVariant) OF // Check datatypes
   Int:  // Move Integer
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outInteger);
   Real:  // Move Real
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outReal);
   typeCustom:  // Move outTypeCustom
      #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outTypeCustom);
   ELSE  // Error, no sufficient datatype
      #MoveVariant := WORD_TO_INT(#NO_CORRECT_DATA_TYPE);
      // 80B4: Error-Code of MOVE_BLK_VARIANT: Data types do
         not correspond
END_CASE;
```

2.9 Anweisungen

Das TIA Portal unterstützt den Programmierer mit vorgefertigten Anweisungen (Bitverknüpfungen, Zeiten, Zähler, Vergleicher…).

Hinweis

Weitere Funktion können Sie sich unter folgenden Beitrag runterladen:

Bibliothek mit generellen Funktionen (LGF) für STEP 7 (TIA Portal) und S7-1200 / S7-1500
https://support.industry.siemens.com/cs/ww/de/view/109479728

2.9.1 MOVE Anweisungen

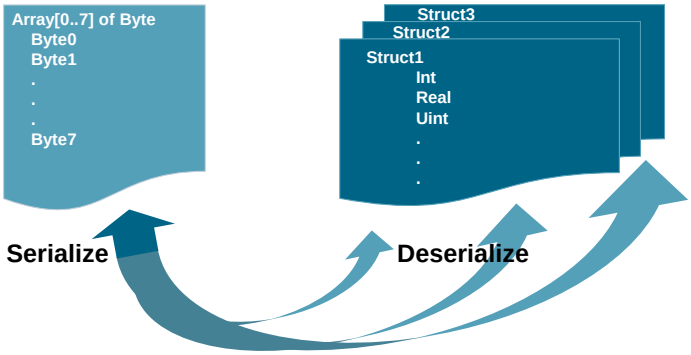
In STEP 7 (TIA Portal) stehen Ihnen die folgenden MOVE-Anweisungen zur Verfügung. Neu ist die Anweisung MOVE_BLK_VARIANT für S7-1200/1500.

Tabelle 2-16:Move-Anweisungen

Anweisung	Verwendung	Eigenschaften
MOVE	Wert kopieren	• Kopiert den Inhalt des Parameters am Eingang IN zum Parameter des Ausgangs OUT. • Parameter am Ein- und Ausgang müssen vom gleichen Datentyp sein. • Parameter können auch strukturierte Variablen (PLC-Datentypen) sein. • Kopiert komplette Arrays und Strukturen.
MOVE_BLK	Bereich kopieren	• Kopiert den Inhalt eines Array-Bereiches in einen anderen Array-Bereich. • Quell- und Zielbereich müssen vom selben Datentyp sein. • Kopiert komplette Arrays und Strukturen. • Kopiert mehrere Array-Elemente auch mit Strukturen. Zusätzlich kann Start und Anzahl der Elemente angegeben werden.
UMOVE_BLK	Bereich ununterbrech-bar kopieren	• Kopiert den Inhalt eines Array-Bereiches konsistent ohne Risiko, dass das Kopieren durch OBs unterbrochen werden. • Quell- und Zielbereich müssen vom selben Datentyp sein.
MOVE_BLK_VARIANT (S7-1500 und S7-1200 ab FW4.1)	Bereich kopieren	• Kopiert einen oder mehrere strukturierte Variablen (PLC-Datentypen) • Erkennt Datentypen zur Laufzeit • Liefert ausführliche Fehlerinformationen • Neben den elementaren und strukturierten Datentypen werden auch PLC-Datentypen, Arrays und Array-DBs unterstützt

Anweisung	Verwendung	Eigenschaften
Serialize (S7-1500 und S7-1200 ab FW4.1)	wandelt strukturierte Daten in ein Byte-Array	• Mehrere Datensätze können zu einem Byte-Array zusammengefasst werden und z.B. als Telegramm an andere Geräte verschickt werden. • Eingangs und Ausgangsparameter können als Datentyp Variant übergeben werden.
Deserialize (S7-1500 und S7-1200 ab FW4.1)	wandelt ein Byte-Array in eine oder mehrere Struktur/en	• Anwendungsfall I-Device: Das I-Device empfängt im Eingangsbereich mehrere Datensätze, die auf unterschiedliche Strukturen kopiert werden. • Mehrere Datensätze können zu einem Byte-Array zusammengefasst werden. Mit Deserialize können diese auf unterschiedliche Strukturen umkopiert werden.

Abbildung 2-15: Serialize und Deserialize (S7-1500 und S7-1200 ab FW4.1)



Eigenschaften

Anweisungen, wie "Serialize: Serialisieren", "Deserialize: Deserialisieren", "CMP" (Vergleicher) und "MOVE: Wert kopieren" können sehr große und komplex strukturierte Variablen verarbeiten. Hierbei analysiert die CPU zur Laufzeit den Aufbau der Variablenstruktur. Die Verarbeitungszeit hängt ab von folgenden Eigenschaften der zu verarbeitenden Variablenstruktur:

- Komplexität der Struktur
- Anzahl der Strukturen ohne Nutzung von PLC-Datentypen
- Array of Byte kann in optimierten Bausteinen gespeichert werden (ab V14).

Empfehlung

- Deklarieren Sie Strukturen mithilfe von PLC-Datentypen anstatt mit "STRUCT"
- Reduzieren Sie die Anzahl verwendeter Strukturen:
  - Vermeiden Sie z. B. die mehrfache Deklaration von sehr ähnlich aufgebauten Strukturen. Fassen Sie diese in einer einzigen Struktur zusammen.
  - Wenn viele Elemente der Struktur denselben Datentyp haben, dann verwenden Sie nach Möglichkeit den Datentyp ARRAY.

- Unterscheiden Sie generell zwischen MOVE, MOVE_BLK und MOVE_BLK_VARIANT
  - Um komplette Strukturen zu kopieren, verwenden Sie die Anweisung MOVE.
  - Um Teile von ARRAYs mit bekanntem Datentyp zu kopieren, verwenden Sie die Anweisung MOVE_BLK.
  - Nur wenn Sie Teile von ARRAYs mit erst zur Programmlaufzeit bekanntem Datentyp kopieren wollen, verwenden Sie die Anweisung MOVE_BLK_VARIANT.

Hinweis UMOVE_BLK: Der Kopiervorgang kann nicht durch andere Tätigkeiten des Betriebssystems unterbrochen werden. Aus diesem Grund können sich die Alarmreaktionszeiten der CPU während der Bearbeitung der Anweisung "Bereich ununterbrechbar kopieren" erhöhen.

Die komplette Beschreibung der MOVE-Anweisungen finden Sie in der Online Hilfe des TIA Portals.

Hinweis Weitere Informationen finden Sie in folgenden Beitrag:

Wie können Sie in STEP 7 (TIA Portal) Speicherbereiche und strukturierte Daten zwischen zwei Datenbausteinen kopieren?
https://support.industry.siemens.com/cs/ww/de/view/42603881

2.9.2 VARIANT Anweisungen (S7-1500 und S7-1200 ab FW4.1)

Tabelle 2-17:Move-Anweisungen

Anweisung	Verwendung	Eigenschaften
MOVE Anweisungen		
VariantGet	Wert lesen	Mit dieser Anweisung lesen Sie den Wert einer Variablen, auf die ein VARIANT zeigt.
VariantPut	Wert schreiben	Mit dieser Anweisung schreiben Sie den Wert einer Variablen, auf die ein VARIANT zeigt.
Aufzählung		
CountOfElements	Elemente zählen	Mit dieser Anweisung fragen Sie die Anzahl der ARRAY-Elemente einer Variablen auf die der VARIANT zeigt, ab.
Vergleicher Anweisungen		
TypeOf() (nur SCL)	Ermittlung des Datentyps	Mit dieser Anweisung fragen Sie den Datentyp einer Variablen, auf die ein VARIANT zeigt, ab
TypeOfElements() (nurSCL)	Ermittlung des Array-Datentyps	Mit dieser Anweisung fragen Sie den Datentyp der ARRAY-Elemente einer Variablen, auf die ein VARIANT zeigt, ab.

Anweisung	Verwendung	Eigenschaften
Umwandler Anweisungen		
VARIANT_TO_DB_ANY (nur SCL)	Ermittlung der Datenbausteinnummer	Mit dieser Anweisung fragen Sie die Datenbausteinnummer eines Instanz-Datenbaustein eines PLC-Datentyps, Systemdatentyps oder Array-DBs.
DB_ANY_TO_VARIANT (nurSCL)	Erzeugt aus einem Datenbaustein eine Variant-Variable.	Mit dieser Anweisung erzeugen Sie eine Variant-Variable aus einem Instanz-Datenbaustein eines PLC-Datentyps, Systemdatentyps oder Array-DBs.

Hinweis Weitere VARIANT-Anweisungen finden Sie in der Online Hilfe des TIA Portals.

Eigenschaften

Variant-Anweisungen brauchen auf Grund ihres komplexen Algorithmus eine längere Abarbeitungszeit als direkte Zuweisungen.

Empfehlung

- Verwenden Sie, wenn möglich keine Variant-Anweisungen in Schleifen (FOR, WHILE…), um die Zykluszeit nicht unnötig zu erhöhen.
- Verwenden Sie zum Kopieren eines Arrays keine Laufschleife über die Elemente, sondern die direkte Zuweisung des ganzen Arrays.

2.9.3 RUNTIME

Mit der Anweisung "RUNTIME" messen Sie die Laufzeit des gesamten Programms, einzelner Bausteine oder von Befehlssequenzen. Aufrufen können Sie diese Anweisung in KOP, FUP, SCL und in AWL (nur S7-1500).

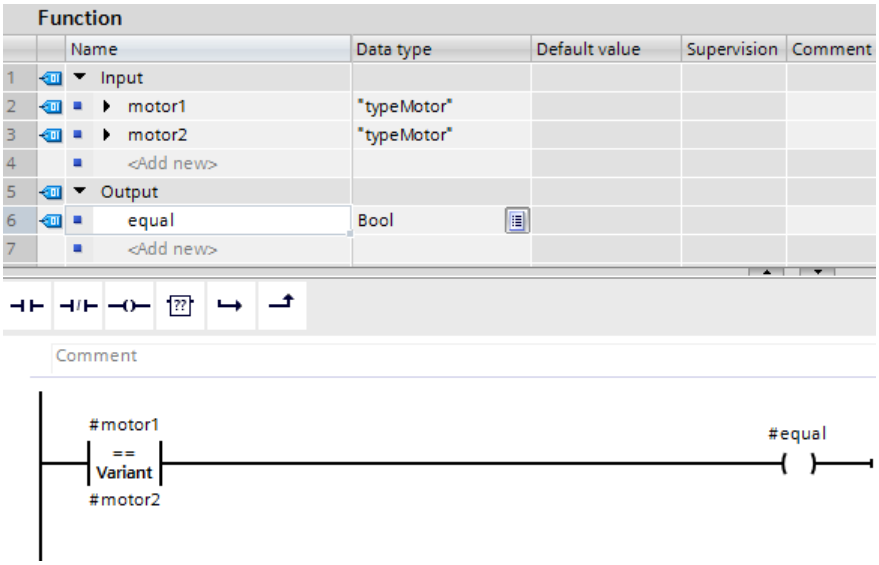
Hinweis Weitere Informationen finden Sie in folgendem Beitrag:

Wie können Sie bei der S7-1200/S7-1500 die Zeit eines Programmteils oder den gesamten Programmzyklus zur Laufzeit erfassen?
https://support.industry.siemens.com/cs/ww/de/view/87668055

2.9.4 Vergleich von Variablen aus PLC-Datentypen (ab V14)

Zwei Variablen vom gleichen PLC-Datentyp können miteinander auf Gleich- bzw. auf Ungleichheitheit überprüft werden.

Abbildung 2-16: Vergleich von Variablen aus PLC-Datentypen in KOP

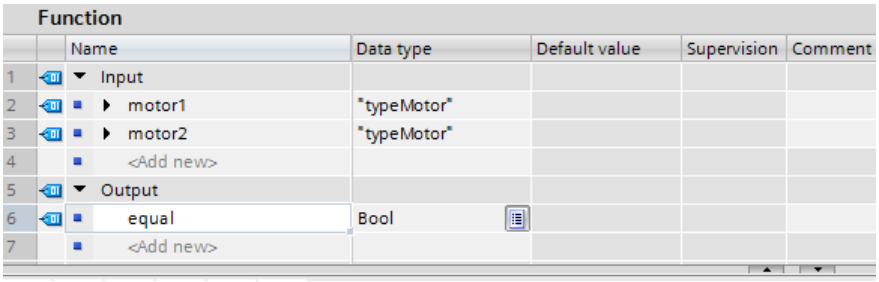


Vorteile

- Symbolisch Programmierung mit strukturierten Variablen
- Vergleich mit optimaler Performance
- Vergleich ist möglich KOP, FUP, SCL.
- Vergleich ist direkt in SCL-Anweisungen möglich.

Beispiel

Abbildung 2-17: Vergleich von Variablen aus PLC-Datentypen in SCL-Anweisungen




```
IF #motor1 = #motor2 THEN
   // Statement section IF
   ;
END_IF;
```

2.9.5 Mehrfachzuweisung (ab V14)

Vorteile

Mehrfachzuweisung ermöglichen eine optimale Programmierung für mehrere Variablen (z.B. bei Initialisierungen).

Beispiel

```
#statFillLevel := #statTemperature := #tempTemperature := 0.0;
```

2.10 Symbolik und Kommentare

2.10.1 Programmiereditor

Vorteile

Mit Verwendung von symbolischen Namen und Kommentaren in Ihrem Programm können Sie den Code für Kollegen verständlich und leicht lesbar machen.
Die komplette Symbolik wird beim Download auf der Steuerung zusammen mit dem Programmcode gespeichert und ermöglicht somit eine schnelle Wartung der Anlage selbst, wenn kein Offline-Projekt vorliegt.

Empfehlung

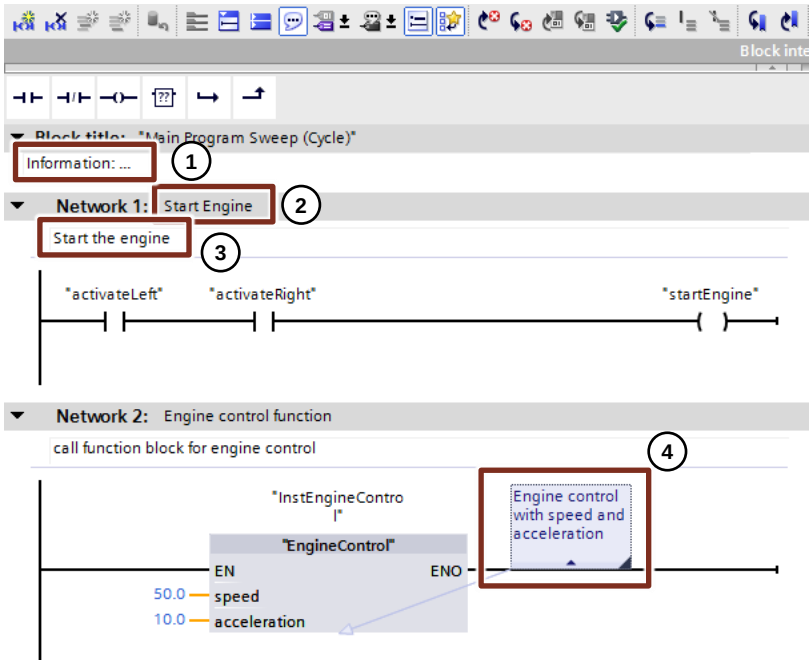
- Nutzen Sie Kommentare in den Programmen, um die Lesbarkeit zu verbessern. Netztitelkommentare sind auch bei eingeklappten Netzwerken sichtbar.
- Gestalten Sie den Programmcode so, dass auch Kollegen das Programm direkt verstehen können.

Im folgenden Beispiel sehen Sie die umfassenden Möglichkeiten zur Kommentierung des Programms in den Editoren.

Beispiel

In folgender Abbildung sehen Sie die Kommentierungsmöglichkeiten im KOP-Editor (gleiche Funktionalität in FUP).

Abbildung 2-18: Kommentieren im Anwenderprogramm (KOP)



Folgende Kommentare sind möglich:

1. Bausteinkommentar
2. Netzwerktitelkommentar
3. Netzwerkkommentar
4. Kommentar zu Anweisungen, Bausteinen und Funktionen (Öffner, Schließer, usw.)

In den Programmiersprachen SCL und AWL kann mit // in jeder Zeile kommentiert werden.

Beispiel

```
statFuellstand := statRadius * statRadius * PI * statHoehe;
// Berechnung des Füllstands für Medium-Tank
```

Hinweis

Weitere Informationen finden Sie unter folgenden Beitrag:

Warum werden in STEP 7 (TIA Portal) die Anzeigetexte, Titel und Kommentare nach dem Öffnen des Projekts im Bausteineditor nicht mehr angezeigt?
https://support.industry.siemens.com/cs/ww/de/view/41995518

2.10.2 Kommentarzeilen in Beobachtungstabellen

Vorteile

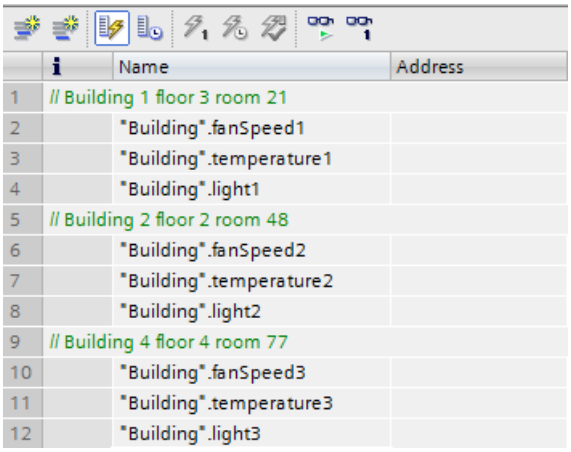
- Zur besseren Strukturierung ist es möglich Kommentarzeilen in Beobachtungstabellen zu erstellen.

Empfehlung

- Verwenden Sie immer Kommentarzeilen und untergliedern Sie damit ihre Variablentabelle.
- Kommentieren Sie auch die einzelnen Variablen.

Beispiel

Abbildung 2-19: Beobachtungstabelle mit Kommentarzeilen



	i	Name	Address
1	// Building 1 floor 3 room 21		
2		"Building".fanSpeed1	
3		"Building".temperature1	
4		"Building".light1	
5	// Building 2 floor 2 room 48		
6		"Building".fanSpeed2	
7		"Building".temperature2	
8		"Building".light2	
9	// Building 4 floor 4 room 77		
10		"Building".fanSpeed3	
11		"Building".temperature3	
12		"Building".light3	

2.11 Systemkonstanten

Bei S7-300/400 Steuerungen erfolgt die Identifikation der Hard- und Softwarekomponenten über logische Adressen bzw. Diagnoseadressen.

Bei S7-1200/1500 erfolgt die Identifikation über die Systemkonstanten. Alle Hard- und Softwarekomponenten (z.B. Schnittstellen, Baugruppen, OBs, ...) der S7-1200/1500 Steuerungen haben ihre eigenen Systemkonstanten. Erzeugt werden die Systemkonstanten automatisch während der Erstellung der Gerätekonfiguration für die zentrale und dezentrale Peripherie.

Vorteile

- Sie können über den Baugruppennamen statt über die Hardwarekennung adressieren.

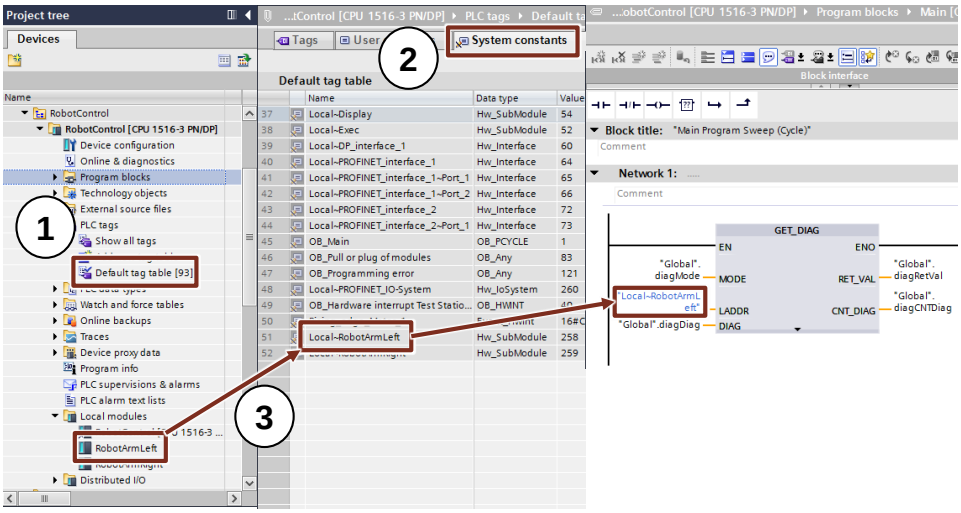
Empfehlung

- Vergeben Sie funktionsbezogene Baugruppennamen, um die Baugruppe bei der Programmierung einfach zu identifizieren.

Beispiel

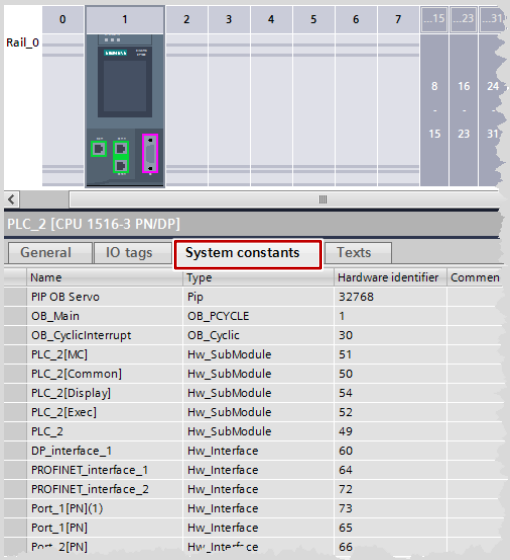
Im folgenden Beispiel sehen Sie, wie Systemkonstanten im Anwenderprogramm eingesetzt werden.

Abbildung 2-20: "Systemkonstanen" ("System constants") im Anwenderprogramm



1. Systemkonstanten einer Steuerung finden Sie im Ordner "PLC tags – Default tag table".
2. Die Systemkonstanten sind in einem eigenen Register in der "Default tag table".
3. In diesem Beispiel wurde der symbolische Name "RobotArmLeft" für eine DI-Baugruppe vergeben.
   Unter diesem Namen finden Sie die Baugruppe auch in der Tabelle der Systemkonstanten.
   Im Anwenderprogramm ist "RobotArmLeft" mit dem Diagnosebaustein "GET_DIAG" verschaltet.

Hinweis Öffnen Sie die "Gerätekonfiguration" ("Device configuration"), um schnell die Systemkonstanten für jedes Gerät zu finden.



Hinweis Weitere Informationen finden Sie in folgenden Beitrag:

Welche Bedeutung haben die Systemkonstanten in STEP 7 (TIA Portal) bei der S7-1200/1500?
https://support.industry.siemens.com/cs/ww/de/view/78782835

2.12 Anwenderkonstanten

Mit Hilfe von Anwenderkonstanten können konstante Werte gespeichert werden. Generell gibt es lokale Konstanten für OBs, FCs und FBs und globale Konstanten für das komplette Anwenderprogramm in einer Steuerung.

Vorteile

- Mit Anwenderkonstanten können konstante Werte global oder lokal für alle Verwendungsstellen geändert werden.
- Mit Anwenderkonstanten kann das Programm lesbarer gestaltet werden.

Eigenschaften

- Lokale Anwenderkonstanten werden in der Bausteinschnittstelle definiert.
- Globale Anwenderkonstanten werden unter "PLC-Variablen" definiert.
- Im Anwenderprogramm kann auf Anwenderkonstanten nur lesend zugegriffen werden.
- Bei Know-How geschützten Bausteinen sind die Anwenderkonstanten nicht sichtbar.

Empfehlung

- Verwenden Sie Anwenderkonstanten zur besseren Lesbarkeit des Programms und zentralen Änderbarkeit von …
  - Fehlercodes,
  - CASE-Anweisungen,
  - Umrechnungsfaktoren,
  - natürlichen Konstanten …

Beispiel

Abbildung 2-21: Lokale Anwenderkonstante eines Bausteins bei CASE-Anweisungen

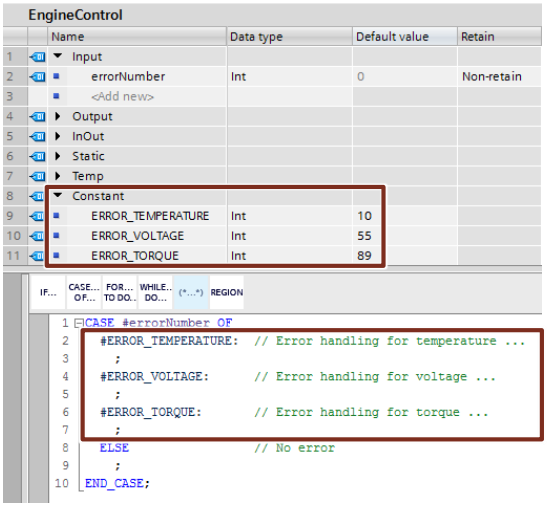
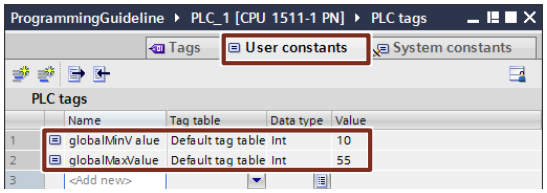


Abbildung 2-22: Globale Anwenderkonstante einer Steuerung



Hinweis

Ein weiteren Anwendungsfall von Konstanten finden Sie unter folgenden FAQ:

Wie können Sie die Einheit einer Variablen in STEP 7 (TIA Portal) umrechnen?
https://support.industry.siemens.com/cs/ww/de/view/61928891

2.13 Interne Referenz ID für Variablen von Steuerung und HMI

STEP 7, WinCC, Startdrive, Safety u.a. integrieren sich in die gemeinsame Datenbasis des Engineering-Frameworks TIA Portal. Änderungen von Daten werden automatisch an allen Stellen des Anwenderprogramms übernommen, unabhängig davon, ob Sie in einer Steuerung, einem Panel oder einem Antrieb erfolgen. So können keine Inkonsistenzen der Daten entstehen.

2.13 Interne Referenz ID für Variablen von Steuerung und HMI

Wenn Sie Variablen erstellen, wird automatisch vom TIA Portal eine eindeutige Referenz-ID angelegt. Die Referenz-ID ist für Sie nicht sicht- oder programmierbar. Es handelt sich um eine interne Referenzierung. Bei Änderung von Variablen (Adresse) bleibt die Referenz-ID unverändert.

In folgender Abbildung ist interne Referenz auf Daten schematisch dargestellt.

Abbildung 2-23: interne Referenz-ID bei PLC und HMI

PLC1 **HMI1**

PLC Symbol Name	Absolute Adresse	Interne PLC Referenz ID	Interne HMI Referenz ID	HMI Symbol Name	Access mode	Verbindung mit PLC
motor1	I0.0	000123	009876	motor1	<symbolic access>	PLC1_HMI1
valve2	Q0.3	000138	000578	valve2	<symbolic access>	PLC1_HMI1

Hinweis

Die ID wird verändert durch …

- Variable umbenennen.
- Typ ändern.
- Löschen der Variable.

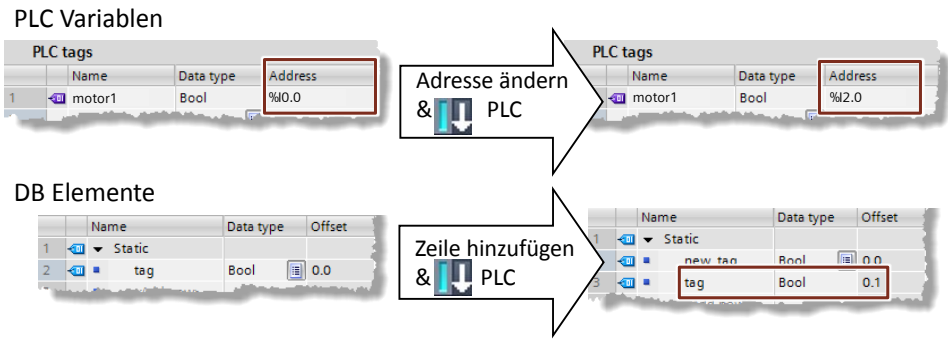
Vorteile

- Sie können Variablen umverdrahten, ohne damit die internen Bezüge zu verändern. Auch die Kommunikation zwischen Steuerung, HMI und Antrieb bleibt unverändert.
- Die Länge des symbolischen Namens beeinflusst nicht die Kommunikationslast zwischen Steuerung und HMI.

Eigenschaften

Ändern Sie die Adressen von PLC-Variablen, müssen Sie nur die Steuerung neu laden, da das System intern mit den Referenz-IDs adressiert. Es ist somit nicht notwendig, die HMI Geräte erneut zu laden (siehe Abbildung 2-24: Adresse ändern oder Zeile hinzufügen).

Abbildung 2-24: Adresse ändern oder Zeile hinzufügen



2.14 Betriebszustand STOP bei Fehler

Im Vergleich zu S7-300/400 gibt es bei S7-1200/1500 weniger Kriterien, die zum Betriebszustand "STOP" führen.

Durch die geänderte Konsistenzprüfung im TIA Portal kann der Betriebszustand "STOP" bei den Steuerungen S7-1200/1500 in den meisten Fällen schon im Vorfeld ausgeschlossen werden. Die Konsistenz von Programmbausteinen wird schon beim Kompilieren im TIA Portal geprüft. Diese Vorgehensweise macht die S7-1200/1500 Steuerungen "fehlertoleranter" als ihre Vorgänger.

Vorteile

Es gibt nur drei Fehlersituationen, die die S7-1200/1500 Steuerungen in den Betriebszustand STOP versetzen. Dadurch ist das Fehlermanagement übersichtlicher und einfacher zu programmieren.

Eigenschaften

Tabelle 2-18: Reaktion auf Fehler von S7-1200/1500

	Fehler	S7-1200	S7-1500
1.	Zyklusüberwachungszeit 1 mal überschritten	RUN	**STOP** (wenn OB80 nicht projektiert ist)
2.	Zyklusüberwachungszeit 2 mal überschritten	**STOP**	**STOP**
3.	Programmierfehler	RUN	**STOP** (wenn OB121 nicht projektiert ist)

Fehler OBs:

- OB80 "Zeitfehler-OB" ("Time error interrupt") wird vom Betriebssystem aufgerufen, wenn die maximale Zykluszeit der Steuerung überschritten wird.
- OB121 "Programmierfehler-OB" ("Programming error") wird vom Betriebssystem aufgerufen, wenn ein Fehler bei der Programmabarbeitung auftritt.

Zusätzlich wird bei jedem Fehler automatisch ein Eintrag im Diagnosepuffer erstellt.

Hinweis Bei S7-1200/1500 Steuerungen gibt es noch weitere programmierbare Fehler-OBs (Diagnosefehler, Baugruppenträgerausfall, usw.).

Weitere Information über die Fehlerreaktionen von S7-1200/1500 finden Sie in der Online Hilfe des TIA Portals unter "Ereignisse und OBs" ("Events and OBs").

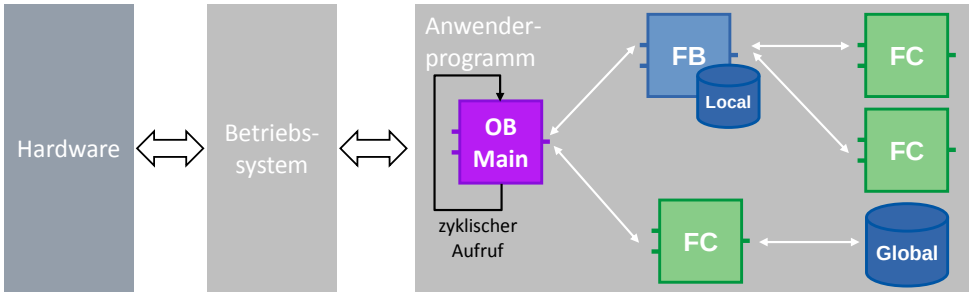
3 Allgemeine Programmierung

3.1 Betriebssystem und Anwenderprogramm

SIMATIC Steuerungen bestehen aus Betriebssystem und Anwenderprogramm.

- Das Betriebssystem organisiert alle Funktionen und Abläufe der Steuerung, die nicht mit einer spezifischen Steuerungsaufgabe (z.B. Abwickeln von Neustart, Aktualisieren des Prozessabbilds, Aufrufen des Anwenderprogramms, Umgang mit Fehlern, Speicherverwaltung, usw.) verbunden sind. Das Betriebssystem ist fester Bestandteil der Steuerung.
- Das Anwenderprogramm enthält alle Bausteine, die zur Bearbeitung Ihrer spezifischen Automatisierungsaufgabe erforderlich sind. Das Anwenderprogramm wird mit Programmbausteinen programmiert und auf die Steuerung geladen.

Abbildung 3-1: Betriebssystem und Anwenderprogramm



Bei SIMATIC Steuerungen wird das Anwenderprogramm immer zyklisch ausgeführt. Der Zyklus-OB "Main" ist bereits im Ordner "Programmbausteine" ("Programm blocks") vorhanden, nachdem eine Steuerung in STEP 7 angelegt wurde. Der Baustein wird von der Steuerung abgearbeitet und in einer Endlosschleife wieder aufgerufen.

3.2 Programmbausteine

Auch in STEP 7 (TIA Portal) gibt es alle bekannten Bausteintypen aus vorherigen STEP 7 Versionen:

- Organisationsbausteine
- Funktionsbausteine
- Funktionen
- Datenbausteine

Erfahrene Anwender von STEP 7 finden sich sofort zurecht und neue Anwender können sich leicht in die Programmierung einarbeiten.

Vorteile

- Mit den verschiedenen Bausteinarten können Sie ihr Programm übersichtlich und strukturiert gliedern.
- Durch ein gutes und strukturiertes Programm erhalten Sie viele Funktionseinheiten, die Sie innerhalb eines Projektes und auch in anderen Projekten mehrfach wiederverwenden können. Diese Funktionseinheiten unterscheiden sich dann in aller Regel nur in einer unterschiedlichen Parametrierung (siehe Kapitel 3.2.9 Wiederverwendbarkeit von Bausteinen).

- Ihr Projekt bzw. Ihre Anlage wird transparent. D.h. Störungszustände einer Anlage können leichter erkannt, analysiert und behoben werden. D.h. Die Wartbarkeit Ihrer Anlage wird einfacher. Dies gilt auch für Fehler in der Programmierung.

Empfehlung

- Strukturieren Sie Ihre Automatisierungsaufgabe.
- Zerlegen Sie die Gesamtfunktion der Anlage in Einzelbereiche und bilden Sie Unterfunktionseinheiten. Gliedern Sie auch diese Funktionseinheiten wieder in kleinere Einheiten und Funktionen. Untergliedern Sie solange, bis Sie Funktionen erhalten, die Sie mit unterschiedlichen Parametern mehrfach verwenden können.
- Legen Sie Schnittstellen zwischen den Funktionseinheiten fest. Definieren Sie eindeutige Schnittstellen zu Funktionalitäten, die von "Fremdfirmen" zuzuliefern sind.

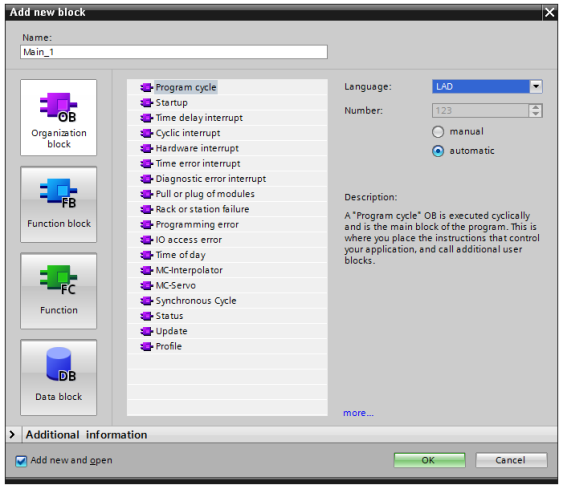
Alle Organisationsbausteine, Funktionsbausteine, und Funktionen können mit folgenden Sprachen programmiert werden:

Table 3-1: Programmiersprachen

Programmiersprache	S7-1200	S7-1500
Kontaktplan (KOP oder LAD)	ja	ja
Funktionsplan (FUP oder FBD)	ja	ja
Structured Control Language (SCL)	ja	ja
Graph	nein	ja
Anweisungsliste (AWL oder STL)	nein	ja

3.2.1 Organisationsbausteine (OB)

Abbildung 3-2: "Neuen Baustein einfügen"-Dialog (OB)



OBs bilden die Schnittstelle zwischen dem Betriebssystem und dem Anwenderprogramm. Sie werden vom Betriebssystem aufgerufen und steuern z. B. folgende Vorgänge:

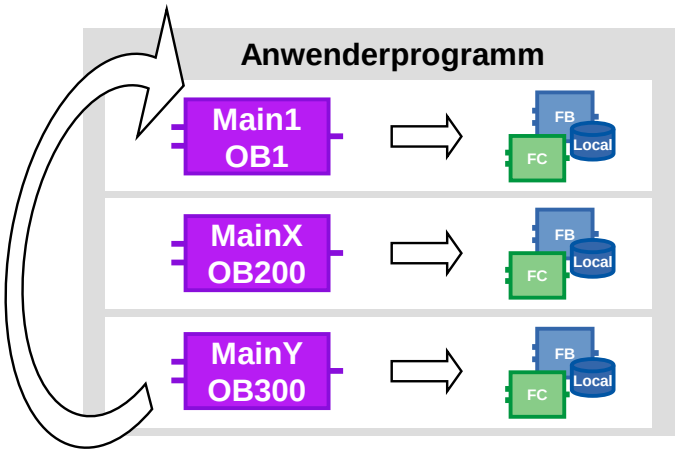
- Anlaufverhalten der Steuerung
- Zyklische Programmbearbeitung
- Alarmgesteuerte Programmbearbeitung
- Behandlung von Fehlern

Je nach Steuerung steht Ihnen eine Vielzahl von verschiedenen OB-Typen zu Verfügung.

Eigenschaften

- OBs werden vom Betriebssystem der Steuerung aufgerufen.
- Mehrere Main-OBs können in einem Programm erstellt werden. Die OBs werden sequentiell nach der OB Nummer abgearbeitet.

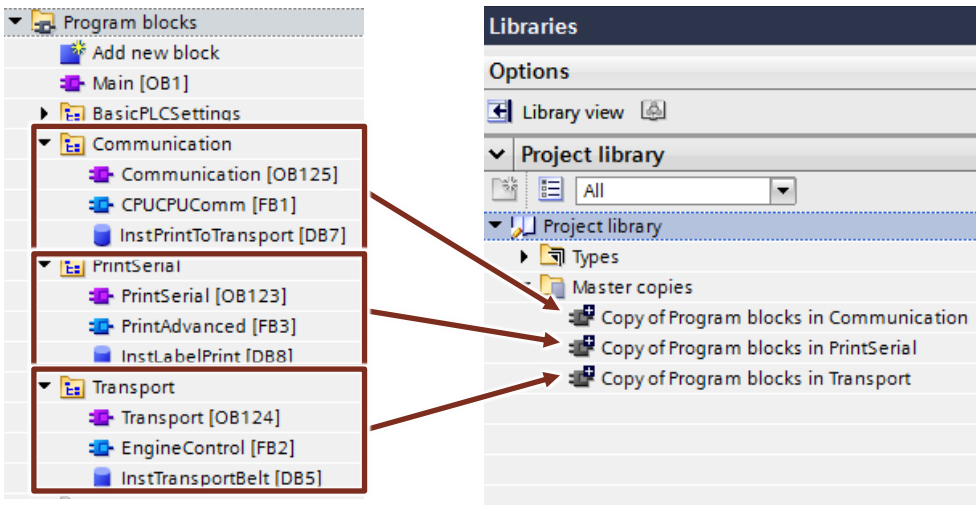
Abbildung 3-3: Verwendung von mehreren Main-OBs



Empfehlung

- Kapseln Sie unterschiedliche Programmteile, die evtl. austauschbar von Steuerung zu Steuerung sein sollen, in mehreren Main-OBs.
- Vermeiden Sie die Kommunikation zwischen den unterschiedlichen Main-OBs. Dann sind sie unabhängig voneinander einsetzbar. Falls Sie dennoch Daten zwischen den einzelnen Main-OBs austauschen, nutzen Sie Global-DBs (siehe Kapitel 4.2 Keine Merker, sondern globale Datenbausteine).
- Gliedern Sie alle Programmanteile, die zusammengehören in Ordner und legen Sie diese zur Wiederverwendung in der Projekt- oder globalen Bibliothek ab.

Abbildung 3-4: Programmteile geordnet in Projektbibliothek ablegen



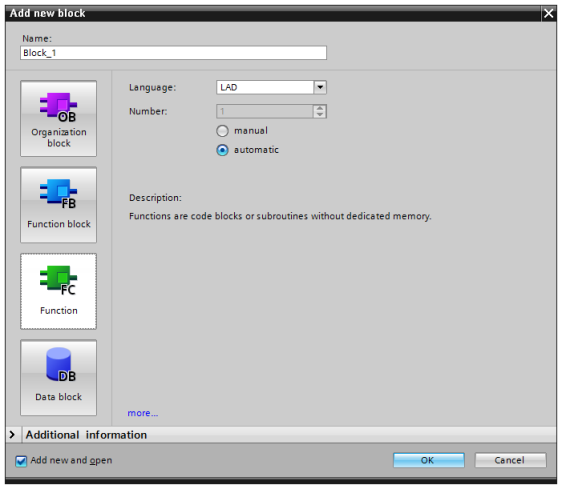
Weitere Informationen finden Sie im Kapitel 3.7 Bibliotheken.

Hinweis Weitere Informationen finden Sie in folgendem Beitrag:

Welche Organisationsbausteine können Sie in STEP 7 (TIA Portal) verwenden?
https://support.industry.siemens.com/cs/ww/de/view/40654862

3.2.2 Funktionen (FC)

Abbildung 3-5: "Neuen Baustein einfügen"-Dialog (FC)



FCs sind Bausteine ohne zyklischen Datenspeicher. Deshalb können Werte von Bausteinparametern nicht bis zum nächsten Aufruf gespeichert werden und müssen beim Aufruf mit Aktualparameter versorgt werden.

Eigenschaften

- FCs sind Bausteine ohne zyklischen Datenspeicher.
- Temporäre Variablen sind beim Aufruf in nicht optimierten Bausteinen undefiniert. Bei optimierten Bausteinen sind die Werte immer mit dem "Defaultwert" vorbelegt (S7-1500 und S7-1200 Firmware ab V4.0). Dadurch entsteht kein zufälliges, sondern ein reproduzierbares Verhalten.
- Um Daten eines FCs dauerhaft zu speichern, stehen den Funktionen globale Datenbausteine zur Verfügung.
- FCs können mehrere Ausgänge haben.
- Der Funktionswert kann in SCL in einer Formel direkt weiterverwendet werden.

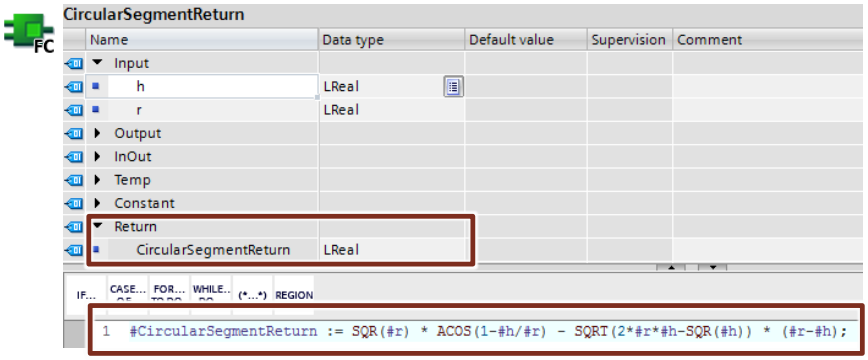
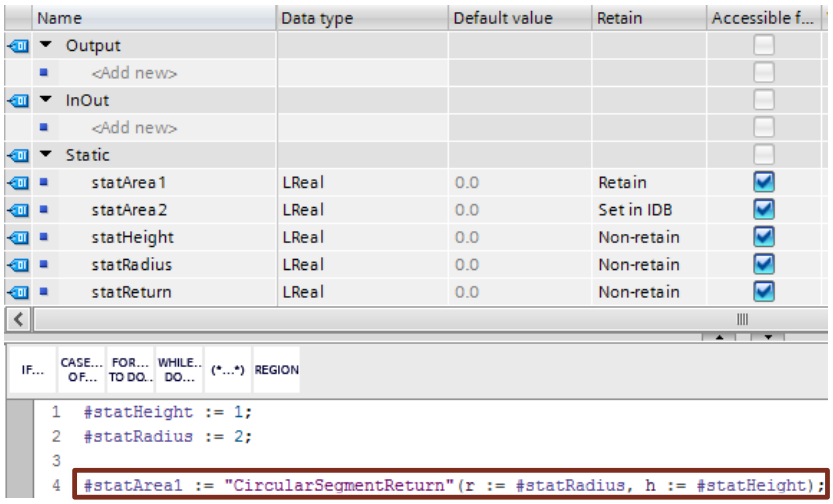
Empfehlung

- Nutzen Sie Funktionen für häufig wiederkehrende Anwendungen, die an verschiedenen Stellen des Anwenderprogramms mehrfach aufgerufen werden.
- Nutzen Sie die Möglichkeit in SCL, den Funktionswert direkt weiter zu verwenden.
  ```
  <Operand> := <FC-Name> (Parameterliste);
  ```

Beispiel

Im folgenden Beispiel wird eine mathematische Formel in einem FC programmiert. Das Ergebnis der Berechnung wird direkt als Rückgabewert deklariert und der Funktionswert direkt weiterverwendet.

Tabelle 3-2: Funktionswert weiter verwenden

Schritt	Anweisung
1.	Erstellen Sie einen FC mit der mathematischen Formel (Kreissegment) und definieren Sie den "Return"-Wert als Ergebnis für die Formel. 
2.	Rufen Sie den FC mit der Kreissegmentberechnung in einem beliebigen Baustein (SCL) auf. `<Operand> := <FC-Name> (Parameterliste);` 

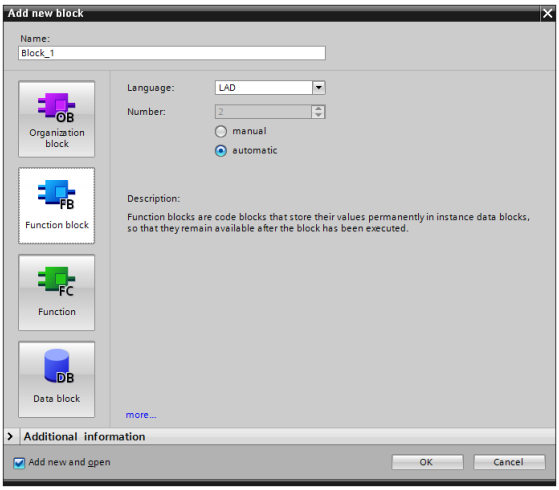
Hinweis

Weitere Informationen finden Sie in folgenden Beitrag:

Wie viele Parameter dürfen Sie in STEP 7 (TIA Portal) für eine Funktion in der S7-1200/S7-1500 CPU maximal definieren?
https://support.industry.siemens.com/cs/ww/de/view/99412890

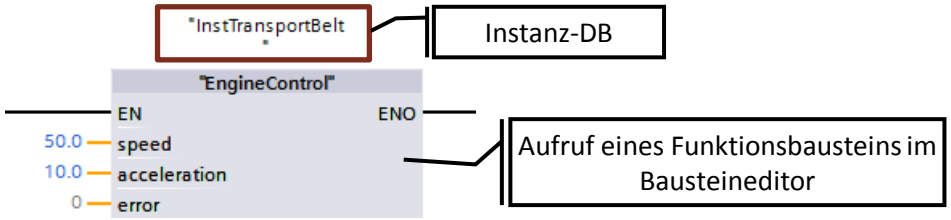
3.2.3 Funktionsbausteine (FB)

Abbildung 3-6: "Neuen Baustein einfügen"-Dialog (FB)



FBs sind Bausteine mit zyklischem Datenspeicher, in dem Werte dauerhaft gespeichert werden. Der zyklische Datenspeicher ist in einem Instanz-DB realisiert.

Abbildung 3-7: Aufruf eines Funktionsbausteins



Eigenschaften

- FBs sind Bausteine mit zyklischem Datenspeicher.
- Temporäre Variablen sind beim Aufruf in nicht optimierten Bausteinen undefiniert. Bei optimierten Bausteinen sind die Werte immer mit dem "Defaultwert" vorbelegt (S7-1500 und S7-1200 Firmware V4). Dadurch entsteht kein zufälliges Verhalten, sondern ein reproduzierbares Verhalten.
- Statische Variablen behalten den Wert von Zyklus zu Zyklus.

Empfehlung

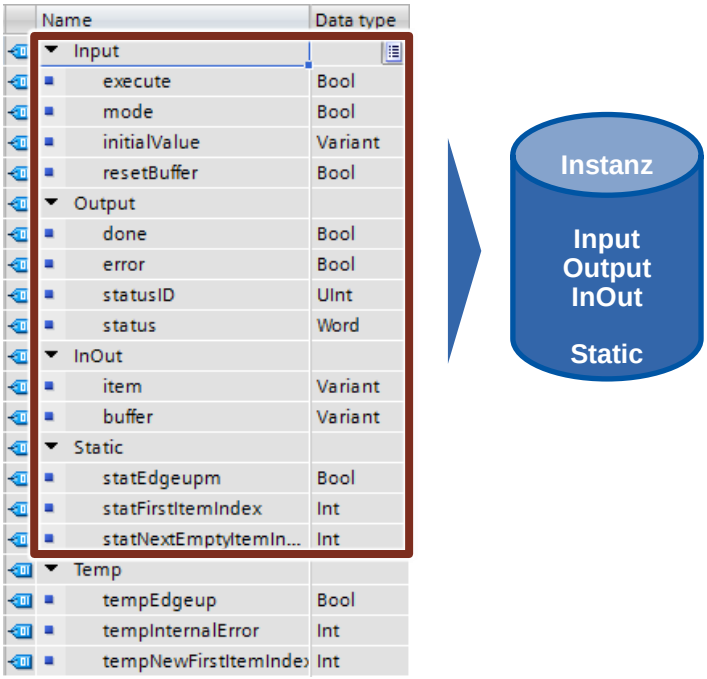
- Nutzen Sie Funktionsbausteine, um Unterprogramme für verschiedene Anwendungen zu erstellen und strukturieren Sie das Anwenderprogramm. Ein Funktionsbaustein kann auch mehrmals an verschiedenen Stellen des Anwenderprogramms aufgerufen werden. Sie erleichtern sich so die Programmierung häufig wiederkehrender Programmteile.
- Nutzen Sie bei mehrfacher Verwendung von Funktionsbausteinen im Anwenderprogramm eigene Instanzen, wenn möglich Multiinstanzen.

3.2.4 Instanzen

Der Aufruf eines Funktionsbausteins wird als Instanz bezeichnet. Die Daten, mit denen die Instanz arbeitet, werden in einem Instanz-DB gespeichert.

Instanz-DBs werden immer nach den Vorgaben in der FB-Schnittstelle erzeugt und können somit nicht im Instanz-DB verändert werden.

Abbildung 3-8: Aufbau der Schnittstellen eines FBs



Der Instanz-DB besteht aus einem dauerhaften Speicher mit den Schnittstellen Input, Output, InOut und Static. In einem flüchtigen Speicher (L-Stack) werden temporäre Variablen gespeichert. Der L-Stack ist immer nur für die aktuelle Bearbeitung gültig. D.h. temporäre Variablen müssen in jedem Zyklus initialisiert werden.

Eigenschaften

- Instanz-DBs sind immer einem FB zugeordnet.
- Instanz-DBs müssen im TIA Portal nicht manuell erstellt werden und werden automatisch beim Aufruf eines FBs angelegt.
- Die Struktur des Instanz-DBs wird im zugehörigen FB festgelegt und kann nur dort geändert werden.

Empfehlung

- Programmieren Sie so, dass die Daten des Instanz-DBs nur vom zugehörigen FB geändert werden. So können Sie gewährleisten, dass der Baustein universell in allen möglichen Projekten einsetzbar ist.

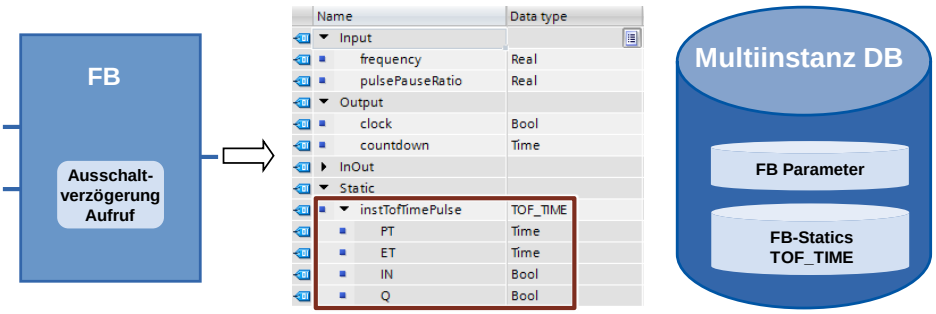
Weitere Informationen finden Sie im Kapitel 3.4.1 Bausteinschnittstellen als Datenaustausch.

3.2.5 Multiinstanzen

Mit Multiinstanzen können aufgerufene Funktionsbausteine, ihre Daten in den Instanz-Datenbaustein des aufrufenden Funktionsbausteins ablegen. D.h. wenn in einem Funktionsbaustein ein weiterer Funktionsbaustein aufgerufen wird, speichert dieser seine Daten im Instanz-DB des übergeordneten FBs. Die Funktionalität des aufgerufenen Bausteins bleibt dadurch auch bei Weitergabe des aufrufenden Bausteins erhalten.

Folgende Abbildung zeigt einen FB, der einen weiteren FB ("IEC-Timer") nutzt. Alle Daten werden in einem Multiinstanz-DB gespeichert. Hierdurch kann ein Baustein mit eigenem Zeitverhalten z.B. ein Taktgeber erstellt werden.

Abbildung 3-9: Multiinstanzen



Vorteile

- Wiederverwendbarkeit
- Mehrfachaufrufe sind möglich
- Übersichtlicheres Programm mit weniger Instanz-DBs
- Einfacheres Kopieren von Programmen
- Gute Strukturierungsmöglichkeiten bei der Programmierung

Eigenschaften

- Multiinstanzen sind Speicherbereiche innerhalb von Instanz-DBs.

Empfehlung

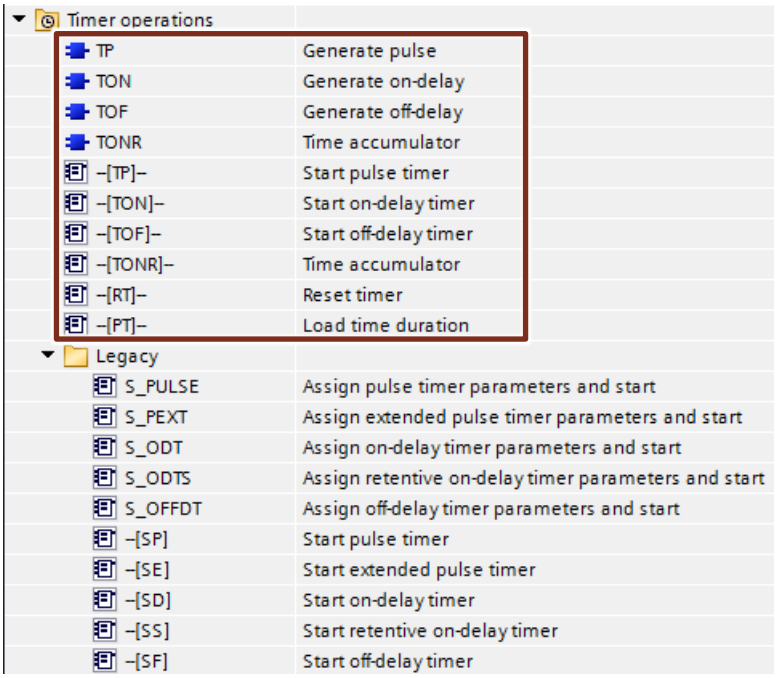
Setzen Sie Multiinstanzen ein, um …

- die Anzahl der Instanz-DBs zu reduzieren.
- wiederverwendbare und übersichtliche Anwenderprogramme zu erstellen.
- lokale Funktionen z.B. Timer, Zähler, Flankenauswertung zu programmieren.

Beispiel

Wenn Sie Zeit- und Zählerfunktionen benötigen, nutzen Sie die "IEC Timer"-Bausteine und "IEC Counter"-Bausteine, anstatt der absolut adressierten SIMATIC Timer. Nutzen Sie auch hier, wenn möglich immer Multiinstanzen. Somit wird die Anzahl an Bausteinen im Anwenderprogramm klein gehalten.

Abbildung 3-10: Bibliothek der IEC Timer



Hinweis

Weitere Informationen finden Sie in folgenden Beitrag:

Wie sollte in STEP 7 (TIA Portal) die Deklaration der Timer und Zähler für die S7-1500 erfolgen?
https://support.industry.siemens.com/cs/ww/de/view/67585220

3.2.6 Instanz als Parameter übergeben (V14)

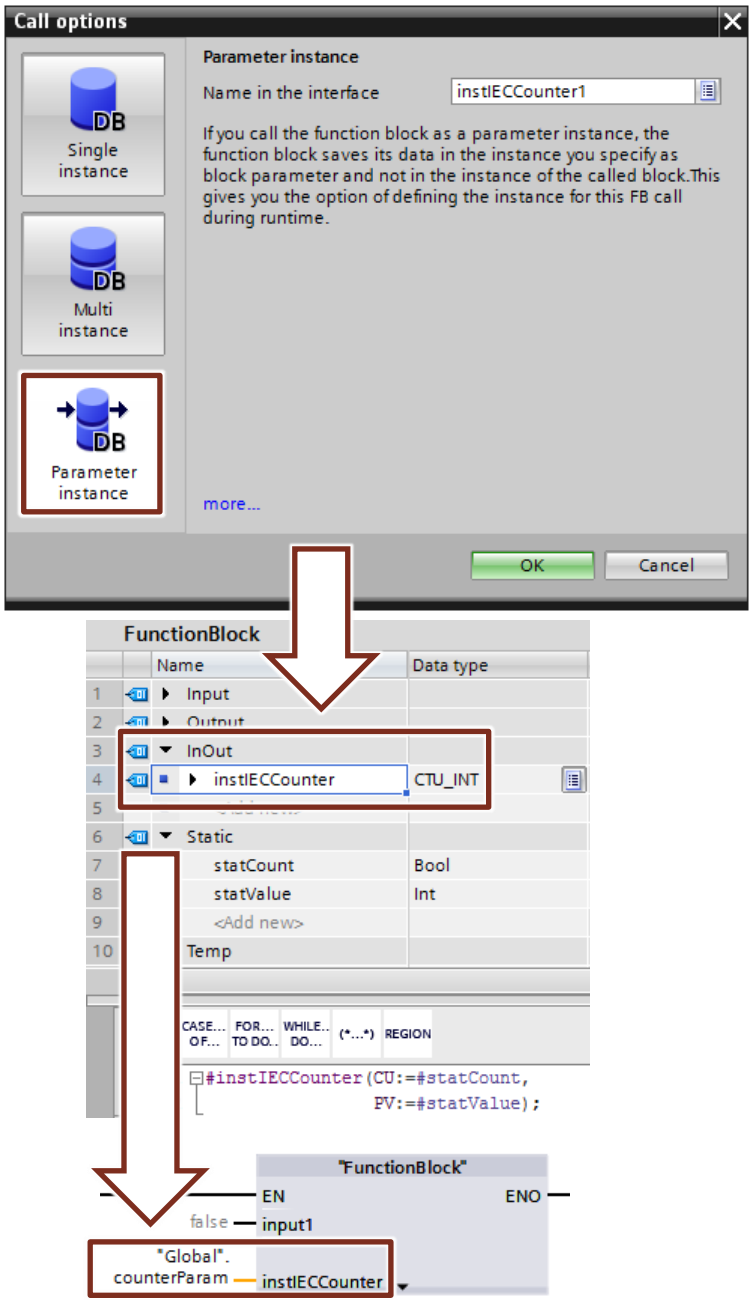
Instanzen von aufgerufenen Bausteinen können als InOut-Parameter definiert werden.

Vorteile

- Erstellung von standardisierten Funktionen ist möglich, denen dynamisch Instanzen übergeben werden.
- Erst beim Aufruf des Bausteins wird festgelegt, welche Instanz verwendet wird.

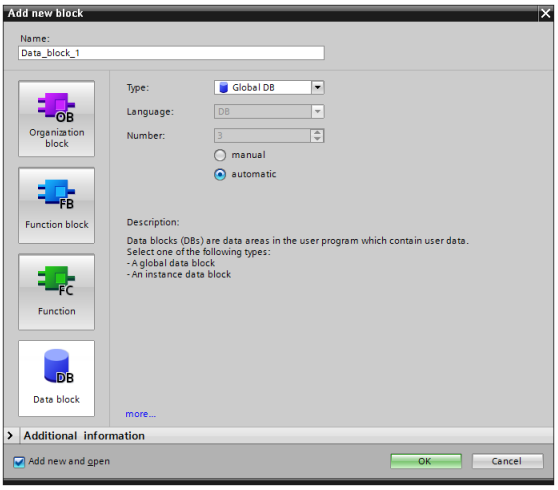
Beispiel

Abbildung 3-11 :Instanz als Parameter übergeben



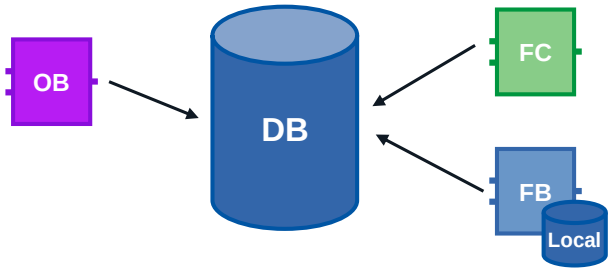
3.2.7 Globale Datenbausteine (DB)

Abbildung 3-12: "Neuen Baustein einfügen"-Dialog (DB)



In Datenbausteinen befinden sich variable Daten, die dem kompletten Anwenderprogramm zu Verfügung stehen.

Abbildung 3-13: Global-DB als zentraler Datenspeicher



Vorteile

- Gut strukturierbarer Speicherbereich
- Hohe Zugriffsgeschwindigkeit

Eigenschaften

- Alle Bausteine im Anwenderprogramm können auf Global-DBs zugreifen.
- Die Struktur des Global-DBs kann beliebig aus allen Datentypen zusammengesetzt werden.
- Global-DBs werden entweder über den Programmeditor oder gemäß eines vorher angelegten "Anwenderdefinierten PLC-Datentyp" erstellt (siehe Kapitel 3.6.4 Datentyp STRUCT und PLC-Datentypen).
- Pro Datenbaustein können maximal 256 strukturierte Variablen (ARRAY, STRUCT) definiert werden. Das betrifft keine Variablen, die von einem PLC-Datentyp abgeleitet sind.

Empfehlung

- Verwenden Sie Global-DBs, wenn Daten in verschiedenen Programmteilen bzw. Bausteinen verwendet werden.

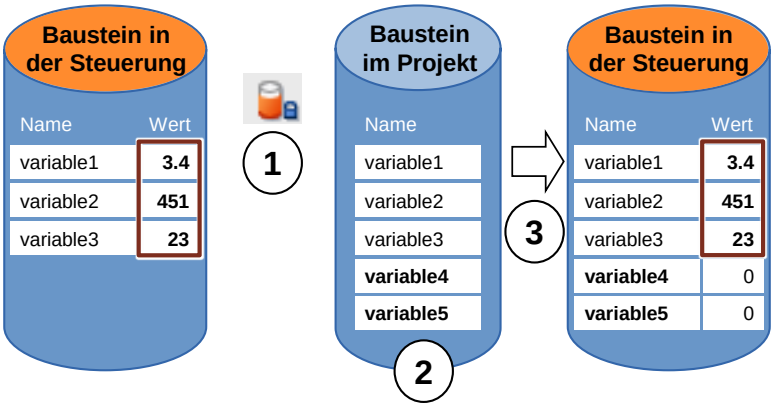
Hinweis Weitere Informationen finden Sie in folgendem Beitrag:

Welche Zugriffsarten, Wertspalten und Bedienmöglichkeiten gibt es für die globalen Datenbausteine in STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/de/view/68015630

3.2.8 Laden ohne Reinitialisierung

Um Anwenderprogramme, die bereits in einer Steuerung laufen, nachträglich zu ändern, bieten S7-1200 (Firmware V4.0) und S7-1500 Steuerungen die Möglichkeit, die Schnittstellen von optimierten Funktions- oder Datenbausteinen im laufenden Betrieb zu erweitern. Die geänderten Bausteine können Sie laden, ohne die Steuerung in STOP zu setzen und ohne die Aktualwerte von bereits geladenen Variablen zu beeinflussen.

Abbildung 3-14: Laden ohne Reinitialisierung



Führen Sie folgende Schritte durch, während die Steuerung im Betriebszustand RUN ist.

1. Aktivieren "Laden ohne Reinitialisierung"
2. Neu definierte Variablen in bestehenden Baustein einfügen
3. Baustein in Steuerung laden

Vorteile

- Nachladen von neu definierten Variablen ohne den laufenden Prozess zu unterbrechen. Die Steuerung bleibt dabei im Betriebszustand "RUN".

Eigenschaften

- Laden ohne Reinitialisierung ist nur bei optimierten Bausteinen möglich.
- Die neu definierten Variablen werden initialisiert. Die bestehenden Variablen behalten ihren aktuellen Wert.
- Ein Baustein mit Reserve benötigt mehr Speicherplatz in der Steuerung.
- Die Speicherreserve ist abhängig vom Arbeitsspeicher der Steuerung, beträgt aber maximal 2 MB.
- Es wird vorausgesetzt, dass eine Speicherreserve für den Baustein definiert ist.
- Die Speicherreserve ist standardmäßig auf 100 Byte eingestellt.

- Die Speicherreserve wird für jeden Baustein einzeln definiert.
- Die Bausteine können variabel erweitert werden.

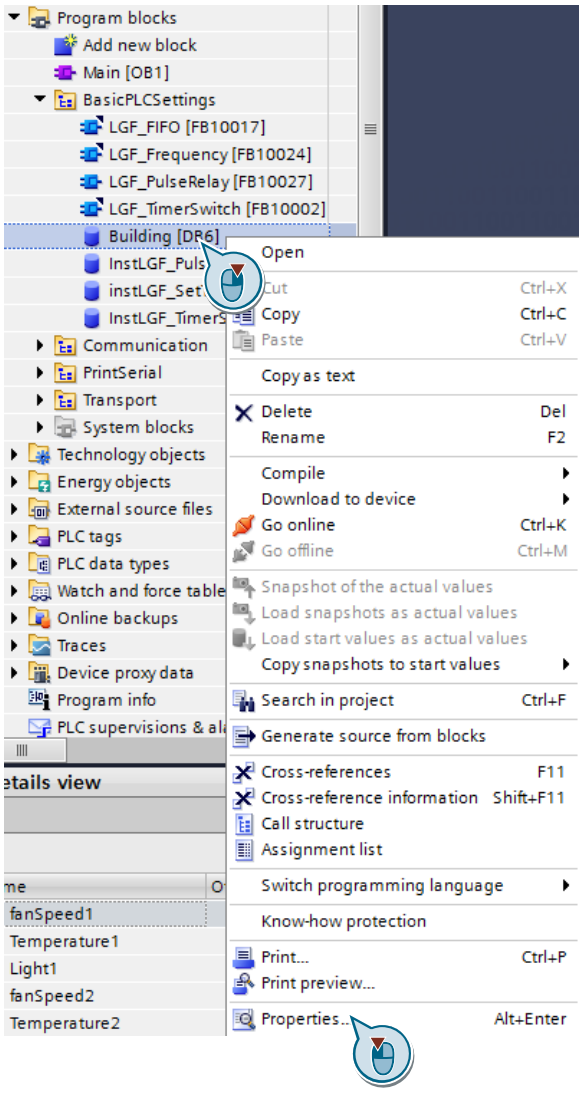
Empfehlung

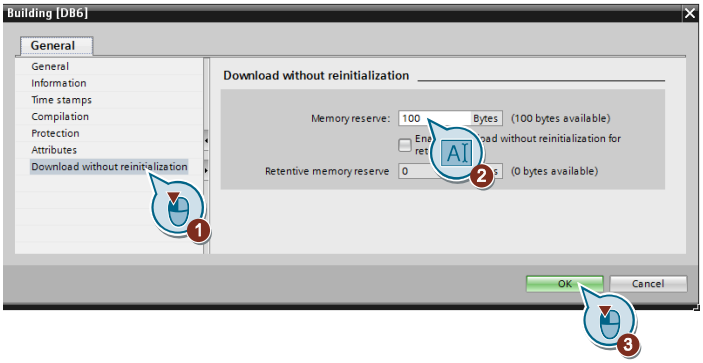
- Definieren Sie eine Speicherreserve für Bausteine, die während der Inbetriebnahme erweitert werden sollen (z.B. Testbausteine). Der Inbetriebnahmeprozess wird nicht durch einen Download von neu definierten Variablen gestört, da die Aktualwerte der bestehenden Variablen erhalten bleiben.

Beispiel: Speicherreserve am Baustein einstellen

In folgender Tabelle ist beschrieben, wie Sie die Speicherreserve für das Laden ohne Reinitialisierung einstellen können.

Tabelle 3-3: Speicherreserve einstellen

Schritt	Anweisung
1.	Klicken Sie mit der rechten Maustaste auf einen beliebigen optimierten Baustein im Projektnavigator und wählen "Eigenschaften" ("Properties"). 

Schritt	Anweisung
2.	 Klicken Sie auf "Laden ohne Reinitialisierung" ("Download without reinitialization"). 3. Geben Sie die gewünschte Speichergröße für die "Speicherreserve"´("Memory reserve") ein. 4. Bestätigen Sie mit "OK"

Hinweis

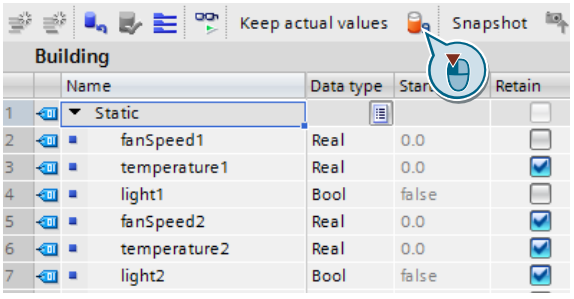
Sie können im TIA Portal auch einen Standardwert für die Größe der Speicherreserve für neue Bausteine einstellen.

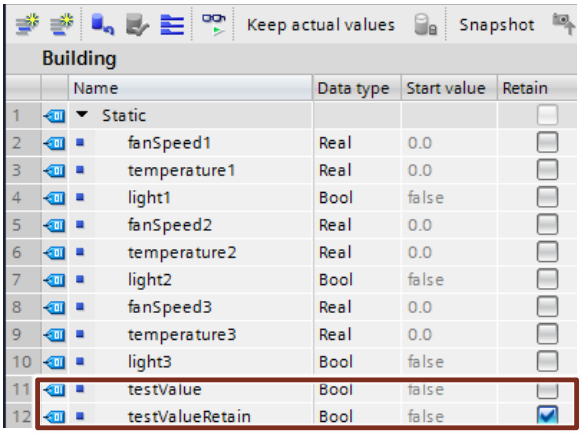
Navigieren Sie in die Menüleiste zu "Extras - Einstellungen" ("Options - Settings") und anschließend zu "PLC-Programmierung – Allgemein – Laden ohne Reinitialisierung" ("PLC programming – General – Download without reinitialization").

Beispiel: Laden ohne Reinitialisierung

Im folgenden Beispiel wird dargestellt, wie ohne Reinitialisierung geladen wird.

Tabelle 3-4 Laden ohne Reinitialisierung

Schritt	Anweisung
1.	Voraussetzung: es muss eine Speicherreserve eingestellt sein (siehe oben).
2.	Öffnen Sie z.B. einen optimierten Global-DB.
3.	Klicken Sie auf die Schaltfläche "Speicherreserve aktivieren" ("Activate memory reserve") und bestätigen Sie den Dialog mit "OK". 

Schritt	Anweisung
4.	Fügen Sie neue Variablen (remanente Variablen sind auch möglich) ein. 
5.	Laden Sie den Baustein auf die Steuerung.
6.	Ergebnis: • Aktualwerte des Bausteins bleiben erhalten

Hinweis

Weitere Information finden Sie in der Online Hilfe des TIA Portals unter "Bausteinerweiterungen ohne Reinitialisierung laden" ("Loading block extensions without reinitialization").

Weitere Informationen finden Sie unter folgendem Beitrag:

Welche Möglichkeiten bietet die S7-1500 beim Download von Daten im RUN?
https://support.industry.siemens.com/cs/ww/de/view/68015630

3.2.9 Wiederverwendbarkeit von Bausteinen

Das Bausteinkonzept bietet Ihnen eine Vielzahl an Möglichkeiten strukturiert und effektiv zu programmieren.

Vorteile

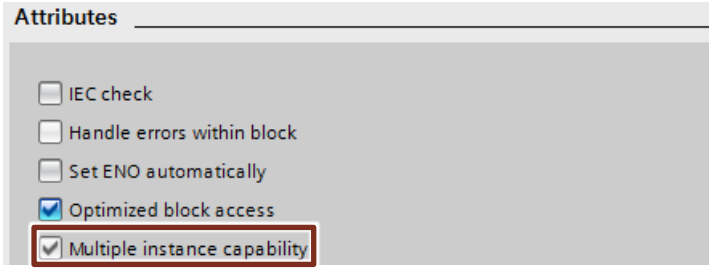
- Bausteine können universell an beliebigen Stellen des Anwenderprogramms eingesetzt werden.
- Bausteine können universell in unterschiedlichen Projekten angewendet werden.
- Wenn jeder Baustein eine eigenständige Aufgabe erhält, entsteht automatisch ein übersichtliches und gut strukturiertes Anwenderprogramm.
- Es entstehen deutlich weniger Fehlerquellen.
- Einfache Fehlerdiagnose möglich.

Empfehlung

Wenn Sie Bausteine wiederverwenden wollen, beachten Sie folgende Empfehlungen:

- Sehen Sie Bausteine immer als gekapselte Funktionen an. D.h. jeder Baustein repräsentiert eine abgeschlossene Teilaufgabe innerhalb des gesamten Anwenderprogramms.
- Setzen Sie mehrere zyklische Main-OBs ein, um Anlagenteile zu gruppieren.
- Führen Sie einen Datenaustausch zwischen den Bausteinen immer über deren Schnittstellen und nicht über deren Instanzen durch (siehe Kapitel 3.4.1 Bausteinschnittstellen als Datenaustausch).
- Verwenden Sie keine projektspezifischen Daten und vermeiden Sie folgende Bausteininhalte:
  - Zugriff auf globale DBs und Nutzung von Einzelinstanz-DBs
  - Zugriff auf Tags
  - Zugriff auf globale Konstanten
- Wiederverwendbare Bausteine haben die gleichen Voraussetzungen wie Know-how geschützte Bausteine in Bibliotheken. Prüfen Sie deshalb die Bausteine auf Wiederverwendbarkeit an Hand der Bausteineigenschaft "Multiinstanzfähig" ("Multiple instance capability"). Übersetzen Sie den Baustein vor der Prüfung.

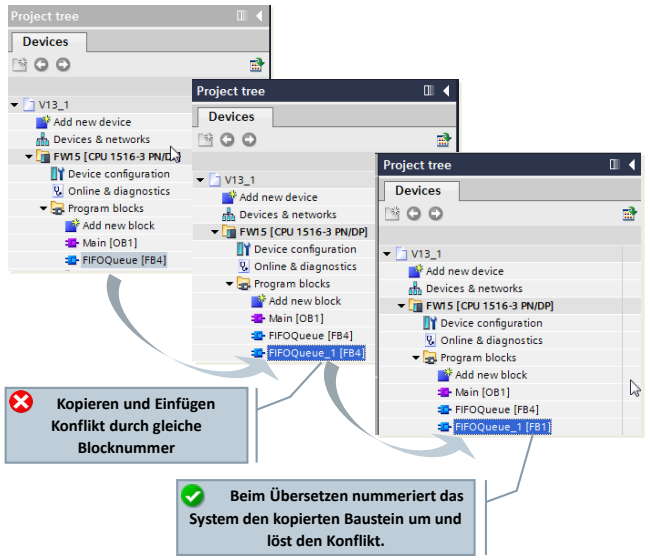
Abbildung 3-15: Bausteinattribute



3.2.10 Autonummerierung von Bausteinen

Zur internen Verarbeitung benötigte Blocknummern werden vom System automatisch (Einstellung in den Blockeigenschaften) vergeben.

Abbildung 3-16: Autonummerierung von Bausteinen



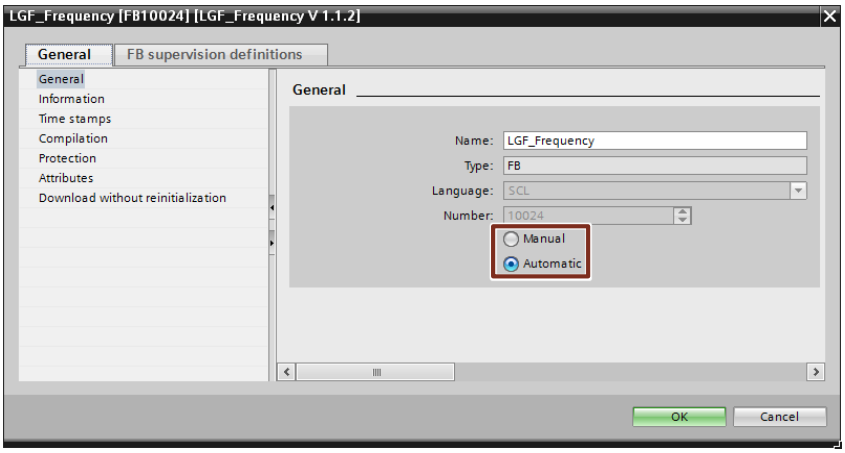
Vorteile

- Konflikte in Blocknummern, die z.B. durch Kopiervorgänge entstehen, löst das TIA Portal beim Übersetzen automatisch.

Empfehlung

- Lassen Sie die bestehende Einstellung "Automatic" unverändert.

Abbildung 3-17: Einstellung im Baustein



3.3 Bausteinschnittstellentypen

FBs und FCs verfügen über drei unterschiedliche Schnittstellentypen: In, InOut und Out. Über diese Schnittstellentypen werden die Bausteine mit Parametern versorgt. Im Baustein werden die Parameter verarbeitet und wieder ausgegeben. Durchgangsparameter (InOut) dienen sowohl der Übergabe von Daten an den aufgerufenen Baustein als auch der Rückgabe von Ergebnissen. Für die Parameterübergabe von Daten gibt es zwei unterschiedliche Möglichkeiten.

3.3.1 Übergabe per Wert (Call-by-value)

Beim Aufruf des Bausteins wird der Wert des Aktualparameters auf den Formalparameter des Bausteins kopiert. Hierfür wird zusätzlicher Speicher im aufgerufenen Baustein bereitgestellt.

Abbildung 3-18: Übergabe des Wertes

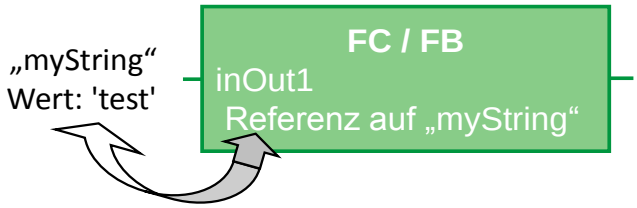
Eigenschaften

- Jeder Baustein zeigt das gleiche Verhalten mit übergebenen Parametern
- Werte werden bei Bausteinaufruf kopiert

3.3.2 Übergabe per Referenz (Call-by-reference)

Beim Aufruf des Bausteins wird eine Referenz auf die Adresse des Aktualparameters übergeben. Hierfür wird kein zusätzlicher Speicher benötigt.

Abbildung 3-19: Referenzieren des Aktualparameters (Zeiger auf die Datenablage des Parameters)



Eigenschaften

- Jeder Baustein zeigt das gleiche Verhalten mit referenzierten Parametern.
- Aktualparameter werden bei Bausteinaufruf referenziert, d.h. die Werte des Aktualparameters werden beim Zugriff direkt gelesen bzw. beschrieben

Empfehlung

- Nutzen Sie bei strukturierten Variablen (z.B. vom Typ ARRAY, STRUCT, STRING, …) generell den InOut-Schnittstellentyp, um den benötigten Datenspeicher nicht unnötig zu vergrößern.

3.3.3 Übersicht zur Übergabe von Parametern

Die folgende Tabelle gibt zusammenfassend einen Überblick wie S7-1200/1500 Bausteinparameter mit elementarem bzw. strukturiertem Datentyp übergeben werden.

Tabelle 3-5: Übersicht zur Übergabe von Parametern

Bausteintyp / Formalparameter		Elementarer Datentyp	Strukturierter Datentyp
FC	Input	Kopie	**Referenz**
	Output	Kopie	**Referenz**
	InOut	Kopie	**Referenz**
FB	Input	Kopie	Kopie
	Output	Kopie	Kopie
	InOut	Kopie	**Referenz**

Hinweis Wenn beim Bausteinaufruf optimierte Daten an einen Baustein mit der Eigenschaft "**nicht** optimierter Zugriff" übergeben werden, werden diese grundsätzlich als Kopie übergeben. Wenn der Baustein viele strukturierte Parameter enthält, kann das schnell dazu führen, dass der temporäre Speicherbereich (Lokaldaten-Stack) des Bausteins überläuft.

Das können Sie vermeiden, indem Sie für beide Bausteine dieselbe Zugriffsart einstellen (siehe Kapitel 2.6.5 Parameterübergabe zwischen Bausteinen mit optimiertem und nicht optimierten Zugriff).

3.4 Speicherkonzept

Bei STEP 7 gibt es generell den Unterschied zwischen globalem und lokalem Speicherbereich. Der globale Speicherbereich ist für jeden Baustein im Anwenderprogramm verfügbar. Der lokale Speicherbereich ist nur innerhalb des jeweiligen Bausteins verfügbar.

3.4.1 Bausteinschnittstellen als Datenaustausch

Wenn Sie Funktionen kapseln und den Datenaustausch zwischen den Bausteinen nur über die Schnittstellen programmieren, erhalten Sie deutliche Vorteile.

Vorteile

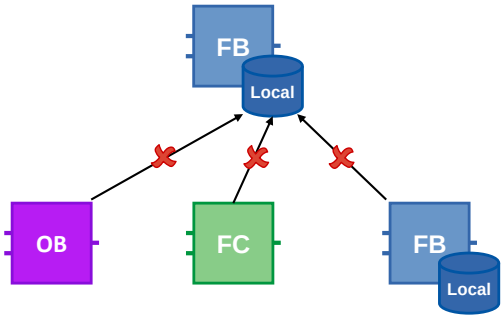
- Programm kann modular aus fertigen Bausteinen mit Teilaufgaben zusammengesetzt werden.
- Programm ist einfach erweiterbar und wartbar.
- Programmcode ist leichter lesbar und testbar, da es keine verdeckten Querzugriffe gibt.

Empfehlung

- Nutzen Sie möglichst nur lokale Variablen. Somit können die Bausteine universell und modular eingesetzt werden.

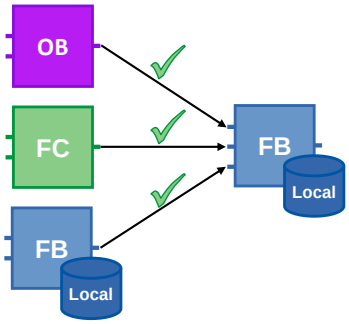
- Nutzen Sie den Datenaustausch über die Bausteinschnittstellen (In, Out, InOut), dadurch wird die Wiederverwendbarkeit der Bausteine gewährleistet.
- Nutzen Sie Instanzdatenbausteine nur als lokale Speicher für den jeweiligen Funktionsbaustein. Andere Bausteine sollten nicht in Instanzdatenbausteine schreiben.

Abbildung 3-20: Zugriffe auf Instanzdatenbausteine vermeiden



Wenn nur die Schnittstellen der Bausteine zum Datenaustausch verwendet werden, ist gewährleistet, dass alle Bausteine unabhängig voneinander eingesetzt werden können.

Abbildung 3-21: Bausteinschnittstellen für Datenaustausch



3.4.2 Globaler Speicher

Speicher wird als global bezeichnet, wenn der Zugriff von jeder Stelle des Anwenderprogramms aus möglich ist. Es gibt hardwareabhängige Speicher (z.B. Merker, Zeiten, Zähler, usw) und Global-DBs. Bei hardwareabhängigen Speicherbereichen besteht die Gefahr, dass das Programm unter Umständen nicht auf jede Steuerung portierbar ist, weil die Bereiche dort evtl. schon genutzt werden. Verwenden Sie deshalb Global-DBs anstelle hardwareabhängiger Speicherbereiche.

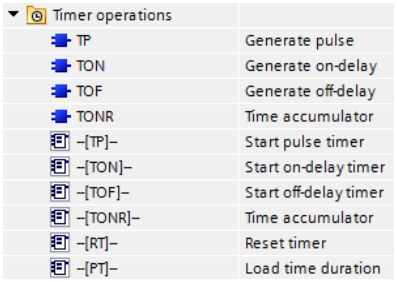
Vorteile

- Anwenderprogramme sind universell und unabhängig von der Hardware einsetzbar.
- Anwenderprogramm kann modular aufgebaut werden, ohne dass Merkerbereiche für verschiedene Anwender aufgeteilt werden müssen.
- Optimierte Global-DBs sind deutlich leistungsfähiger als der Merkerbereich, der aus Kompatibilitätsgründen nicht optimiert ist.

Empfehlung

- Verwenden Sie keine Merker und nutzen Sie stattdessen Global-DBs.
- Verzichten Sie auf hardwareabhängige Speicher, wie beispielsweise Taktmerker oder Zähler. Stattdessen nutzen Sie die IEC-Zähler und Timer in Verbindung mit Multiinstanzen (siehe Kapitel 3.2.5 Multiinstanzen). Die IEC-Zähler finden Sie unter "Anweisungen – Einfache Anweisungen - Zähler" ("Instructions – Basic Instructions – Timer operations").

Abbildung 3-22: IEC-Zeiten



3.4.3 Lokaler Speicher

- Statische Variablen
- Temporäre Variablen

Empfehlung

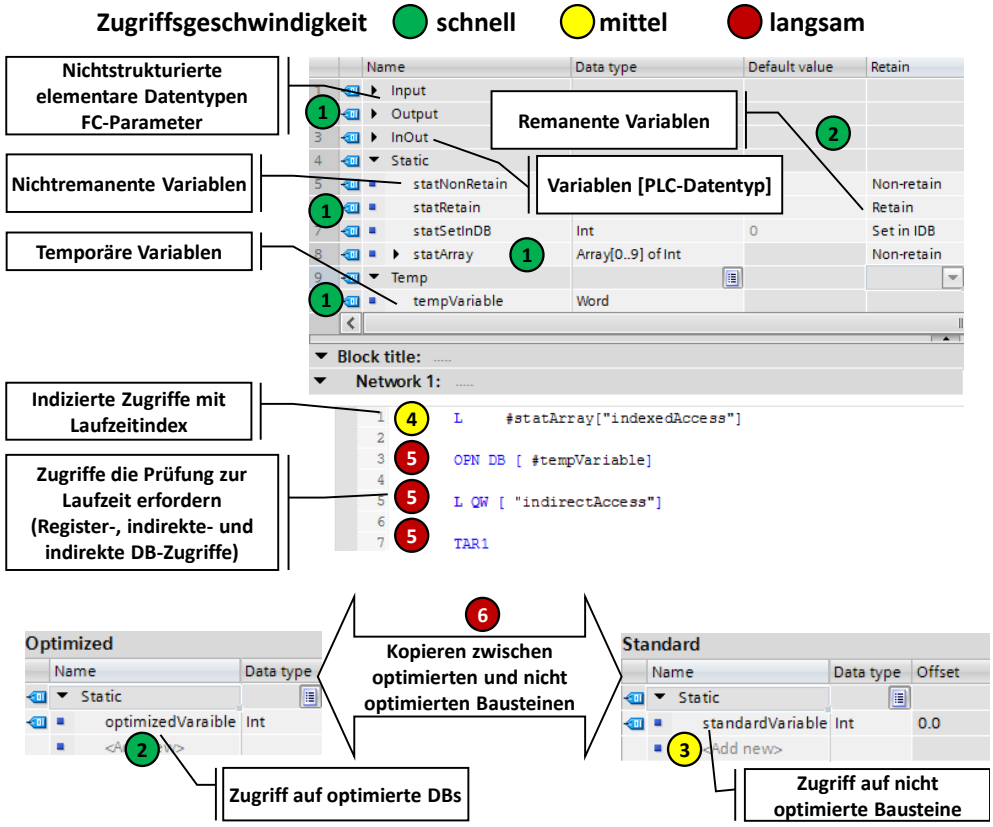
- Nutzen Sie statische Variablen für Werte die im nächsten Zyklus benötigt werden.
- Nutzen Sie temporäre Variablen im aktuellen Zyklus als Zwischenspeicher. Die Zugriffszeit ist bei temporären Variablen kürzer als bei Statischen.
- Falls sehr häufig auf eine Input/Output Variablen zugegriffen wird, nutzen sie eine temporäre Variable als Zwischenspeicher, um Laufzeit einzusparen.

Hinweis Optimierte Bausteine: Temporäre Variablen werden in jedem Aufruf des Bausteins mit dem "Defaultwert" initialisiert (S7-1500 / S7-1200 ab Firmware V4).
Nicht optimierte Bausteine: Temporäre Variablen sind bei jedem Aufruf des Bausteins undefiniert.

3.4.4 Zugriffsgeschwindigkeit von Speicherbereichen

STEP 7 bietet unterschiedliche Möglichkeiten an Speicherzugriffen. Systembedingt gibt es schnellere und langsamere Zugriffe für die verschiedenen Speicherbereiche.

Abbildung 3-23: Unterschiedliche Speicherzugriffe



Schnellste Zugriffe in der S7-1200/1500 in absteigender Reihenfolge

1. Optimierte Bausteine: Temporäre Variablen, Parameter eines FCs und FBs, nichtremanente statische Variablen, Variablen [PLC-Datentyp]
2. Optimierte Bausteine deren Zugriffe zur Kompilierung bekannt sind:
   - Remanente FB-Variablen
   - Optimierte Global-DBs
3. Zugriff auf nicht optimierte Bausteine
4. Indizierte Zugriffe mit zur Laufzeit berechnetem Index (z.B. `Motor [i]`)
5. Zugriffe, die Prüfungen zur Laufzeit erfordern
   - Zugriffe auf DBs, die zur Laufzeit erstellt werden oder indirekt geöffnet wurden (z.B. `OPN DB[i]`)
   - Registerzugriff oder indirekter Speicherzugriff
6. Kopieren von Strukturen zwischen optimierten und nicht optimierten Bausteinen (außer Array of Bytes)

3.5 Remanenz

Bei Ausfall der Versorgungsspannung kopiert die Steuerung mit ihrer Pufferenergie die remanenten Daten vom Arbeitsspeicher der Steuerung auf einen nicht flüchtigen Speicher. Nach Neustarten der Steuerung wird mit den remanenten Daten die Programmabarbeitung wieder aufgenommen. Je nach Steuerung ist die Datenmenge für Remanenz unterschiedlich groß.

Tabelle 3-6: Remanenter Speicher bei S7-1200/1500

Steuerung	Nutzbarer remanenter Speicher für Merker, Zeiten, Zähler, DBs und Technologieobjekte
CPU 1211C,1212C, 1214C, 1215C, 1217C	10 kByte
CPU 1511-1 PN	88 kByte
CPU 1513-1 PN	88 kByte
CPU 1515-2 PN, CPU 1516-3 PN/DP	472 kByte
CPU 1518-4 PN/DP	768 kByte

Tabelle 3-7: Unterschiede bei S7-1200 und S7-1500

S7-1200	S7-1500
Remanenz einstellbar **nur** für Merker	Remanenz einstellbar für Merker, Zeiten und Zähler

Vorteile

- Remanente Daten behalten ihren Wert, wenn die Steuerung in STOP und wieder in RUN geht, bzw. bei Stromausfall und Wiederanlauf der Steuerung.

Eigenschaften

Für elementare Variablen eines optimierten DB kann die Remanenz separat eingestellt werden. Nicht optimierte Datenbausteine können nur komplett remanent oder nicht remanent definiert werden.

Remanente Daten können mit den Aktionen "Urlöschen" oder "Rücksetzen auf Werkseinstellungen" gelöscht werden über:

- Betriebsschalter an der Steuerung (MRES)
- Display der Steuerung
- Online über STEP 7 (TIA Portal)

Empfehlung

- Verzichten Sie auf die Einstellung "Im IDB setzen" ("Set in IDB"). Stellen Sie remanente Daten immer im Funktionsbaustein ein und nicht im Instanz-Datenbaustein.
  Die Einstellung "Im IDB setzen" ("Set in IDB") erhöht die Abarbeitungszeit des Programmablaufs. Wählen Sie für die Schnittstellen im FB immer entweder "Nicht remanent" ("Non-retain") oder "Remanent" ("Retain").

Abbildung 3-24: Programmeditor (Funktionsbausteinschnittstellen)

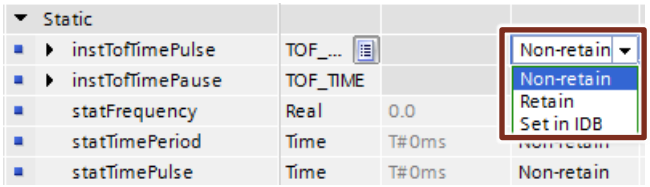
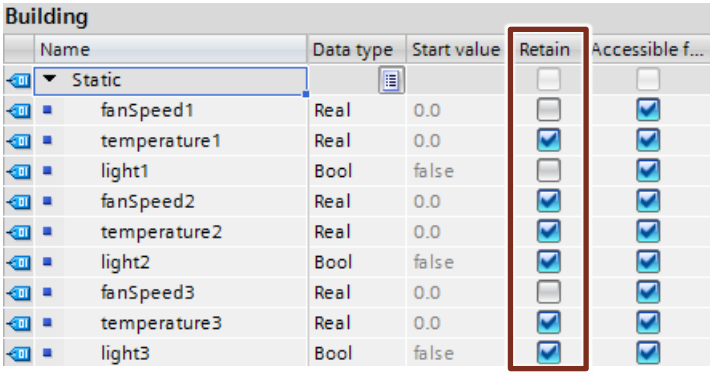


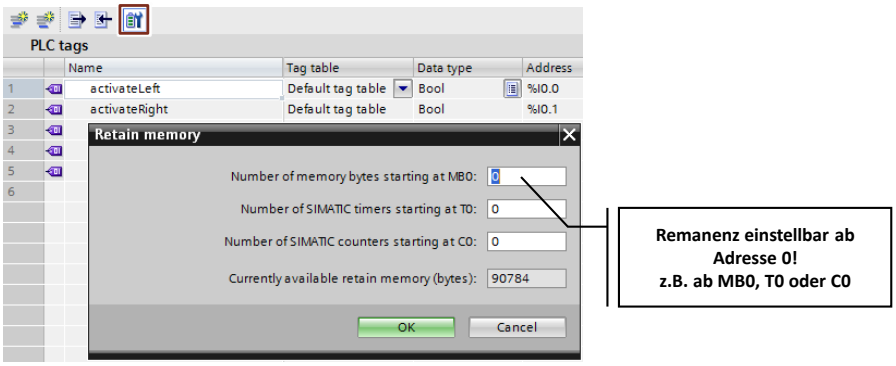
Abbildung 3-25: Programmeditor (Datenbaustein)



Beispiel: Remanenz in PLC-Variablen

Die Einstellung der remanenten Daten erfolgt in den Tabellen der PLC-Variablen, Funktionsbausteine und Datenbausteine.

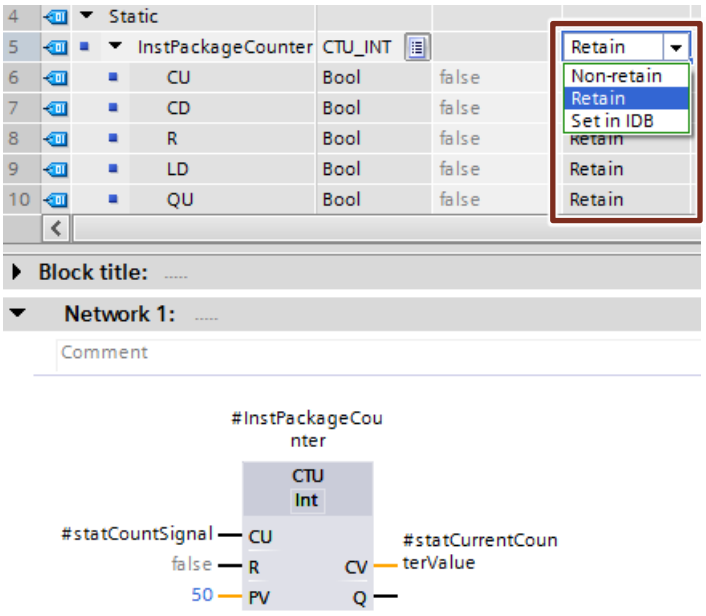
Abbildung 3-26: Einstellung der remanenten Variablen in Tabelle von PLC-Variablen



Beispiel: Remanenter Zähler

Sie können auch Instanzen von Funktionen (Timer, Zähler, usw.) remanent deklarieren. Wie schon im Kapitel 3.2.5 Multiinstanzen beschrieben, sollten Sie solche Funktionen immer als Multiinstanz programmieren.

Abbildung 3-27: Remanenter Zähler als Multiinstanz



Hinweis

Falls der remanente Speicher auf der SPS nicht ausreicht, besteht auch die Möglichkeit, Daten in Form von Datenbausteinen, die sich nur im Ladespeicher der SPS befinden, abzulegen. Der folgende Beitrag ist am Beispiel einer S7-1200 beschrieben. Diese Programmierung funktioniert auch für S7-1500.

Weitere Informationen finden Sie in folgenden Beiträgen:

Wie können Sie in STEP 7 (TIA Portal) Datenbausteine mit dem Attribut "Nur im Ladespeicher ablegen" für eine S7-1200 projektieren?
https://support.industry.siemens.com/cs/ww/de/view/53034113

Verwendung von Rezeptfunktionen für persistente Daten mit SIMATIC S7-1200 und S7-1500
https://support.industry.siemens.com/cs/ww/de/view/109479727

3.6 Symbolische Adressierung

3.6.1 Symbolische statt absolute Adressierung

Das TIA Portal ist für symbolische Programmierung optimiert. Dadurch ergeben sich für Sie viele Vorteile. Durch symbolische Adressierung können Sie programmieren, ohne auf die interne Datenablage zu achten. Die Steuerung kümmert sich darum, wo die Daten speicheroptimal abgelegt werden. Sie können sich dadurch komplett auf die Lösung Ihrer Applikationsaufgabe konzentrieren.

Vorteile

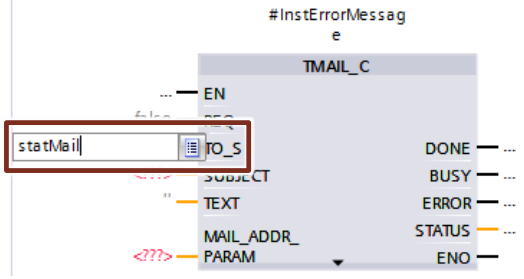
- Leichter lesbare Programme durch symbolische Variablennamen
- Automatische Aktualisierung der Variablennamen an allen Verwendungsstellen im Anwenderprogramm
- Speicherablage der Programmdaten muss nicht manuell verwaltet werden (absolute Adressierung)
- Leistungsfähiger Datenzugriff
- Keine manuellen Optimierungen aus Leistungs– oder Programmgrößengründen notwendig
- Autovervollständigung für schnelle Symboleingabe
- Weniger Programmfehler durch Typsicherheit (Gültigkeit von Datentypen wird bei allen Zugriffen geprüft)

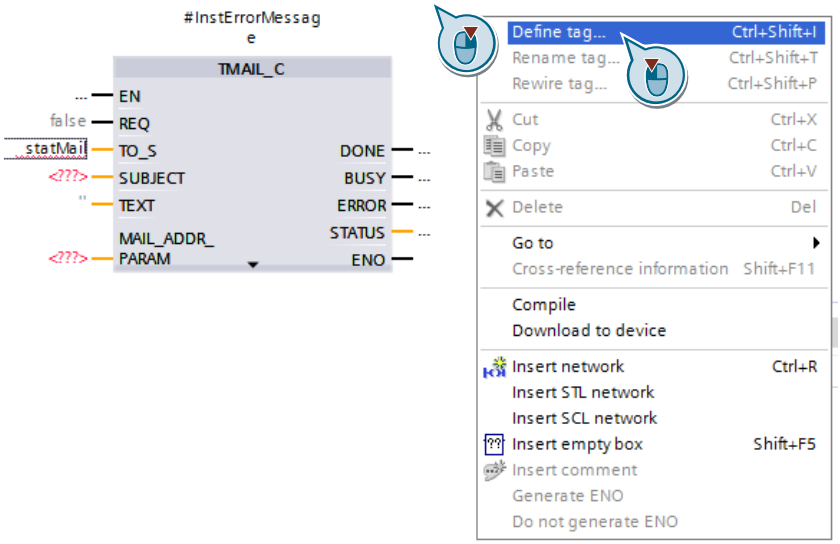
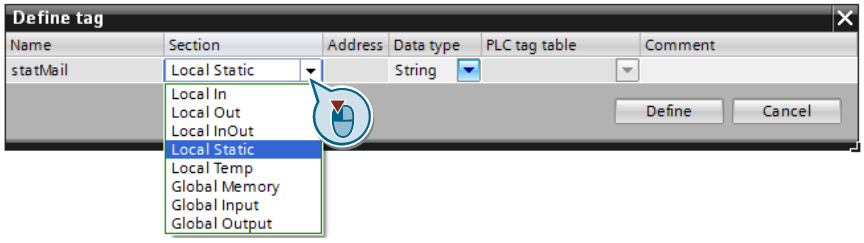
Empfehlung

- "Machen Sie sich keine Gedanken über die Ablage der Daten"
- "Denken" Sie symbolisch. Geben Sie jeder Funktion, Variablen oder Daten "sprechende" Namen, wie z.B. Pumpe_Kessel_1, Heizung_Raum_4, usw. Somit kann ein erstelltes Programm einfach gelesen werden, ohne viele Kommentare zu benötigen.
- Geben Sie allen verwendeten Variablen direkt eine symbolische Bezeichnung und definieren Sie diese anschließend mit Rechtsklick.

Beispiel

Tabelle 3-8: Beispiel zum Erstellen von symbolischen Variablen

Schritt	Anweisung
1.	Öffnen Sie den Programmeditor und öffnen Sie einen beliebigen Baustein.
2.	Geben Sie an einem Eingang einer Anweisung direkt einen symbolischen Namen ein. 

Schritt	Anweisung
3.	Klicken Sie mit der rechten Maustaste neben den Baustein und wählen Sie im Kontextmenü "Variable definieren…" ("Define tag…"). 
4.	Definieren Sie die Variable. 

Falls Sie mehrere Variablen in einem Netzwerk definieren wollen, gibt es eine elegante Methode, Zeit zu sparen: Vergeben Sie zuerst alle Variablennamen. Definieren Sie dann alle Variablen mit dem Dialog von Schritt 4 gleichzeitig.

Hinweis

Weitere Informationen finden Sie in folgendem Beitrag:

Warum ist die durchgängige Definition und Nutzung von Symbolen in STEP 7 (TIA Portal) für die S7-1500 obligatorisch?
https://support.industry.siemens.com/cs/ww/de/view/67598995

3.6.2 Datentyp ARRAY und indirekte Feldzugriffe

Der Datentyp ARRAY repräsentiert eine Datenstruktur, die sich aus mehreren Elementen eines Datentyps zusammensetzt. Der Datentyp ARRAY eignet sich z.B. für die Ablage von Rezepturen, Materialverfolgung in einer Warteschlange, zyklische Prozesswerterfassung, Protokolle, usw.

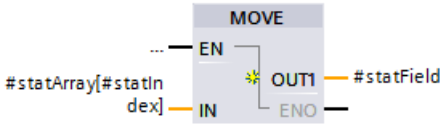
Abbildung 3-28: ARRAY mit 10 Elementen vom Datentyp Integer (INT)

Name	Data type
statArray	Array[0..9] of Int
statArray[0]	Int
statArray[1]	Int
statArray[2]	Int
statArray[3]	Int
statArray[4]	Int
statArray[5]	Int
statArray[6]	Int
statArray[7]	Int
statArray[8]	Int
statArray[9]	Int

Mit einem Index (`array ["index"]`) können Sie auf einzelne Elemente im ARRAY indirekt zugreifen.

Abbildung 3-29: Indirekter Feldzugriff

KOP / FUP:



SCL:

```
1  #statField := #statArray[#statIndex];
```

Vorteile

- Einfacher Zugriff, durch ARRAY-Index
- Keine umständliche Zeigererstellung notwendig
- Schnelle Erstellung und Erweiterung möglich
- Nutzbar in allen Programmiersprachen

Eigenschaften

- Strukturierter Datentyp
- Datenstruktur aus fester Anzahl von Elementen des gleichen Datentyps
- ARRAYs können auch mehrdimensional angelegt werden
- Indirekter Zugriff mit Laufvariable mit dynamischer Indexberechnung zur Laufzeit möglich

Empfehlung

- Benutzen Sie ARRAY für indizierte Zugriffe statt Zeiger (z.B. ANY-Pointer). Dadurch wird das Programm leichter lesbar, da ein ARRAY mit einem

symbolischen Namen aussagekräftiger ist als ein Zeiger in einen Speicherbereich.

- Nutzen Sie als Laufvariable den Datentyp DINT als temporäre Variable für höchste Leistungsfähigkeit.
- Nutzen Sie die Anweisung "MOVE_BLK", um Teile eines ARRAYs in ein anderes zu kopieren.
- Nutzen Sie die Anweisung "GET_ERR_ID", um Zugriffsfehler innerhalb des Arrays abzufangen.

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Wie kann ein Array-Zugriff bei einer S7-1500 mit variablem Index realisiert werden?
https://support.industry.siemens.com/cs/ww/de/view/67598676

Wie können Sie in STEP 7 (TIA Portal) sicher und indirekt adressieren?
https://support.industry.siemens.com/cs/ww/de/view/97552147

Wie können Sie in STEP 7 (TIA Portal) für die S7-1500 Daten zwischen zwei Variablen vom Datentyp "Array of Bool" und "Word" transferieren?
https://support.industry.siemens.com/cs/ww/de/view/108999241

3.6.3 Formalparameter Array [*] (ab V14)

Mit dem Formalparameter Array [*] können Arrays mit variabler Länge an Funktionen und Funktionsbausteine übergeben werden.

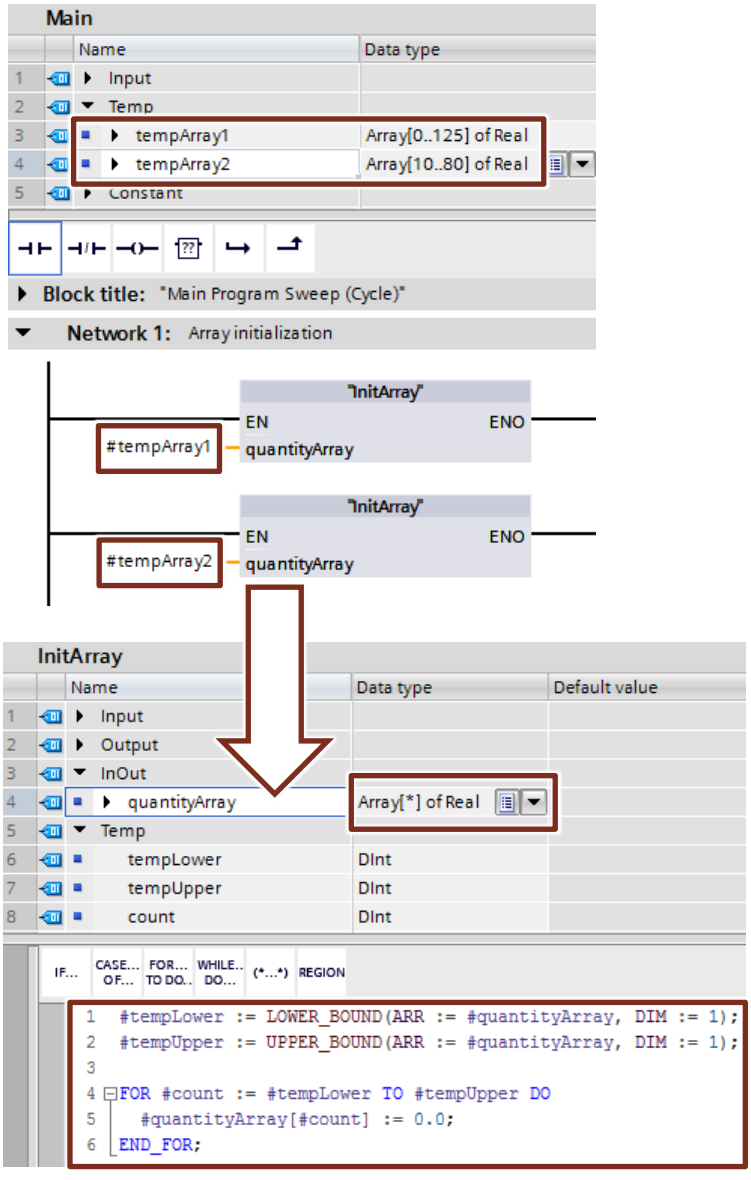
Mit den Anweisungen "LOWER_BOUND" und "UPPER_BOUND" können die Array-Grenzen ermittelt werden.

Vorteile

- Bausteine, die flexibel Arrays unterschiedlicher Länge verarbeiten können
- Optimale Lesbarkeit durch vollsymbolische Programmierung
- Keine Pointer-Programmierung für Arrays unterschiedlicher Länge mehr notwendig

Beispiel

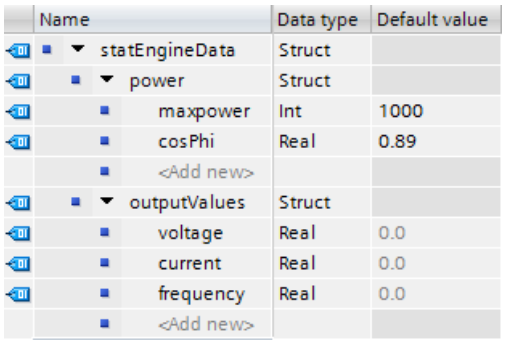
Abbildung 3-30:Unterschiedliche Arrays initialisieren



3.6.4 Datentyp STRUCT und PLC-Datentypen

Der Datentyp STRUCT repräsentiert eine Datenstruktur, die sich aus Elementen unterschiedlicher Datentypen zusammensetzt. Die Deklaration einer Struktur erfolgt im jeweiligen Baustein.

Abbildung 3-31: Struktur mit Elementen unterschiedlichen Datentyps

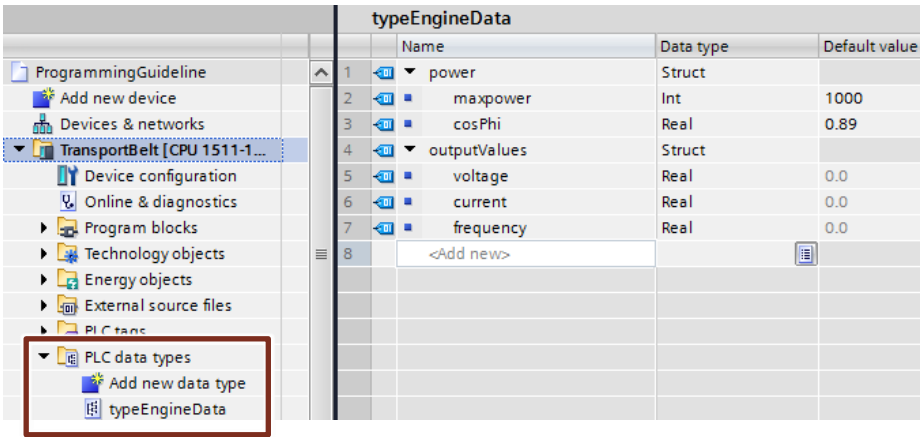


Name	Data type	Default value
statEngineData	Struct	
power	Struct	
maxpower	Int	1000
cosPhi	Real	0.89
<Add new>		
outputValues	Struct	
voltage	Real	0.0
current	Real	0.0
frequency	Real	0.0
<Add new>		

Im Unterschied zu Strukturen werden PLC-Datentypen steuerungsweit im TIA Portal definiert und können zentral geändert werden. Alle Verwendungsstellen werden automatisch aktualisiert.

PLC Datentypen werden vor ihrer Verwendung in dem Ordner "PLC-Datentypen" ("PLC data types") in der Projektnavigation deklariert.

Abbildung 3-32: PLC-Datentyp (PLC data types)



Vorteile

- Eine Änderung in einem PLC-Datentyp wird an allen Verwendungsstellen im Anwenderprogramm automatisch aktualisiert.
- Einfacher Datenaustausch über Bausteinschnittstellen zwischen mehreren Bausteinen
- In PLC-Datentypen können STRING-Variablen mit definierter Länge deklariert werden (z.B. String[20]). Ab TIA V14 kann für die Länge auch eine globale Konstante verwendet werden (z.B. String[LENGTH]).
  Falls eine STRING-Variable ohne definierte Länge deklariert wird, hat die Variable die maximale Länge von 255 Zeichen.

Eigenschaften

- PLC-Datentypen enden immer an WORD-Grenzen (siehe folgende Abbildung).
- Beachten Sie diese Systemeigenschaft, wenn Sie …
  - Strukturen auf E/A-Bereiche verwenden (siehe Kapitel 3.6.5 Zugriff mit PLC-Datentypen auf E/A-Bereiche).
  - Telegramme mit PLC-Datentypen zur Kommunikation verwenden.
  - Parametersätze mit PLC-Datentypen für Peripherie verwenden.
  - Nicht optimierte Bausteine und absolute Adressierung verwenden.

Abbildung 3-33: PLC-Datentypen enden immer an WORD-Grenzen

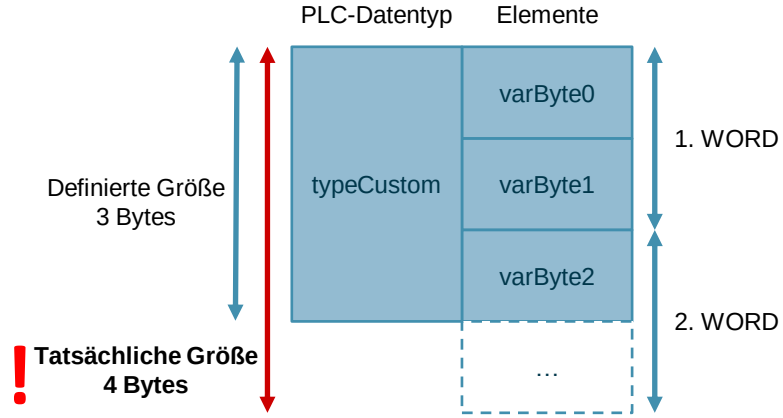
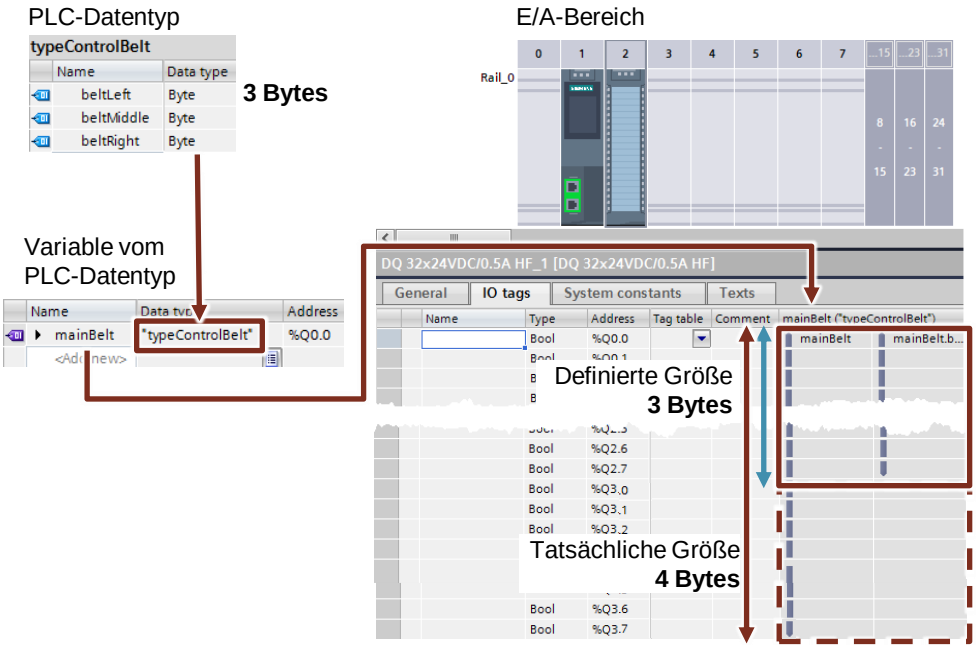


Abbildung 3-34: PLC-Datentyp auf E/A Bereich



Empfehlung

- Nutzen Sie PLC-Datentypen zum Zusammenfassen von mehreren zusammengehörigen Daten, wie z.B. Telegramme oder Daten eines Motors (Sollwert, Drehzahl, Drehrichtung, Temperatur, usw.)

- Nutzen Sie immer PLC-Datentypen statt Strukturen für die mehrfache Verwendung im Anwenderprogramm.
- Nutzen Sie PLC-Datentypen zum Strukturieren in Datenbausteinen.
- Nutzen Sie PLC-Datentypen, um eine Struktur für einen Datenbaustein festzulegen. Der PLC-Datentyp kann auf beliebig viele DBs angewendet werden. Sie können einfach und komfortabel beliebig viele DBs der gleichen Struktur anlegen und zentral am PLC-Datentyp anpassen.

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Bibliothek mit PLC-Datentypen (LPD) für STEP 7 (TIA Portal) und S7-1200 / S7-1500
https://support.industry.siemens.com/cs/ww/de/view/109482396

Wie können Sie in STEP 7 (TIA Portal) Strukturen in optimierten Speicherbereichen bei der S7-1500 initialisieren?
https://support.industry.siemens.com/cs/ww/de/view/78678760

Wie legen Sie bei einer S7-1500 Steuerung einen PLC-Datentyp an?
https://support.industry.siemens.com/cs/ww/de/view/67599090

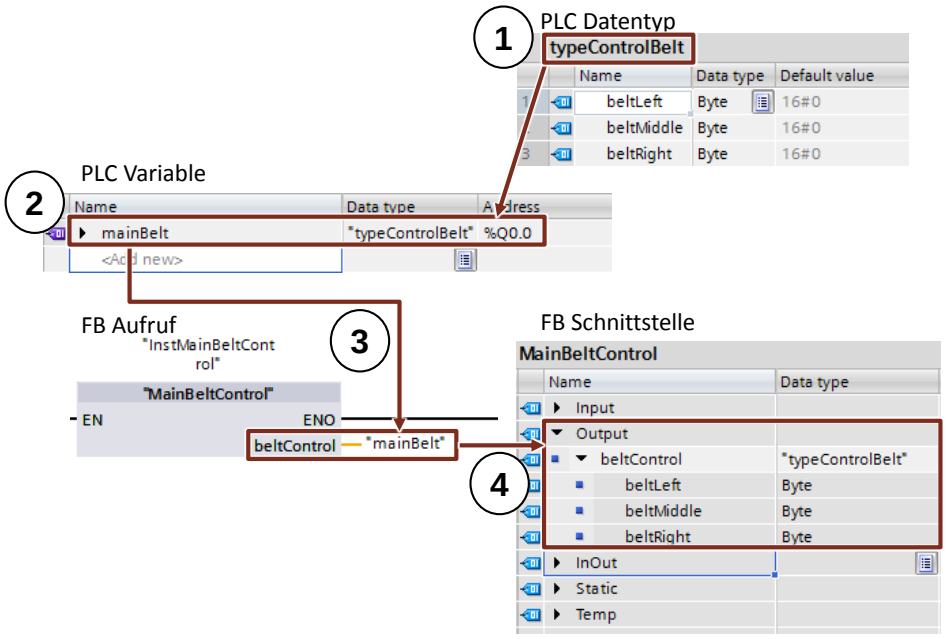
Wie erfolgt in STEP 7 (TIA Portal) die gezielte Anwendung eigener Datentypen UDT?
https://support.industry.siemens.com/cs/ww/de/view/67582844

Warum sollten für die S7-1500 beim Bausteinaufruf ganze Strukturen übergeben werden, anstelle vieler Einzelkomponenten?
https://support.industry.siemens.com/cs/ww/de/view/67585079

3.6.5 Zugriff mit PLC-Datentypen auf E/A-Bereiche

Mit S7-1500 Steuerungen können Sie PLC-Datentypen anlegen und dazu benutzen, auf Ein- und Ausgänge strukturiert und symbolisch zuzugreifen.

Abbildung 3-35: Zugriff mit PLC-Datentypen auf E/A Bereiche



1. PLC-Datentyp mit allen benötigten Daten
2. PLC-Variable vom Typ des erstellten PLC-Datentyps und Anfangsadresse des E/A-Datenbereichs (%Ix.0 oder %Qx.0 z.B. %I0.0, %Q12.0, …)
3. Übergabe der PLC-Variable als Aktualparameter an den Funktionsbaustein
4. Output des Funktionsbausteins ist vom Typ des erstellten PLC-Datentyps

Vorteile

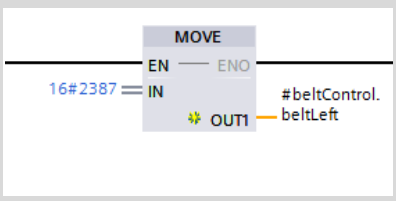
- hohe Programmiereffizienz
- einfache Mehrfachverwendbarkeit, dank PLC-Datentypen

Empfehlung

- Nutzen Sie für den Zugriff auf E/A-Bereiche PLC-Datentypen, um z.B. Antriebstelegramme symbolisch zu empfangen und zu senden.

Hinweis

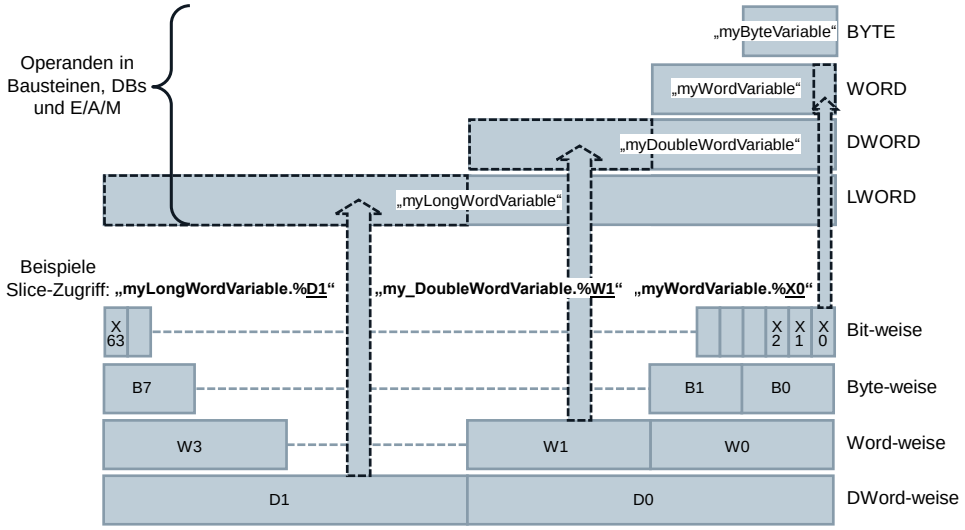
Auf einzelne Elemente eines PLC-Datentyps einer Variablen kann auch direkt im Anwenderprogramm zugegriffen werden:



3.6.6 Slice Zugriff

Bei S7-1200/1500 Steuerungen können Sie mit einem beliebigen Speicherbereich auf Variablen vom Datentyp Byte, Word, DWord oder LWord zugreifen. Die Aufteilung eines Speicherbereiches (z.B. Byte oder Word) in einem kleineren Speicherbereich (z.B. Bool) wird auch Slice genannt. Im folgenden Bild sind die symbolischen Bit-, Byte- und Wort-Zugriffe auf die Operanden dargestellt.

Abbildung 3-36:Symbolischer Bit, Byte, Word, DWord Slice-Zugriff



Vorteile

- hohe Programmiereffizienz
- keine zusätzliche Definition in der Variablendeklaration erforderlich
- einfacher Zugriff (z.B. Steuerbits)

Empfehlung

- Nutzen Sie den Slice-Zugriff anstatt über AT-Konstrukt auf bestimmte Datenbereiche in Operanden zuzugreifen.

Hinweis

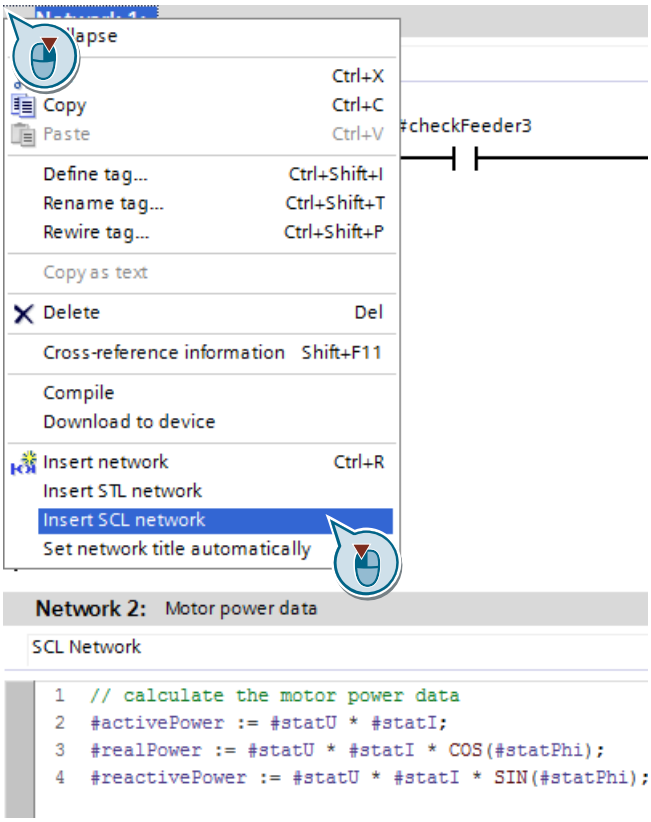
Weitere Informationen finden Sie in folgendem Beitrag:

Wie können Sie in STEP 7 (TIA Portal) bit-, byte- oder wortweise und symbolisch auf die unstrukturierten Datentypen zugreifen?
https://support.industry.siemens.com/cs/ww/de/view/57374718

3.6.7 SCL-Netzwerke in KOP und FUP (ab V14)

Mit SCL-Netzwerken können Sie in KOP und FUP Berechnungen durchführen, die mit KOP- und FUP-Anweisungen nur aufwendig zu programmieren sind.

Abbildung 3-37: SCL-Netzwerk einfügen



Vorteile

- Zeitersparnis durch effiziente Programmierung
- Übersichtlicher Code, dank symbolischer Programmierung

Eigenschaften

- Unterstützt alle SCL-Anweisungen
- Unterstützt Kommentare

Empfehlung

- Nutzen Sie für mathematische Berechnungen SCL-Netzwerke in KOP und FUP anstatt Anweisungen wie z.B. ADD, SUB, usw.

3.7 Bibliotheken

Mit dem TIA Portal können Sie aus unterschiedlichen Projektelementen eigene Bibliotheken aufbauen, die sich leicht wiederverwenden lassen.

Vorteile

- Einfache Ablage für die im TIA Portal projektierten Daten:
  - Komplette Geräte (Steuerung, HMI, Antrieb, usw.)
  - Bausteine, Variablentabellen, PLC-Datentypen, Beobachtungstabellen, usw.
  - HMI-Bilder, HMI-Variablen, Skripte, usw.
- Projektübergreifender Austausch durch Bibliotheken
- Zentrale Updatefunktion von Bibliothekselementen
- Versionierung von Bibliothekselementen
- Weniger Fehlerquellen bei der Verwendung von Steuerungsbausteinen durch systemunterstützte Berücksichtigung von Abhängigkeiten

Empfehlungen

- Erstellen Sie Kopiervorlagen für die einfache Wiederverwendung von Bausteinen, Hardwarekonfigurationen, HMI-Bilder, usw.
- Erstellen Sie Typen für die systemunterstützte Wiederverwendung von Bibliothekselementen:
  - Versionierung von Bausteinen
  - Zentrale Updatefunktion aller Verwendungsstellen
- Verwenden Sie globale Bibliotheken zum Austausch mit anderen Anwendern oder als zentrale Ablage zur gleichzeitigen Verwendung von mehreren Anwendern.
- Konfigurieren Sie den Ablageort ihrer globalen Bibliotheken, damit sie beim Starten des TIA Portals automatisch geöffnet werden.
  Weitere Informationen finden Sie unter:
  https://support.industry.siemens.com/cs/ww/de/view/100451450

Hinweis

Weitere Informationen finden Sie in folgenden Beiträgen:

Welche Elemente aus STEP 7 (TIA Portal) und WinCC (TIA Portal) können in einer Bibliothek als Typ oder als Kopiervorlage abgelegt werden?
https://support.industry.siemens.com/cs/ww/de/view/109476862

Wie können Sie in STEP 7 (TIA Portal) eine globale Bibliothek mit Schreibrechten öffnen?
https://support.industry.siemens.com/cs/ww/de/view/37364723

3.7.1 Arten von Bibliotheken und Bibliothekselemente

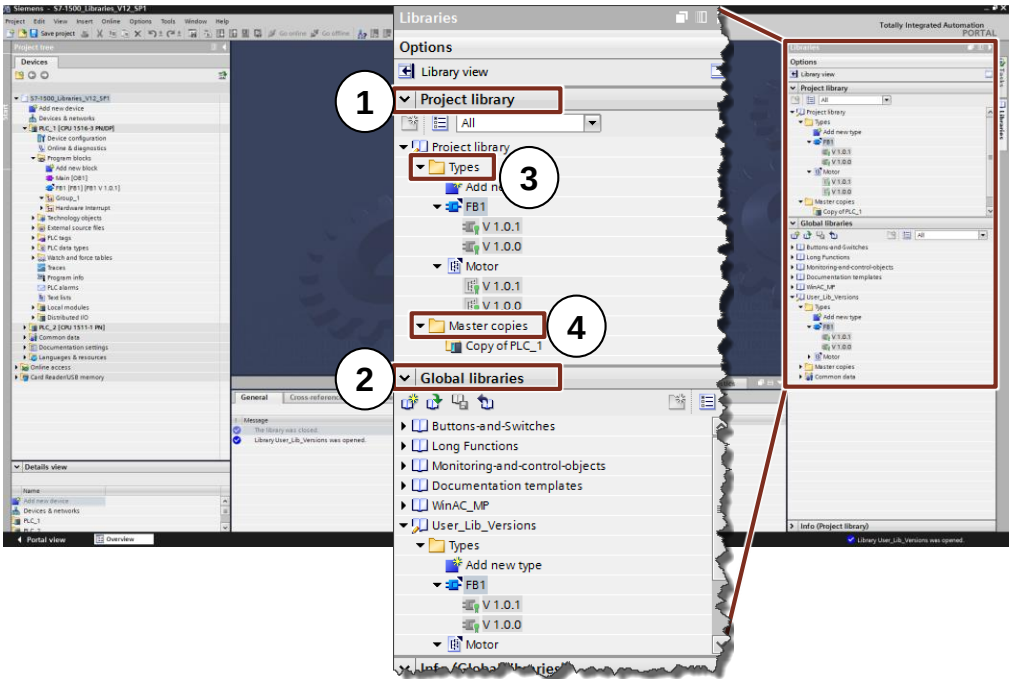
Generell gibt es zwei unterschiedliche Arten von Bibliotheken:

- "Projektbibliothek" ("Project library")
- "Globale Bibliothek" ("Global library")

Der Inhalt besteht jeweils aus zwei Ablagearten:

- "Typen" ("Types")
- "Kopiervorlagen" ("Master Copies")

Abbildung 3-38: Bibliotheken im TIA Portal



(1) "Projektbibliothek" ("Project library")

- Im Projekt integriert und wird mit dem Projekt verwaltet
- Ermöglicht die Wiederverwendung innerhalb des Projekts

(2) "Globale Bibliothek" ("Global library")

- Eigenständige Bibliothek
- Verwendung innerhalb von mehreren Projekten möglich

Eine Bibliothek beinhaltet zwei unterschiedliche Ablagearten von Bibliothekselementen:

(3) "Kopiervorlagen" ("Master copies")

- Kopien von Projektierungselementen in der Bibliothek (z.B. Bausteine, Hardware, PLC-Variablentabellen, usw.)
- Kopien sind mit den Elementen im Projekt nicht verbunden.
- Kopiervorlagen können aus mehreren Projektierungselementen bestehen.

(4) "Typen" ("Types")

- Typen sind mit ihren Verwendungsstellen im Projekt verbunden. Wenn Typen verändert werden, können die Verwendungsstellen im Projekt automatisch aktualisiert werden.

- Unterstützte Typen sind Steuerungsbausteine (FCs, FBs), PLC Datentypen, HMI-Bilder, HMI-Bildbausteine, HMI-UDT, Skripte).
- Unterlagerte Elemente werden automatisch mittypisiert.
- Typen sind versioniert: Änderungen können durch Erzeugung einer neuen Version durchgeführt werden.
- Innerhalb einer Steuerung kann es nur eine Version eines verwendeten Typs geben.

3.7.2 Typ-Konzept

Das Typ-Konzept ermöglicht die Erstellung von standardisierten Automatisierungsfunktionen, die Sie in mehreren Anlagen oder Maschinen einsetzen können. Das Typ-Konzept unterstützt Sie mit Versionierungs- und Aktualisierungsfunktionen.

Im Anwenderprogramm können Sie Typen aus der Bibliothek verwenden. Dadurch ergeben sich folgende Vorteile:

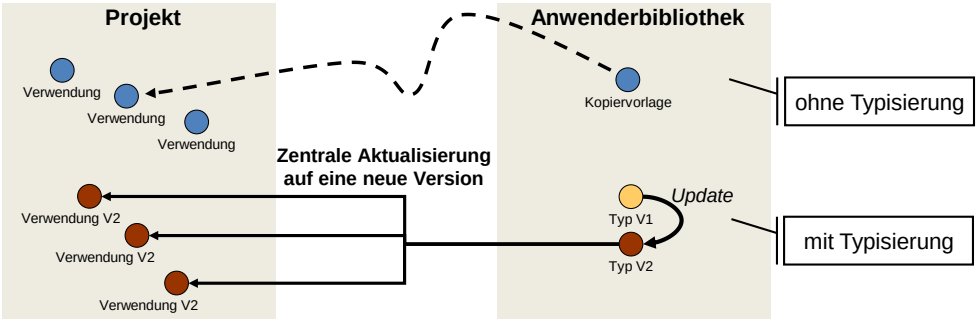
Vorteile

- Zentrale Aktualisierung aller Verwendungsstellen im Projekt
- Ungewollte Modifikationen an Verwendungsstellen von Typen sind nicht möglich.
- Das System garantiert, dass Typen immer konsistent bleiben, indem es ungewollte Löschoperationen verhindert.
- Wenn ein Typ gelöscht wird, werden alle Verwendungsstellen im Anwenderprogramm gelöscht.

Eigenschaften

Durch die Verwendung von Typen können Sie Änderungen zentral vornehmen und im kompletten Projekt aktualisieren.

Abbildung 3-39: Typisierung mit Anwenderbibliotheken



- Typen werden zur einfachen Erkennung immer gekennzeichnet

3.7.3 Unterschiede zwischen typisierbaren Objekten bei CPU und HMI

Systembedingt gibt es Unterschiede zwischen den typisierbaren Objekten bei Steuerungen und HMI:

Tabelle 3-9: Unterschiede Typen bei Steuerung und HMI

Steuerung	HMI
Unterlagerte Steuerungselemente werden mit typisiert.	Unterlagerte HMI Elemente werden **nicht** mit typisiert.
Unterlagerte Steuerungselemente werden mit instanziiert.	Unterlagerte HMI Elemente werden **nicht** mit instanziiert.
Steuerungselemente werden in einer **Testumgebung** editiert.	HMI Bilder und HMI Skripte werden in einer Testumgebung editiert. Faceplates und HMI - UDTs werden direkt in der Bibliothek **ohne Testumgebung** editiert.

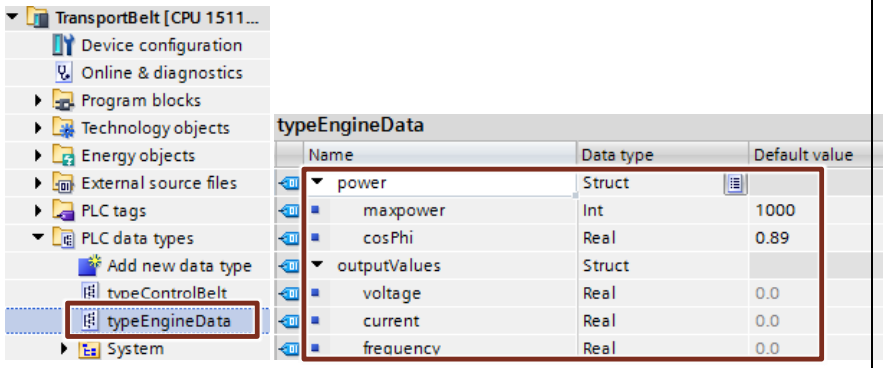
Weitere Informationen zum Umgang mit Bibliotheken finden Sie im folgenden Beispiel.

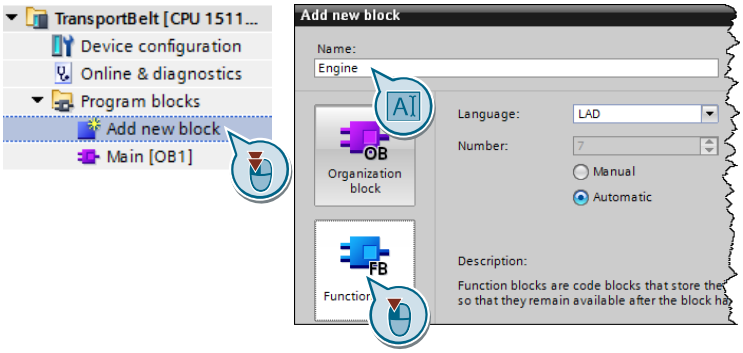
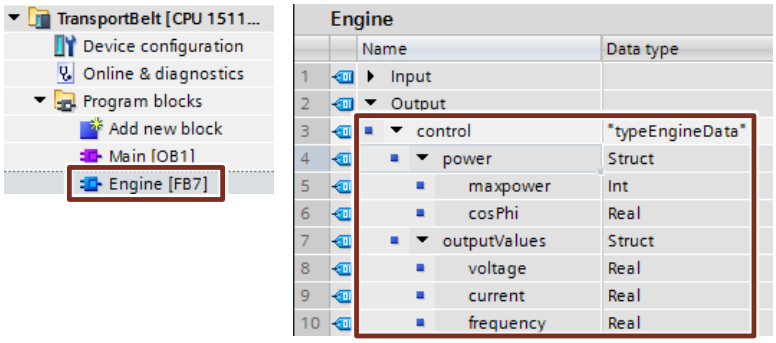
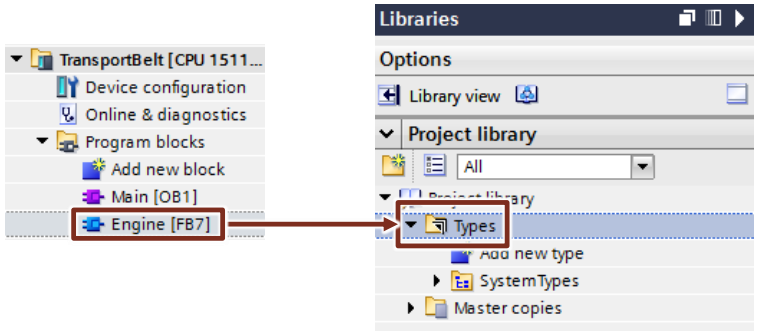
3.7.4 Versionierung eines Bausteins

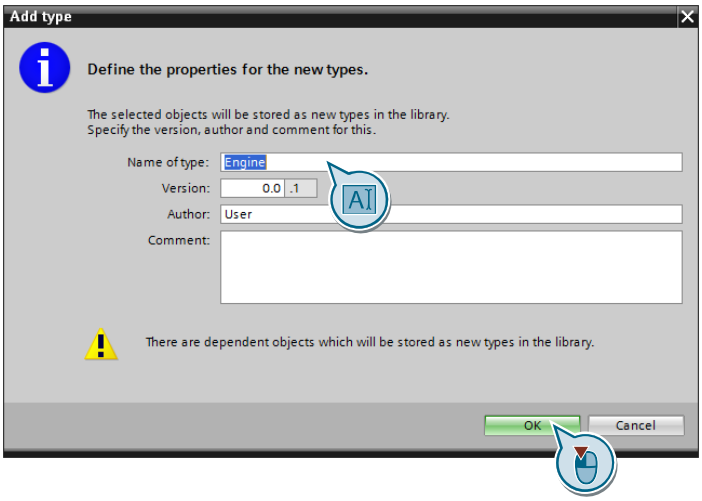
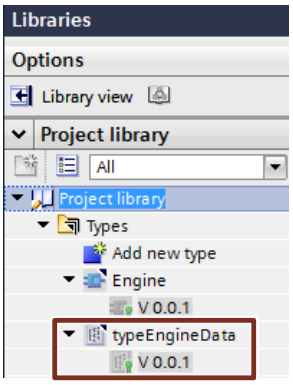
Beispiel: Erstellung eines Typen

Im folgenden Beispiel wird Ihnen gezeigt, wie grundlegenden Funktionen der Bibliotheken mit Typen angewendet werden.

Tabelle 3-10: Erstellung eines Typen

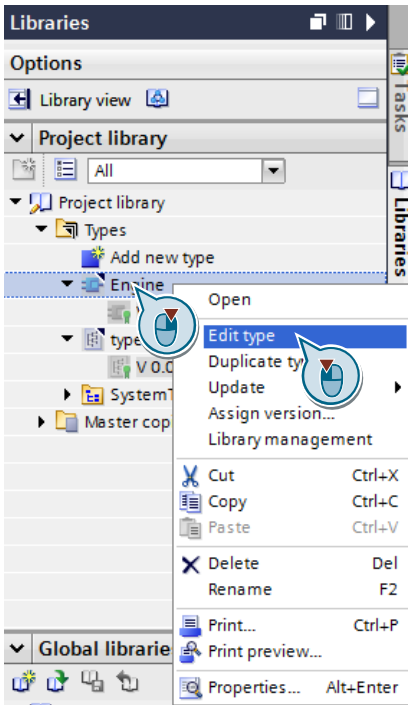
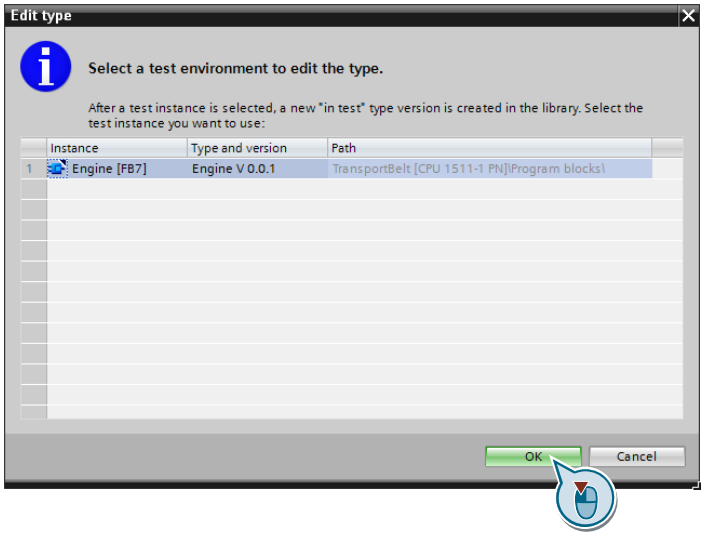
Schritt	Anweisung
1.	Erstellen Sie einen neuen PLC-Datentyp mit "Add new datatype" und erstellen Sie einige Variablen. Das ist später der unterlagerte Typ. 

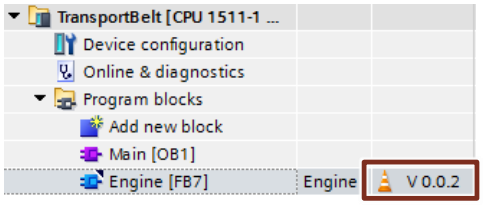
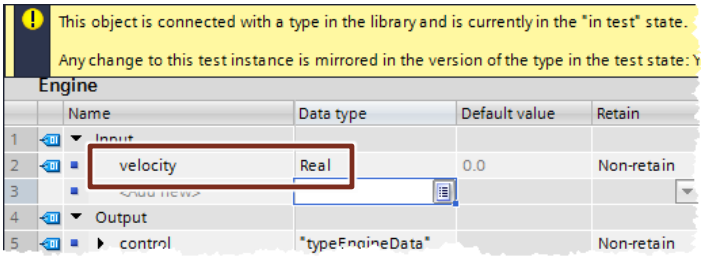
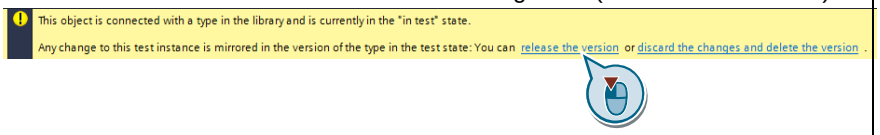
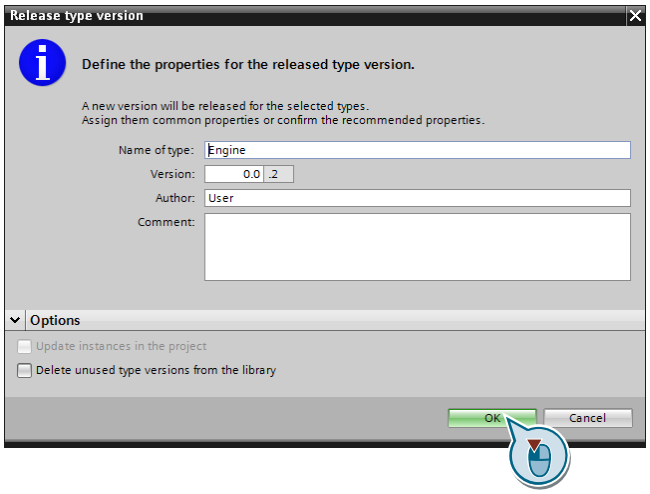
Schritt	Anweisung
2.	Erstellen Sie einen neuen Funktionsbaustein mit "Add new Block". Das ist der überlagerte Typ. 
3.	Definieren Sie eine Ausgangsvariable vom Datentyp, den Sie erstellt haben. Somit ist der PLC-Datentyp dem Funktionsbaustein unterlagert. 
4.	Ziehen Sie den Funktionsbaustein per Drag&Drop in den Ordner "Typen" ("Types") in der Projektbibliothek. 

Schritt	Anweisung
5.	Vergeben Sie optional: Typname, Version, Autor und Kommentar und bestätigen Sie den Dialog mit "OK". 
6.	Der unterlagerte PLC Datentyp wird automatisch in der Bibliothek mit abgelegt. 

Beispiel: Ändern eines Typen

Tabelle 3-11: Ändern eines Typen

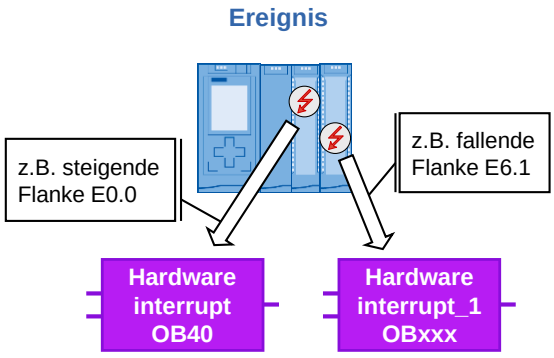
Schritt	Anweisung
1.	Machen Sie einen Rechtsklick auf den Baustein in der "Projektbibliothek" ("Project library") und wählen "Edit type". 
2.	Wählen Sie aus welche Steuerung als Testumgebung genutzt werden soll und bestätigen Sie den Dialog mit "OK".  Falls mehrere Steuerungen im Projekt den gewählten Baustein verwenden, muss eine Steuerung als Testumgebung ausgewählt werden.

Schritt	Anweisung
3.	Es öffnet sich der Baustein. Eine neue Version des Bausteins wird angelegt. 
4.	Fügen Sie eine Inputvariable hinzu. An dieser Stelle haben Sie die Möglichkeit die Änderung am Baustein zu testen, indem Sie das Projekt auf eine Steuerung laden. Wenn Sie mit Testen des Bausteins fertig sind, fahren Sie fort mit folgenden Schritten. 
5.	Klicken Sie auf die Schaltfläche "die Version freigeben" ("release the version"). 
6.	Es öffnet sich ein Dialog. Hier können Sie einen versionsbezogenen Kommentar hinterlegen. Bestätigen Sie den Dialog mit "OK".  Falls sich mehrere Verwendungsstellen des Bausteins auf unterschiedlichen Steuerungen des Projekts befinden können Sie alle gleichzeitig aktualisieren: "Instanzen im Projekt aktualisieren" ("Update instances in the project"). Falls ältere Versionen des Elements nicht mehr benötigt werden, können Sie diese direkt mit "Nicht verwendete Typ-Versionen aus der Bibliothek löschen" ("Delete unused type versions from library")

3.8 Performancesteigerung bei Prozessalarmen

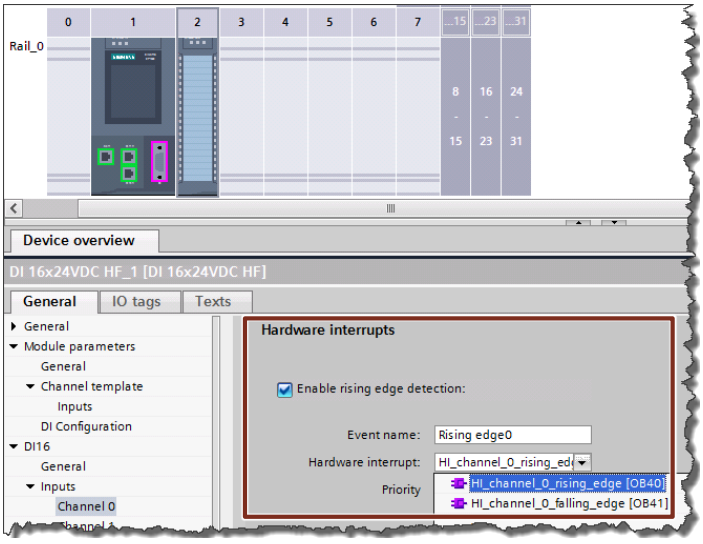
Die Abarbeitung des Anwenderprogramms kann durch Ereignisse wie Prozessalarme beeinflusst werden. Wenn Sie schnelle Reaktionen der Steuerungen auf Hardware-Ereignisse (z.B. eine steigende Flanke eines Kanals einer digitalen Eingangsbaugruppe) benötigen, konfigurieren Sie einen Prozessalarm. Zu jedem Prozessalarm kann ein separater OB programmiert werden. Dieser OB wird vom Betriebssystem der Steuerung im Falle des Prozessalarms aufgerufen. Der Zyklus der Steuerung wird dadurch unterbrochen und nach der Abarbeitung des Prozessalarms wieder weitergeführt.

Abbildung 3-40: Prozessalarm ruft OB auf



In folgender Abbildung sehen Sie die Konfiguration eines "Prozessalarms" ("Hardware Interrupt") in der Hardware-Konfiguration einer digitalen Eingangsbaugruppe.

Abbildung 3-41: Prozessalarm konfigurieren



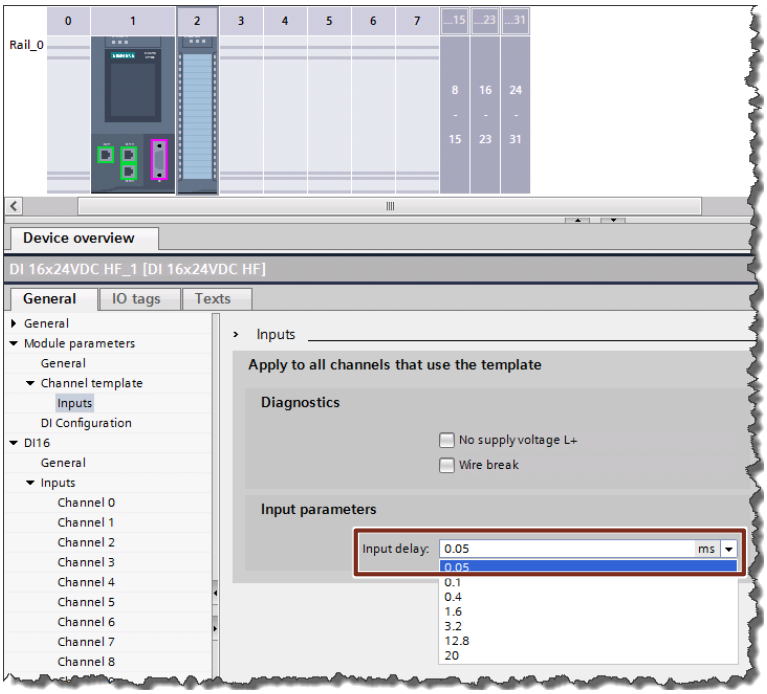
Vorteile

- Schnelle Systemreaktion auf Ereignisse (steigende, fallende Flanke, usw.)
- Jedes Ereignis kann einen separaten OB starten.

Empfehlung

- Nutzen Sie Prozessalarme, um schnelle Reaktionen auf Hardware-Ereignisse zu programmieren.
- Falls die Systemreaktion trotz Programmierung eines Prozessalarms nicht schnell genug ist, können Sie die Reaktion noch beschleunigen. Stellen Sie eine möglichst kleine "Eingangsverzögerung" ("Input delay") in der Baugruppe ein. Eine Reaktion auf ein Ereignis kann immer erst erfolgen, wenn die Eingangsverzögerung abgelaufen ist. Die Eingangsverzögerung wird zur Filterung des Eingangssignals verwendet, um z.B. Störungen wie Kontaktprellen zu kompensieren.

Abbildung 3-42: Eingangsverzögerung einstellen



3.9 Weitere Performance-Empfehlungen

Hier finden Sie noch allgemeine Empfehlungen, die eine schnellere Programmabarbeitung der Steuerung ermöglichen.

Empfehlung

Beachten Sie folgende Empfehlungen bei der Programmierung von S7-1200/1500 Steuerungen, um eine hohe Leistungsfähigkeit zu erreichen:

- KOP/FUP: Deaktivieren Sie "ENO auswerten" bei Bausteinen. Dadurch werden Prüfungen zur Laufzeit vermieden.
- AWL: Verwenden Sie keine Register, da Adress- und Datenregister nur aus Kompatibilitätsgründen von S7-1500 emuliert werden.
- S7-Graph Bausteine brauchen aufgrund ihrer Funktionalität eine längere Abarbeitungszeit in der CPU. Dies ist bedingt durch
  - die zusätzliche implizite Diagnose
  - die integrierte Koordinierung des Ablaufs
  - die realisierten Betriebsarten bzgl. Kettenabarbeitung

  Wenn sie diese unterstützte Funktionalität selbst programmieren, würde dies neben zusätzlichem Programmieraufwand zu ähnlichen Laufzeiten führen.
- Nutzen Sie vorzugsweise optimierte Daten bzw. vermeiden Sie die häufige Konvertierung zwischen optimierten und nicht optimierten Variablen. Häufige Konvertierung, die auch bei Zuweisungen bzw. Parameterübergaben ausgeführt wird, führt zu verlängerten Laufzeiten in der CPU.
- Das Prozessabbild befindet sich grundsätzlich in nichtoptimiertem Speicher. Falls Strukturen (UDT) auf dem Prozessabbild definiert werden, dann sollten diese einmalig mit einer MOVE / Zuweisung auf eine Variable in einem optimierten DB kopiert werden.
- Generell sollten einzelne Bausteine zur „Performancesteigerung“ **nicht** auf Optimierung umgeschaltet werden, sondern immer das gesamte Programm. Eine Mischung von optimierten und nicht optimierten Bausteinen hat, wegen impliziter Konvertierung bei Zuweisungen und Parameterübergaben, in der Regel den gegenteiligen Effekt.
- Reduzieren sie die Verwendung von nicht optimierten Bausteinen auf Situationen, in denen sie funktional notwendig sind.
  Zum Beispiel zur Kommunikation mit Systemen der PLC-Familien AS300 und AS400. Vermeiden sie unnötige wiederholte Konvertierungen, indem sie diese nur anstoßen, wenn neue Daten vorhanden sind.
- Verwendung des Datentyps VARIANT
  - Solange die Datentypen bekannt sind, sollten diese Datentypen konkret und nicht der Datentyp VARIANT verwendet werden. Die Bestimmung des Datentyps von Variablen, die als Datentyp VARIANT übergeben wurden, benötigt Laufzeit. Deshalb sollte der Datentyp VARIANT nur für generische Funktionen genutzt werden, die frei verwendbar auf strukturierte Variablen (UDT) verschiedenen Datentyps reagieren können.
  - Anweisungen, wie "Serialize: Serialisieren", "Deserialize: Deserialisieren", "CMP" (Vergleicher) und "Move_Blk_Variant: Wert kopieren" können sehr große und komplexe strukturierte Variablen verarbeiten. Hierbei analysiert die CPU bei Datentyp VARIANT zur Laufzeit den Aufbau der Variablenstruktur. Dies führt zu längeren Verarbeitungszeiten in der CPU.

Hier sollte geprüft werden, ob ein konkreter Datentyp verwendet werden kann.

- Bei der Übergabe von Arrays mittels Datentyp VARIANT sollte geprüft werden, ob stattdessen das Array mittels Array [*] übergeben werden kann.

Hinweis

Weitere Informationen finden Sie in folgenden Beitrag:

Wie kann bei einer Anweisung der Freigabeausgang ENO deaktiviert werden?
https://support.industry.siemens.com/cs/ww/de/view/67797146

Wie kann die Performance in STEP 7 (TIA Portal) und in den S7-1200/S7-1500 CPUs gesteigert werden?
https://support.industry.siemens.com/cs/ww/de/view/37571372

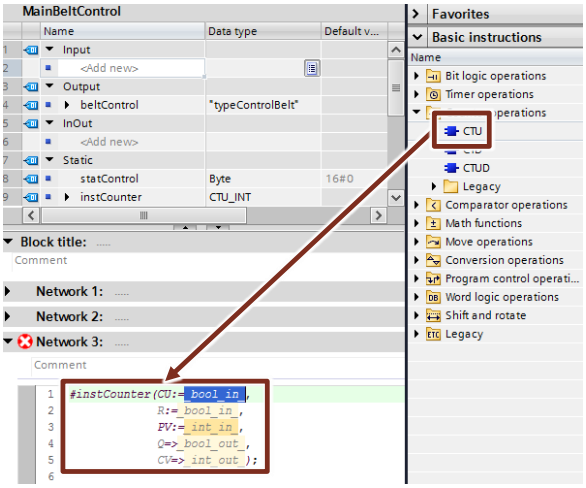
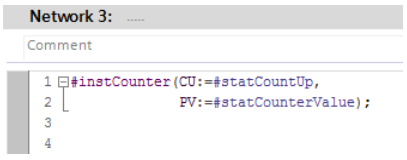
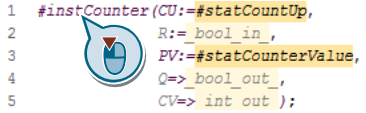
3.10 Programmiersprache SCL: Tipps und Tricks

3.10.1 Nutzung von Aufruftemplates

Viele Anweisungen der Programmiersprachen bieten ein Aufruftemplate mit einer Liste der vorhandenen Formalparameter.

Beispiel

Tabelle 3-12: Einfaches Erweitern des Aufruftemplates

Schritt	Anweisung
1.	Ziehen Sie eine Anweisung aus der Bibliothek in das SCL Programm. Der Editor zeigt das komplette Aufruf-Template an. 
2.	Füllen Sie nun die notwendigen Parameter "CU" und "PV" aus und beenden Sie die Eingabe mit der "Return"-Taste.
3.	Der Editor reduziert automatisch des Aufruf-Template. 
4.	Falls Sie später wieder den kompletten Aufruf bearbeiten wollen, gehen Sie wie folgt vor. Klicken Sie an einer beliebigen Stelle in den Aufruf und drücken Sie anschließend "**CTRL+SHIFT+SPACE**". Jetzt sind Sie im Call Template Modus. Der Editor expandiert den Aufruf wieder. Sie können mit der Taste "TAB" durch die Parameter navigieren. 
5.	Hinweis: Im "Call Template"-Modus ist die Schrift kursiv geschrieben.

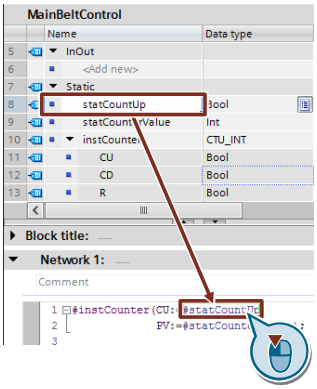
3.10.2 Welche Parameter einer Anweisung sind zwingend notwendig?

Wenn Sie das Aufruf-Template expandieren, zeigt Ihnen die Farbkodierung sofort an, welche Formalparameter einer Anweisung optional sind und Welche nicht. Zwingend notwendige Parameter sind dunkel gekennzeichnet.

3.10.3 Drag & Drop mit ganzen Variablennamen

Im SCL Editor können Sie auch Drag & Drop Funktionen nutzen. Bei Variablennamen werden Sie zusätzlich unterstützt. Wenn Sie eine Variable durch eine andere ersetzen wollen, gehen Sie wie folgt vor.

Tabelle 3-13: Drag & Drop mit Variablen in SCL

Schritt	Anweisung
1.	Ziehen Sie die Variable per Drag & Drop auf die Variable im Programm, die ersetzt werden soll. Halten Sie die Variable länger als 1 Sekunde bevor Sie sie loslassen.  > 1 Sekunde gedrückt halten Die komplette Variable wird ersetzt.

3.10.4 Strukturierung mit dem Schlüsselwort REGION (ab V14)

Mit dem Schlüsselwort REGION kann der SCL Code in Bereiche unterteilt werden. Diese Bereiche können mit einem Namen gekennzeichnet und auf- und zugeklappt werden.

Vorteile

- Bessere Übersichtlichkeit
- Einfache Navigation auch in großen Bausteinen
- Fertige Codefragmente können ausgeblendet werden.

Eigenschaften

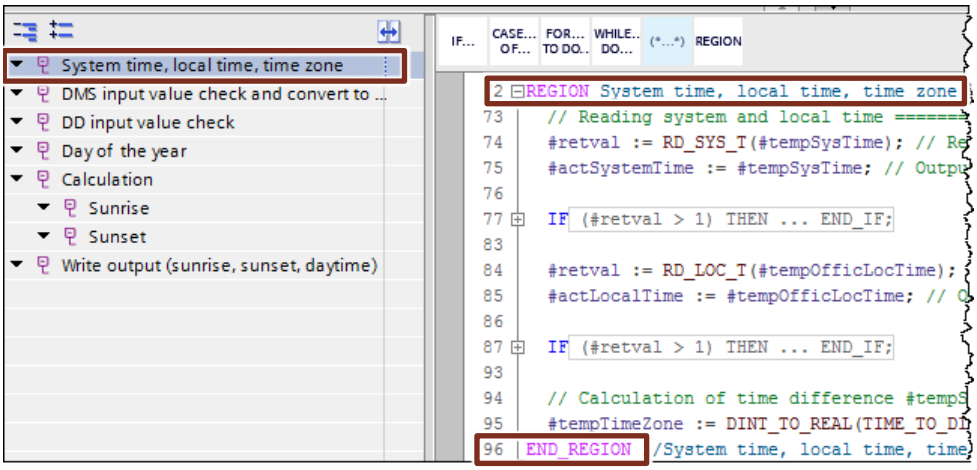
REGIONs können verschachtelt werden.

Empfehlung

Verwenden Sie das Schlüsselwort REGION zur Strukturierung ihrer SCL-Bausteine.

Beispiel

Abbildung 3-43: SCL-Editor



3.10.5 Richtige Verwendung von FOR, REPEAT und WHILE-Schleifen

Bei der Verwendung von Schleifen gibt es unterschiedliche Ausführungen und Anwendungsfälle. Folgende Beispiele zeigen die Unterschiede.

Eigenschaften: FOR-Schleife

Die FOR-Schleife durchläuft eine **definierte Anzahl von Durchläufen**. Der Laufvariablen wird beim Beginn ein Startwert zugewiesen. Anschließend wird sie in jedem Schleifendurchlauf mit der angegebenen Schrittweite bis zu dem Endwert hochgezählt.

Aus Performancegründen werden sowohl Start- als auch Endwert einmal zu Beginn berechnet. Die Laufvariable kann demzufolge im Schleifencode nicht mehr beeinflusst werden.

Syntax

```
FOR statCounter := statStartCount TO statEndCount DO
    // Statement section ;
END_FOR;
```

Mit dem Befehl EXIT kann die Schleife jederzeit abgebrochen werden.

Eigenschaften: WHILE-Schleife

Die WHILE-Schleife wird durch eine Abbruchbedingung beendet. Die **Abbruchbedingung wird vor Beginn** des Schleifencodes geprüft. D.h. die Schleife wird, falls die Bedingung sofort erfüllt ist, nicht ausgeführt. Im Schleifencode kann jede Variable für den nächsten Durchlauf angepasst werden.

Syntax

```
WHILE condition DO
     // Statement section ;
END_WHILE;
```

Eigenschaften: REPEAT-Schleife

Die REPEAT-Schleife wird durch eine Abbruchbedingung beendet. Die **Abbruchbedingung wird am Ende** des Schleifencodes geprüft. D.h. die Schleife wird **mindestens einmal durchlaufen**. Im Schleifencode kann jede Variable für den nächsten Durchlauf angepasst werden.

Syntax

```
REPEAT
    // Statement section ;
UNTIL condition
END_REPEAT;
```

Empfehlung

- Verwenden Sie FOR-Schleifen, wenn die Laufvariable klar definiert wird.
- Verwenden Sie die WHILE oder der REPEAT -Schleife, wenn eine Laufvariable während der Schleifenbearbeitung angepasst werden muss.

3.10.6 CASE-Anweisung effizient einsetzen

Mit der CASE-Anweisung wird in SCL genau der mit der Bedingung angewählte CASE Block angesprungen. Nach Ausführung des CASE Blockes wird die Anweisung beendet. Das erlaubt Ihnen z.B. oft benötigte Wertebereiche gezielter und einfacher zu überprüfen.

Beispiel

```
CASE #myVar OF
        5:
                #Engine(#myParam);
        10,12:
                #Transport(#myParam);
        15:
                #Lift(#myParam);
        0..20:
                #Global(#myParam);
// Global wird niemals für die Werte 5, 10, 12 oder 15 aufgerufen!
        ELSE
END_CASE;
```

Hinweis CASE Anweisungen funktionieren auch mit CHAR, STRING Datentypen, sowie mit Elementen (siehe Beispiel im Kapitel 2.8.5 Datentyp VARIANT).

3.10.7 Keine Manipulation von Schleifenzähler bei FOR-Schleife

FOR-Schleifen in SCL sind reine Zählschleifen, d.h. die Anzahl die Iterationen steht beim Eintritt in die Schleife fest. In einer FOR-Schleife können Sie den Schleifenzähler nicht ändern.

Mit der Anweisung EXIT kann eine Schleife jederzeit abgebrochen werden.

Vorteile

- Der Compiler kann das Programm besser optimieren, da er die Anzahl der Iterationen kennt.

Beispiel

```
FOR #statVar := #statLower TO #statUpper DO
  #statVar := #statVar + 1; // kein Effekt, Compiler warning
END_FOR;
```

3.10.8 FOR-Schleifen Rückwärts

Sie können in SCL den Index von FOR-Schleifen auch rückwärts oder in anderen Schrittweiten hochzählen. Nutzen Sie dafür im Schleifenkopf das optionale Schlüsselwort "BY".

Beispiel

```
FOR #statVar := #statUpper TO #statLower BY -2 DO


END_FOR;
```

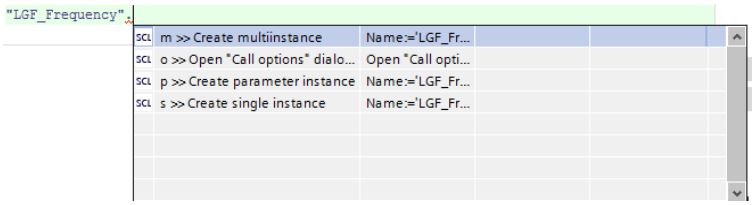
Wenn Sie wie im Beispiel "BY" als "-2" definieren, wird der Zähler in jeder Iteration um 2 erniedrigt. Wenn Sie "BY" weglassen, ist die Standardeinstellung für "BY" 1

3.10.9 Einfaches Erzeugen von Instanzen bei Aufrufen

Wenn Sie bevorzugt mit der Tastatur arbeiten, gibt es in SCL eine einfache Möglichkeit Instanzen für Bausteinaufrufe zu erzeugen.

Beispiel

Tabelle 3-14: Einfaches Erstellen von Instanzen

Schritt	Anweisung
1.	Geben Sie den Bausteinnamen ein gefolgt von einem "." (Punkt). Die Autovervollständigung zeigt Ihnen nun folgendes an. 
2.	Oben sehen Sie die bereits vorhandenen Instanzen. Zusätzlich können Sie direkt eine neue Singleinstanz oder Multiinstanz erzeugen. Verwenden Sie die Tastaturkürzel "s" bzw. "m" um im Autovervollständigungsfenster direkt zu den jeweiligen Einträgen zu springen.

3.10.10 Handhabung von Zeit-Variablen

In SCL können die Zeit-Variablen genauso rechnen, wie mit normalen Zahlen. D.h. Sie brauchen keine zusätzlichen Funktionen, wie z.B. T_COMBINE zu suchen, sondern nutzen einfache Arithmetik. Diese Vorgehensweise nennt man "Überladen von Operanden". Der SCL Compiler nutzt automatisch die passenden Funktionen. Sie können bei Zeittypen eine sinnvolle Arithmetik anwenden und so effizienter programmieren.

Beispiel

```
zeitdifferenz := zeitstempel1 - zeitstempel2;
```

Die folgende Tabelle fasst die überladenen Operatoren und die dahinter stehenden Operationen zusammen:

Tabelle 3-15: Überladene Operanden bei SCL

überladener Operand	Operation
ltime + time	T_ADD LTime
ltime – time	T_SUB LTime
ltime + lint	T_ADD LTime
ltime – lint	T_SUB LTime
time + time	T_ADD Time
time - time	T_SUB Time
time + dint	T_ADD Time
time - dint	T_SUB Time
ldt + ltime	T_ADD LDT / LTime
ldt – ltime	T_SUB LDT / LTime
ldt + time	T_ADD LDT / Time
ldt – time	T_SUB LDT / Time
dtl + ltime	T_ADD DTL / LTime
dtl – ltime	T_SUB DTL / LTime
dtl + time	T_ADD DTL / Time
dtl – time	T_SUB DTL / Time
ltod + ltime	T_ADD LTOD / LTime
ltod – ltime	T_SUB LTOD / LTime
ltod + lint	T_ADD LTOD / LTime
ltod – lint	T_SUB LTOD / LTime
ltod + time	T_ADD LTOD / Time
ltod – time	T_SUB LTOD / Time
tod + time	T_ADD TOD / Time
tod – time	T_SUB TOD / Time
tod + dint	T_ADD TOD / Time
tod – dint	T_SUB TOD / Time
dt + time	T_ADD DT / Time
dt – time	T_SUB DT / Time
ldt – ldt	T_DIFF LDT
dtl – dtl	T_DIFF DTL
dt – dt	T_DIFF DT
date – date	T_DIFF DATE
ltod – ltod	T_DIFF LTOD
date + ltod	T_COMBINE DATE / LTOD
date + tod	T_COMBINE DATE / TOD

3.10.11 Unnötige IF-Anweisungen

Programmierer denken oft in IF-THEN-ELSE-Anweisungen. Dadurch ergeben sich in den Programmen oft unnötige Konstrukte.

Beispiel

```
IF (statOn1 = TRUE AND statOn2 = TRUE) THEN
  statMotor := TRUE;
ELSE
  statMotor := FALSE;
END_IF
```

Empfehlung

Denken Sie bei boolschen Abfragen daran, dass oft eine direkte Zuweisung effektiver ist. Das komplette Konstrukt kann mit einer Zeile programmiert werden.

Beispiel

```
statMotor := statOn1 AND statOn2;
```

4 Hardwareunabhängige Programmierung

Um sicher zu stellen, dass ein Baustein problemlos auf allen Steuerungen ohne weitere Anpassungen eingesetzt werden kann, ist es wichtig, keine handwareabhängigen Funktionen und Eigenschaften zu nutzen.

4.1 Datentypen von S7-300/400 und S7-1200/1500

Im Folgenden sind alle elementaren Datentypen und Datengruppen aufgelistet.

Empfehlung

- Nutzen Sie nur die Datentypen, die von den Steuerungen unterstützt, werden auf denen das Programm laufen soll.

Tabelle 4-1: Elementare Datentypen entsprechend der Norm EN 61131-3

	Beschreibung	S7-300/400	S7-1200	S7-1500
Bit-Datentypen	• BOOL • BYTE • WORD • DWORD	ja	ja	ja
	• LWORD	nein	nein	ja
Zeichentyp	• CHAR (8 Bit)	ja	ja	ja
Numerische Datentypen	• INT (16 Bit) • DINT (32 Bit) • REAL (32 Bit)	ja	ja	ja
	• SINT (8 Bit) • USINT (8 Bit) • UINT (16 Bit) • UDINT (32 Bit) • LREAL (64 Bit)	nein	ja	ja
	• LINT (64 Bit) • ULINT (64 Bit)	nein	nein	ja
Zeittypen	• TIME • DATE • TIME_OF_DAY	ja	ja	ja
	• S5TIME	ja	nein	ja
	• LTIME • L_TIME_OF_DAY	nein	nein	ja

Tabelle 4-2: Datengruppen, die sich aus anderen Datentypen zusammensetzen

	Beschreibung	S7-300/400	S7-1200	S7-1500
Zeittypen	• DT (DATE_AND_TIME)	ja	nein	ja
	• DTL	nein	ja	ja
	• LDT (L_DATE_AND_TIME)	nein	nein	ja
Zeichentyp	• STRING	ja	ja	ja
Feld	• ARRAY	ja	ja	ja
Struktur	• STRUCT	ja	ja	ja

Tabelle 4-3: Parametertypen für Formalparameter, die zwischen Bausteinen übergeben werden

	Beschreibung	S7-300/400	S7-1200	S7-1500
Zeiger	• POINTER • ANY	ja	nein	ja [1)]
	• VARIANT	nein	ja	ja
Bausteine	• TIMER • COUNTER	ja	ja [2)]	ja
	• BLOCK_FB • BLOCK_FC	ja	nein	ja
	• BLOCK_DB • BLOCK_SDB	ja	nein	nein
	• VOID	ja	ja	ja
PLC-Datentypen	• PLC-DATENTYP	ja	ja	ja

1) Bei optimierten Zugriffen ist nur symbolische Adressierung möglich

2) Bei S7-1200/1500 wird der Datentyp TIMER und COUNTER durch IEC_TIMER und IEC_Counter repräsentiert.

4.2 Keine Merker, sondern globale Datenbausteine

Vorteile

- Optimierte Global-DBs sind deutlich leistungsfähiger als der Merkerbereich, der nur aus Kompatibilitätsgründen nicht optimiert ist.

Empfehlung

- Der Umgang mit Merkern (auch System- und Taktmerker) ist problematisch, da jede Steuerung einen unterschiedlich großen Merkerbereich hat. Nutzen Sie für die Programmierung keine Merker, sondern immer globale Datenbausteine. Somit bleibt das Programm universell einsetzbar.

4.3 Programmieren von "Takt Bits"

Empfehlung

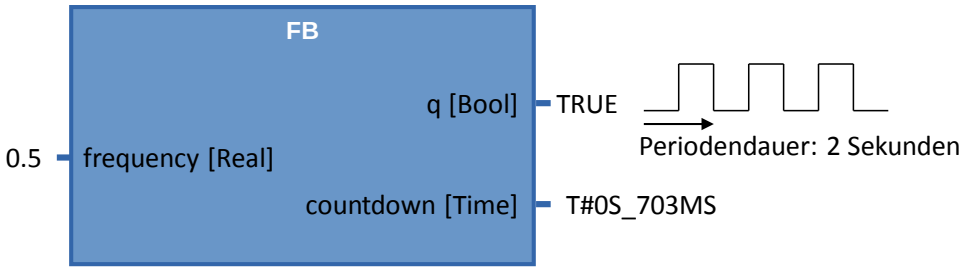
Bei der Programmierung von Taktmerkern, muss immer eine korrekte Einstellung in der Hardware-Konfiguration vorliegen.

Verwenden Sie einen programmierten Baustein als Taktgeber. Im Folgenden finden Sie ein Programmierbeispiel für einen Taktgeber in der Programmiersprache SCL.

Beispiel

Der programmierte Baustein hat folgende Funktionalität. Es wird eine gewünschte Frequenz ("frequency") vorgegeben. Der Ausgang "q" ist ein boolscher Wert, der in der gewünschten Frequenz toggelt. Der Ausgang "countdown" gibt die verbleibende Zeit des aktuellen Zustands von "q" aus.

Falls die gewünschte Frequenz kleiner oder gleich 0.0 ist, ist der Ausgang q = FALSE und Countdown = 0.0.



Hinweis Das komplette Programmbeispiel finden Sie zum Download unter folgendem Beitrag:

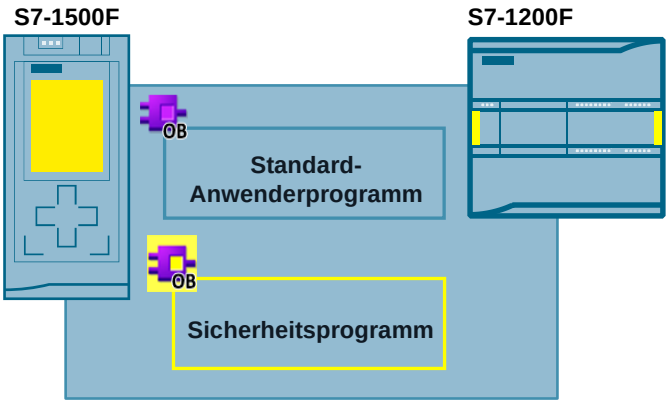
https://support.industry.siemens.com/cs/ww/de/view/109479728

5 STEP 7 Safety im TIA Portal

5.1 Einleitung

Ab TIA Portal V13 SP1 werden fehlersichere S7-1200F und S7-1500F CPUs unterstützt. In diesen Steuerungen ist sowohl Standard- als auch fehlersichere Programmierung in einem Gerät möglich. Für die Programmierung des fehlersicheren Anwenderprogramms wird das Optionspaket SIMATIC STEP 7 Safety (TIA Portal) eingesetzt.

Abbildung 5-1: Standard- und Sicherheitsprogramm



Vorteile

- Einheitliche Programmierung in Standard- und fehlersicheren Programm mit einem Engineering Tool: TIA Portal
- Gewohnte Programmierung in KOP (LAD) und FUP (FBD)
- Einheitliche Diagnose- und Onlinefunktionen

Hinweis Fehlersicher bedeutet nicht, dass das Programm frei von Fehlern ist. Der Programmierer ist für die Richtigkeit der Programmierlogik verantwortlich.

Fehlersicher bedeutet, dass die korrekte Abarbeitung des fehlersicheren Anwenderprogramms in der Steuerung sichergestellt ist.

Hinweis Weitere Informationen zum Thema Safety wie z.B. Sicherheitsanforderungen oder die Prinzipien von Sicherheitsprogrammen Finden Sie unter:

TIA Portal - Ein Überblick der wichtigsten Dokumente und Links - Safety
https://support.industry.siemens.com/cs/ww/de/view/90939626

Applikationen & Tools – Safety Integrated
https://support.industry.siemens.com/cs/ww/de/ps/14675/ae

STEP 7 Safety (TIA Portal) - Handbücher
https://support.industry.siemens.com/cs/ww/de/ps/14675/man

5.2 Begriffe

In diesem Dokument werden durchgehend die Begriffe mit folgender Bedeutung verwendet.

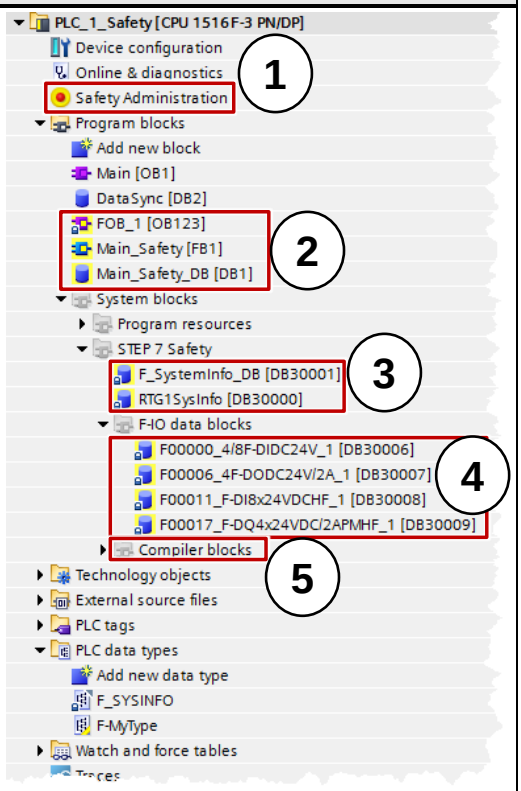
Tabelle 5-1:Safety Begriffe

Begriff	Erklärung
Standard-Anwenderprogramm	Das Standard-Anwenderprogramm ist der Programmteil, der nichts mit der F-Programmierung zu tun hat.
Sicherheitsprogramm (F-Programm, fehlersicheres Anwenderprogramm)	Das fehlersichere Anwenderprogramm ist der Programmteil, der eigenständig von der Steuerung sicherheitsgerichtet abgearbeitet wird. Alle fehlersicheren Bausteine und Anweisungen werden zur Unterscheidung von Bausteinen und Anweisungen des Standard-Anwenderprogramms an der Software-Oberfläche (z. B. in der Projektnavigation) gelb hinterlegt. Ebenso werden die fehlersicheren Parameter von F-CPUs und F-Peripherie in der Hardware-Konfiguration gelb hinterlegt.

5.3 Bestandteile des Sicherheitsprogramms

Das Sicherheitsprogramm besteht immer aus anwendererstellten, systemgenerierten F-Bausteinen und dem "Safety Administration"-Editor.

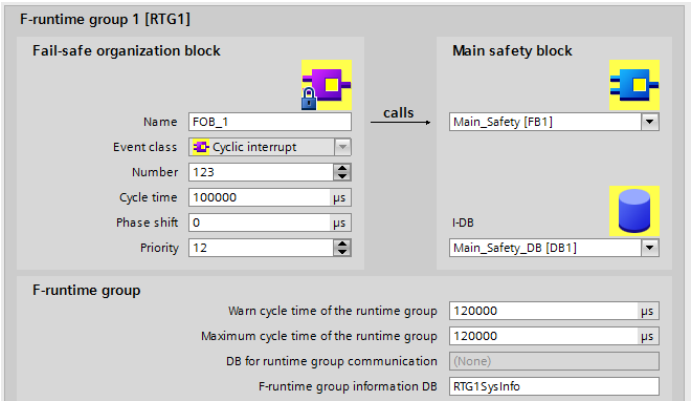
Tabelle 5-2: Bestandteile Sicherheitsprogramm

Erklärung	Bild
1. "Safety Administration"-Editor - Status des Sicherheitsprogramms - F-Gesamtsignatur - Status des Sicherheitsbetriebs - F-Ablaufgruppen anlegen/organisieren - Informationen zu den F-Bausteinen - Informationen zu F-konformen PLC-Datentypen - Zugriffsschutz festlegen/ändern 2. Anwendererstellte F-Bausteine 3. Systemgenerierte F-Ablaufgruppenbausteine - Bausteine enthalten Statusinformationen über die F-Ablaufgruppe. 4. Systemgenerierte F-Peripheriedatenbausteine - Bausteine enthalten Variablen zur Auswertung der F-Baugruppen. 5. "Compiler blocks" Systemgenerierte Überprüfungsbausteine - Diese laufen im Hintergrund der Steuerung und sorgen für die sicherheitsgerichtete Abarbeitung des Sicherheitsprogramms. - Diese Bausteine können nicht vom Anwender verarbeitet werden.	

5.4 F-Ablaufgruppe

Ein Sicherheitsprogramm wird immer in einer F-Ablaufgruppe mit definiertem Zyklus abgearbeitet. Eine F-Ablaufgruppe besteht aus einem "Fail-safe organization block" der einen "Main safety block" aufruft. Alle anwendererstellten Sicherheitsfunktionen werden vom "Main safety block" aufgerufen.

Abbildung 5-2: F-Ablaufgruppe im "Safety Administrator"-Editor



Vorteile

- Ablaufgruppen können einfach im "Safety Administrator" erstellt und konfiguriert werden.
- F-Bausteine der Ablaufgruppe werden automatisch erstellt.

Eigenschaften

- Es können maximal zwei F-Ablaufgruppen erstellt werden.

5.5 F-Signatur

Jede F-Komponente (Station, Peripherie, Bausteine) hat eine eindeutige F-Signatur. Mit Hilfe der F-Signatur kann schnell festgestellt werden, ob eine F-Gerätekonfiguration, F-Bausteine oder eine komplette Station noch der ursprünglichen Parametrierung oder Programmierung entspricht.

Vorteile

- Einfache und schnelle Vergleichsmöglichkeit von F-Bausteinen und F-Gerätekonfigurationen

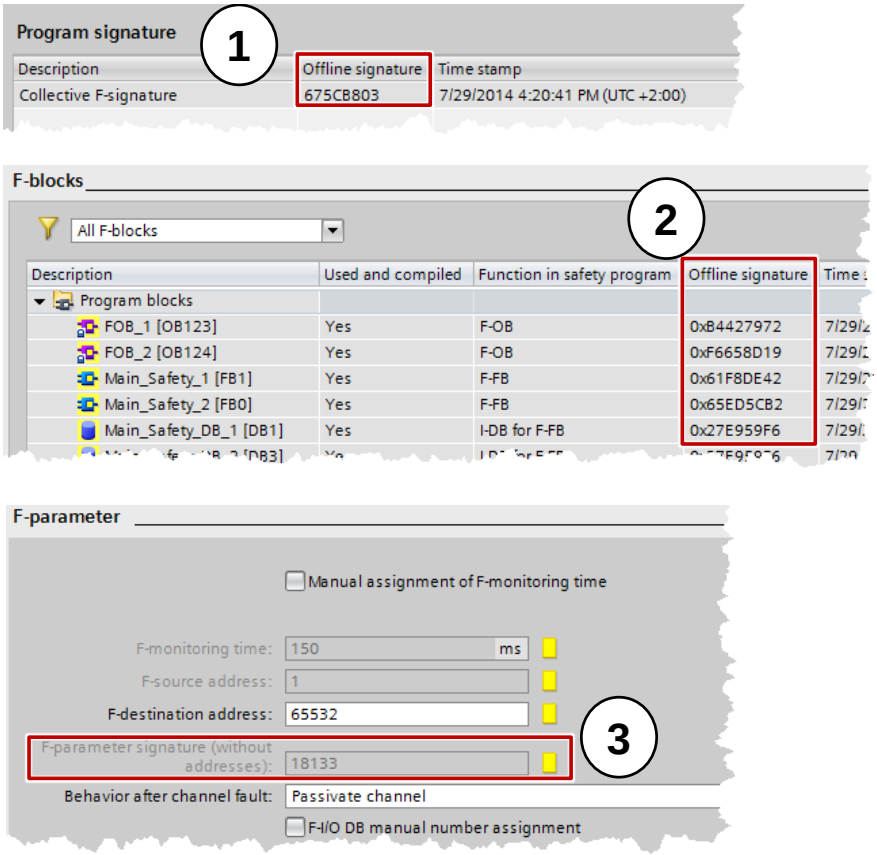
Eigenschaften

- F-Parameter Signatur (ohne Adresse von F-Peripherie)…
  - wird nur durch Anpassen der Parameter geändert.
  - bleibt unverändert beim Ändern der PROFIsafe-Adresse. Es ändert sich aber die F-Gesamtsignatur der Station.
- F-Baustein Signatur wird nur geändert, wenn sich die Logik im F-Baustein ändert.

- F-Baustein Signatur bleibt unverändert durch Ändern der
  - Bausteinnummer,
  - Bausteinschnittstelle,
  - Baustein-Version.

Beispiel

Abbildung 5-3: Beispiele für F-Signaturen



1. F-Gesamtsignatur der Station im "Safety Administrator"-Editor
2. F-Bausteinsignaturen im "Safety Administrator"-Editor (kann auch in den Eigenschaften des Bausteins ausgelesen werden)
3. F-Parametersignatur in der "Gerätesicht" ("Device view") unter "Geräte & Netze" ("Devices & network")

Hinweis Bei S7-1500F Steuerungen ist es möglich, die F-Gesamtsignatur direkt auf dem eingebauten Display oder im integrierten Webserver abzulesen.

5.6 Vergabe der PROFIsafe-Adresse bei F-Peripherie

Jedes F-Peripherie-Gerät hat eine PROFIsafe-Adresse zur Identifikation und Kommunikation mit F-Steuerungen. Bei der Vergabe der PROFIsafe-Adresse gibt es zwei unterschiedliche Parametrierungen.

Tabelle 5-3: Einstellen der F-Adresse

ET 200M / ET 200S (PROFIsafe-Adresstyp 1)	ET 200MP / ET 200SP (PROFIsafe-Adresstyp 2)
Vergabe der PROFIsafe-Adresse über DIL-Schalter direkt an den Baugruppen Die PROFIsafe-Adresse muss in der Gerätekonfiguration im TIA Portal und der DIL-Schalterstellung an der Peripherie übereinstimmen.	Vergabe der PROFIsafe-Adresse ausschließlich über TIA Portal Mit dem TIA Portal wird die konfigurierte PROFIsafe-Adresse auf das intelligente Kodiermodul der Baugruppe geladen.

Vorteile

- Austausch eines F-Moduls ist möglich ohne erneute Vergabe der PROFIsafe-Adresse bei ET 200MP und ET 200SP. Das intelligente Kodiermodul bleibt in der BaseUnit beim Modultausch.
- Einfache Parametrierung, da TIA Portal auf fehlerhafte Vergabe der PROFIsafe-Adresse Warnungen anzeigt.
- Die PROFIsafe-Adressen aller F-Module innerhalb einer ET 200SP Station können auf einmal zugewiesen werden.

Hinweis Weitere Informationen zur Vergabe der PROFIsafe-Adresse bei F-Peripherie finden Sie unter:

SIMATIC Industrie Software SIMATIC Safety - Projektieren und Programmieren
https://support.industry.siemens.com/cs/ww/de/view/54110126

5.7 Auswertung von der F-Peripherie

In den F-Peripherie-Bausteinen sind alle aktuellen Zustände der jeweiligen F-Peripherie gespeichert. Im Sicherheitsprogramm können die Zustände ausgewertet und verarbeitet werden. Hierbei gibt es folgende Unterschiede zwischen S7-1200F/1500F und S7-300F/400F.

Tabelle 5-4: Variablen im F-Peripherie-DB mit S7-300F/400F und S7-1500F

Variable im F-Peripherie-DB bzw. Wertstatus im PAE	F-Peripherie mit S7-300F/400F	F-Peripherie mit S7-1200F/1500F
ACK_NEC	ja	ja
QBAD	ja	ja
PASS_OUT	ja	ja
QBAD_I_xx *	ja	nein
QBAD_O_xx *	ja	nein
Wertstatus	nein	ja

* QBAD_I_xx und QBAD_O_xx zeigen die Gültigkeit des Kanalwertes an und entsprechen damit dem **invertierten** Wertstatus bei S7-1200/1500 (weitere Informationen finden Sie in folgendem Kapitel).

5.8 Wertstatus (S7-1200/1500)

Zusätzlich zu den Diagnosemeldungen und der Status- und Fehleranzeige stellt das F-Modul für jedes Ein- und Ausgangssignal eine Information über dessen Gültigkeit zur Verfügung - den Wertstatus. Der Wertstatus wird wie das Eingangssignal im Prozessabbild abgelegt.

Der Wertstatus gibt Auskunft über die Gültigkeit des dazugehörigen Kanalwertes:

- 1: Für den Kanal wird ein gültiger Prozesswert ausgegeben.
- 0: Für den Kanal wird ein Ersatzwert ausgegeben

Tabelle 5-5: Unterschiede Q_BAD (S7-300F/400F) und Wertstatus (S7-1200F/1500F)

Szenario	QBAD (S7-300F/400F)	Wertstatus (S7-1200F/1500F)
Gültige Werte an F-Peripherie (kein Fehler)	FALSE	TRUE
Kanalfehler tritt auf	TRUE	FALSE
Kanalfehler ist gegangen (ACK_REQ)	TRUE	FALSE
Quittierung des Fehlers (ACK_REI)	FALSE	TRUE

Eigenschaften

- Der Wertstatus wird in das Prozessabbild der Eingänge und Ausgänge eingetragen.
- Auf den Kanalwert und Wertstatus einer F-Peripherie darf nur aus derselben F-Ablaufgruppe zugegriffen werden.

Empfehlung

- Vergeben Sie zu besseren Lesbarkeit als symbolischen Namen für den Wertstatus die Endung "`VS`", z.B. "`TagIn1VS`".

Beispiel

Lage der Wertstatus-Bits im Prozessabbild am Beispiel einer F-DI 8x24VDC HF Baugruppe.

Tabelle 5-6: Wertstatus-Bits im Prozessabbild am Beispiel einer F-DI 8x24VDC HF

Byte in der F-CPU	Belegte Bits in der F-CPU							
	7	6	5	4	3	2	1	0
x + 0	DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0
x +1	Wert-status für DI_7	Wert-status für DI_6	Wert-status für DI_5	Wert-status für DI_4	Wert-status für DI_3	Wert-status für DI_2	Wert-status für DI_1	Wert-status für DI_0

x = Modulanfangsadresse

Hinweis Weitere Informationen über den Wertstatus aller ET 200SP Baugruppen finden Sie unter:

Fehlersichere CPUs - Handbücher
https://support.industry.siemens.com/cs/ww/de/ps/13719/man

Fehlersichere Peripheriemodule - Handbücher
https://support.industry.siemens.com/cs/ww/de/ps/14059/man

5.9 Datentypen

5.9.1 Überblick

Für Sicherheitsprogramme der S7-1200/1500F gibt es einen eingeschränkten Umfang an Datentypen.

Tabelle 5-7: Ganzzahlige Datentypen

Typ	Größe	Wertebereich
BOOL	1 Bit	0 .. 1
INT	16 Bit	-32.768 .. 32.767
WORD	16 Bit	-32.768 .. 65.535
DINT	32 Bit	-2,14 .. 2,14 Mio
TIME	32 Bit	T#-24d20h31m23s648ms to T#+24d20h31m23s647ms

5.9.2 Implizite Konvertierung

In sicherheitsgerichteten Anwendungen kann es notwendig sein, mathematische Funktionen mit Variablen unterschiedlicher Datentypen durchzuführen. Die dafür notwendigen Funktionsblöcke verlangen dafür ein definiertes Datenformat der Formalparameter. Sollte der Operand nicht dem erwarteten Datentyp entsprechen, muß zuerst eine Konvertierung durchgeführt werden.

Unter folgenden Voraussetzungen kann die S7-1200/1500 die Datenkonvertierung auch implizit durchführen:

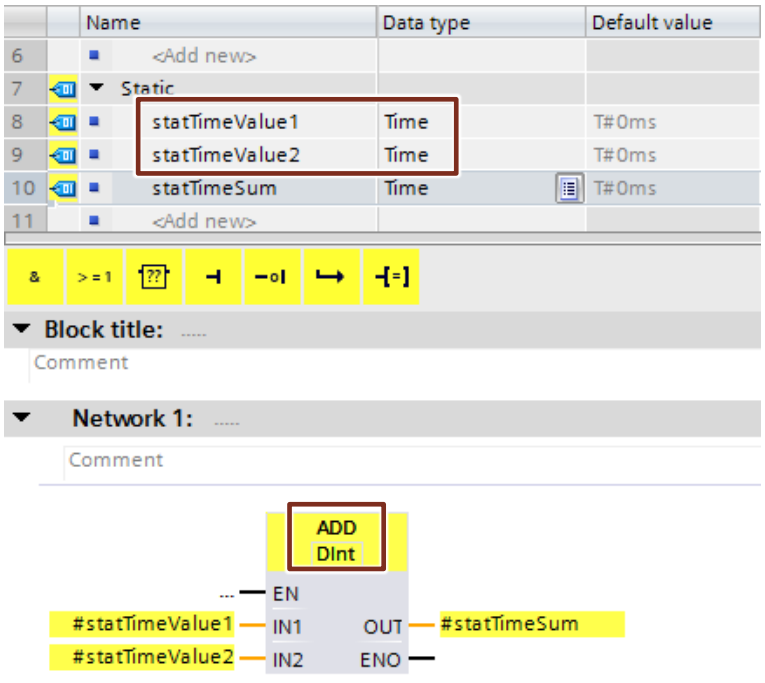
- IEC-Prüfung ist deaktiviert.
- Die Datentypen weisen die gleiche Länge auf.

Im Sicherheitsprogramm können daher folgende Datentypen implizit konvertiert werden:

- WORD ↔ INT
- DINT ↔ TIME

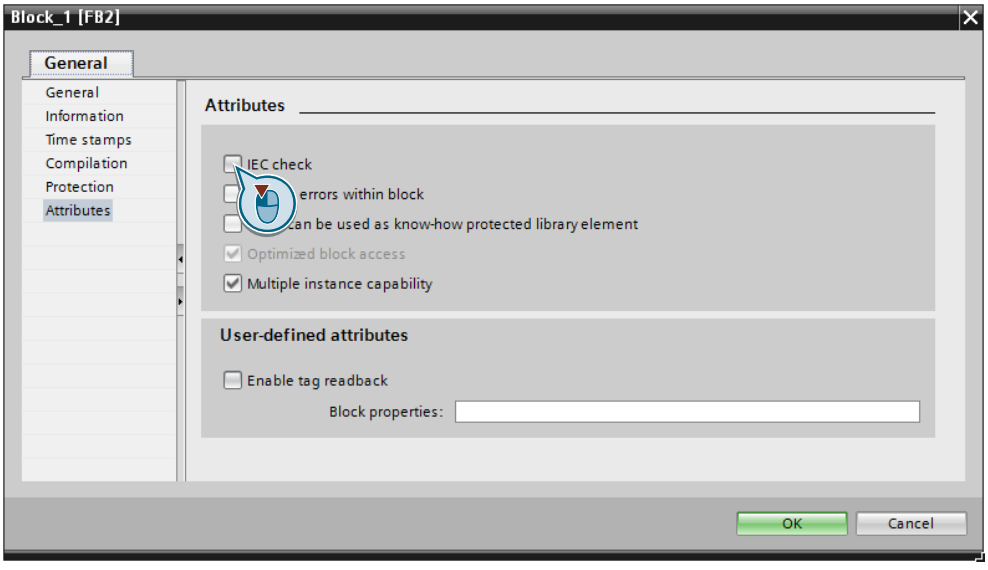
Eine praktische Anwendung ist die Addition von zwei Zeitwerten, obwohl für die Funktion "Add" als Eingang "DInt" benötigt wird. Das Ergebnis wird dann ebenso als "Time" Variable ausgegeben.

Abbildung 5-4: Addition von zwei Zeitwerten



Die IEC-Prüfung aktivieren bzw. deaktivieren Sie in den Eigenschaften des jeweiligen Funktionsbausteins bzw. Funktion.

Abbildung 5-5: IEC-Prüfung deaktivieren



5.10 F-konformer PLC-Datentyp

Auch bei Sicherheitsprogrammen ist es möglich, Daten optimal mit PLC-Datentypen zu strukturieren.

Vorteile

- Eine Änderung in einem PLC-Datentyp wird an allen Verwendungsstellen im Anwenderprogramm automatisch aktualisiert.

Eigenschaften

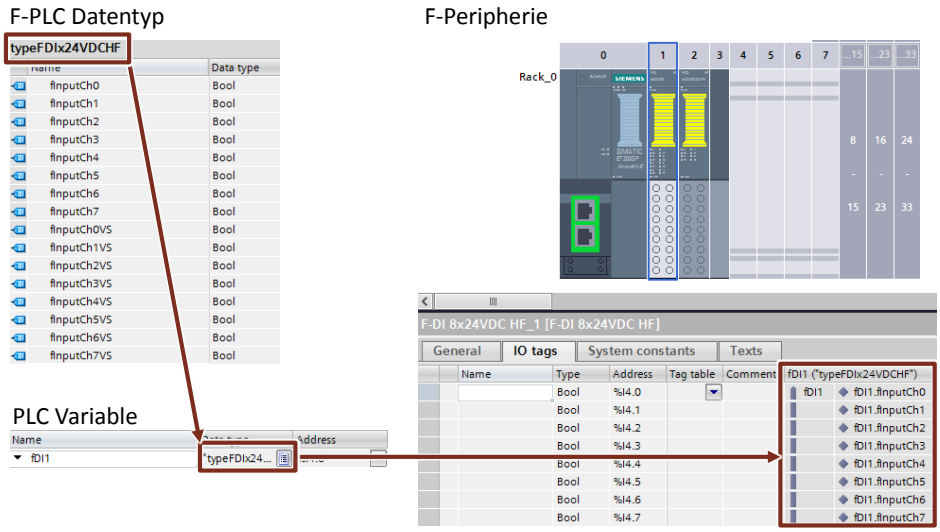
- F-PLC-Datentypen werden genauso wie PLC-Datentypen deklariert und verwendet.
- Als F-PLC-Datentypen können alle Datentypen verwendet werden, die im Sicherheitsprogramm erlaubt sind.
- Die Verschachtelung von F-PLC-Datentypen innerhalb anderer F-PLC-Datentypen wird nicht unterstützt.
- F-PLC-Datentypen können sowohl im Sicherheitsprogramm als auch im Standard-Anwenderprogramm eingesetzt werden.

Empfehlung

- Nutzen Sie für den Zugriff auf E/A-Bereiche F-PLC-Datentypen (wie in Kapitel 3.6.5 Zugriff mit PLC-Datentypen auf E/A-Bereiche)
- Folgende Regeln müssen dabei beachtet werden:
  - Die Struktur der Variablen des F-konformen PLC-Datentyps muss mit der Kanalstruktur der F-Peripherie übereinstimmen.
  - Ein F-konformer PLC-Datentyp für eine F-Peripherie mit 8 Kanälen ist z.B.:
    - 8 BOOL Variablen (Kanalwert) oder
    - 16 BOOL Variablen (Kanalwert + Wertstatus)
  - Zugriffe auf F-Peripherie sind nur für aktivierte Kanäle erlaubt. Bei der Parametrierung einer 1oo2 (2v2)-Auswertung wird immer der höherwertige Kanal deaktiviert.

Beispiel

Abbildung 5-6: Zugriff auf E/A-Bereiche mit F-PLC-Datentypen



5.11 TRUE / FALSE

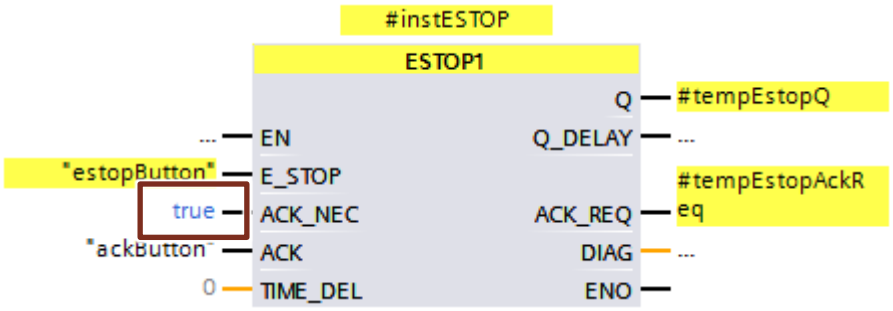
Die Verwendung von "TRUE" und "FALSE"-Signale in Sicherheitsprogrammen, kann in zwei Anwendungsfälle unterschieden werden:

- als Aktualparameter an Bausteinen
- als Zuweisungen an Operationen

Aktualparameter an Bausteinen

Bei S7-1200F/1500F Steuerungen können Sie zur Versorgung von Formalparametern bei Bausteinaufrufen im Sicherheitsprogramm die boolschen Konstanten "FALSE" für 0 und "TRUE" für 1 als Aktualparameter verwenden. Es wird an den Formalparameter nur das Stichwort "FALSE" oder "TRUE" geschrieben.

Abbildung 5-7: "TRUE" bzw. "FALSE"-Signale als Aktualparameter



Zuweisungen an Operationen

Um "TRUE" oder "FALSE"-Signale für Operationen zu erzeugen, gehen Sie diese wie folgt vor:

1. Erstellen Sie zwei statische Variablen "statTrue" und "statFalse" vom Typ BOOL.
2. Geben Sie der Variable statFalse den Defaultwert "false".
3. Geben Sie der Variable statTrue den Defaultwert "true".

Die Variablen können Sie im kompletten Funktionsbaustein als "True" und "False"-Signale lesend einsetzen.

Abbildung 5-8: "TRUE" und "FALSE"-Signale

	Name	Data type	Default value	Retain
	Static			
	statTrue	Bool	true	Non-retain
	statFalse	Bool	false	Non-retain

5.12 Optimierung der Übersetzungs- und Programmlaufzeit

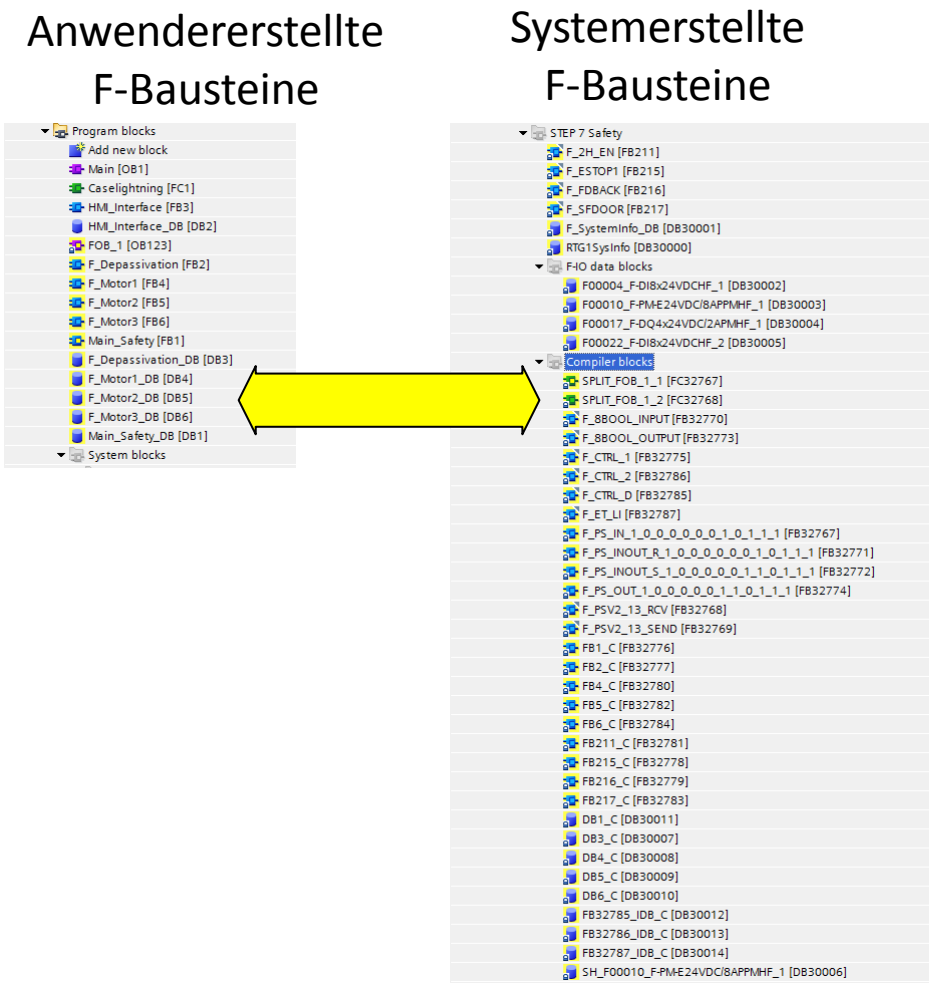
Ein wichtiger Bestandteil eines Sicherheitsprogramms ist die Absicherung der Anwenderprogrammierung durch das Coded Processing. Ziel ist es, jegliche Datenverfälschung im Sicherheitsprogramm aufzudecken und damit unsichere Zustände zu verhindern.

Dieses Absicherungsprogramm wird während der Übersetzung erzeugt und verlängert so die Übersetzungsdauer. Auch die Laufzeit der F-CPU wird durch das Absicherungsprogramm verlängert, da die F-CPU dieses zusätzlich bearbeitet und die Ergebnisse mit dem Anwenderprogramm vergleicht.

Das Absicherungsprogramm, das automatisch vom System generiert wird, finden Sie im Systembausteinordner Ihrer F-CPU.

Beispiel

Abbildung 5-9: Anwender- und systemerstellte F-Bausteine



In diesem Kapitel werden Ihnen verschiedene Möglichkeiten zur Verkürzung der Übersetzungs- und Programmlaufzeit aufgezeigt.

Es ist je nach Anwendung nicht immer möglich, alle Vorschläge zu nutzen. Sie geben aber Aufschluss, warum bestimme Programmiermethoden kürzere

Übersetzungs- und Programmlaufzeiten als ein nicht-optimiertes Programm verursachen.

5.12.1 Vermeiden von zeitverarbeitenden Bausteine: TP, TON, TOF

Jeder zeitverarbeitende Baustein (TP, TON, TOF) erfordert im Absicherungscode zusätzliche Bausteine und Globaldatenkorrekturen.

Empfehlung

Verwenden Sie diese Bausteine so wenig wie möglich.

5.12.2 Vermeiden von tiefen Aufrufhierarchien

Tiefe Aufrufhierarchien vergrößern den Code der systemerstellten F-Bausteine, da ein größerer Umfang an Absicherungsfunktionen und Prüfungen notwendig ist. Wenn die Schachtelungstiefe von 8 überschritten wird, gibt das TIA Portal beim Übersetzen eine Warnung aus.

Empfehlung

Strukturieren Sie ihr Programm so, dass Sie nicht unnötig tiefe Aufrufhierarchien generieren.

5.12.3 Vermeiden von JMP/Label Strukturen

Wenn ein Bausteinaufruf per JMP/LABEL übersprungen wird, führt es zu einer zusätzlichen Absicherung in den systemseitigen F-Bausteinen. Hier muss zum übersprungenen Bausteinaufruf ein Korrekturcode ausgeführt werden. Das kostet Performance und Zeit bei der Übersetzung

Empfehlung

Vermeiden Sie JMP/Label-Strukturen so weit wie möglich, um die systemseitigen F-Bausteine zu reduzieren.

5.13 Datenaustausch zwischen Standard- und F-Programm

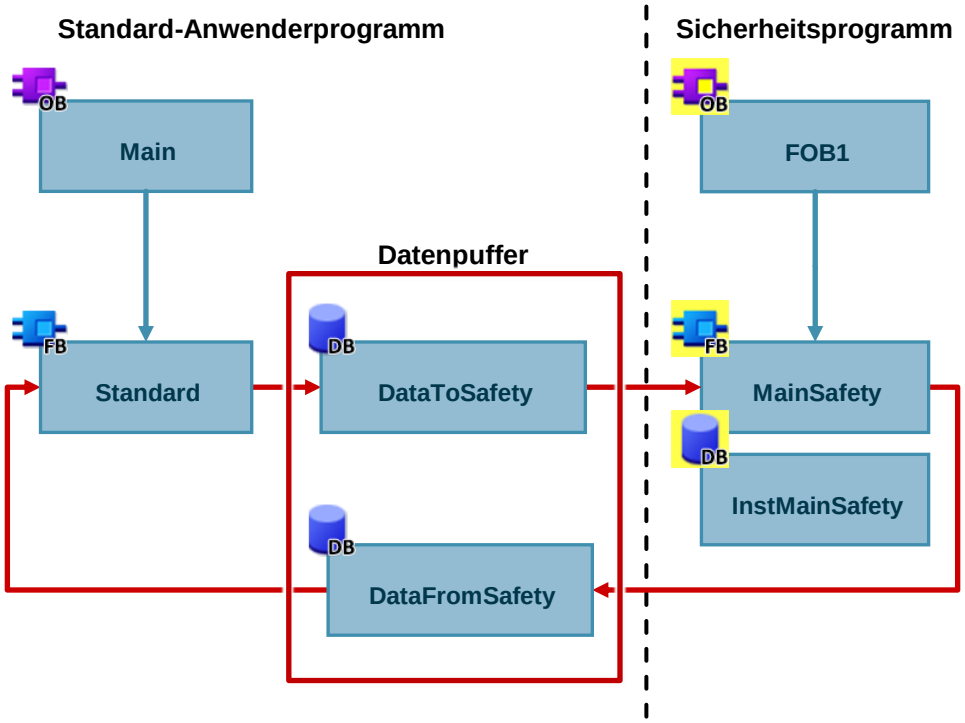
In einigen Fällen ist es notwendig, Daten zwischen dem Sicherheitsprogramm und dem Standard-Anwenderprogramm auszutauschen. Hierbei sollten folgenden Empfehlungen unbedingt beachtet werden, um die Datenkonsistenz zwischen Standard und Sicherheitsprogramm zu gewährleisten.

Empfehlungen

- Kein Datenaustausch über Merker (siehe Kapitel 4.2 Keine Merker, sondern globale Datenbausteine)
- Konzentrieren Sie Zugriffe zwischen Sicherheitsprogramm und dem Standard-Anwenderprogramm auf zwei Standard-DBs.

  Änderungen im Standardprogramm haben dadurch keinen Einfluss auf das Sicherheitsprogramm. Die Steuerung muss auch nicht im Betriebszustand STOP sein, um das Standardprogramm zu laden.

Abbildung 5-10: Datenaustausch zwischen Standard- und Sicherheitsprogramm



5.14 Sicherheitsprogramm testen

Zusätzlich zu den immer steuerbaren Daten eines Standard-Anwenderprogramms können Sie folgende Daten eines Sicherheitsprogramms im deaktivierten Sicherheitsbetrieb ändern:

- Prozessabbild von F-Peripherie
- F-DBs (außer DB für F-Ablaufgruppenkommunikation), Instanz-DBs von F-FBs
- F Peripherie DBs

Eigenschaften

- Das Steuern von F-Peripherie ist nur im RUN der F-CPU möglich.
- Von einer Beobachtungstabelle aus können Sie maximal 5 Ein-/ Ausgänge in einem Sicherheitsprogramm steuern.
- Sie können mehrere Beobachtungstabellen verwenden.
- Als Triggerpunkt müssen Sie "Zyklusbeginn" oder "Zyklusende", entweder "permanent" oder "einmalig" einstellen.
- Forcen ist für die F-Peripherie nicht möglich.
- Wenn Sie zum Testen dennoch Haltepunkte verwenden wollen, müssen Sie vorher den Sicherheitsbetrieb deaktivieren. Das führt weiterhin zu folgenden Fehlern:
  - Fehler bei der Kommunikation mit der F-Peripherie
  - Fehler bei der sicherheitsgerichteten CPU-CPU-Kommunikation

5.15 Betriebszustand STOP bei F-Fehlern

In folgenden Fällen wird der Betriebszustand STOP bei F-CPUs ausgelöst:

- Im Ordner "Systembausteine" dürfen Sie keine Bausteine hinzufügen, ändern oder löschen.
- -Zugriffe auf Instanz-DBs von F-FBs, die nicht im Sicherheitsprogramm aufgerufen werden.
- Die "Maximale Zykluszeit der F-Ablaufgruppe" darf nicht überschritten werden. Wählen Sie für "Maximale Zykluszeit der F-Ablaufgruppe" die maximale Zeit aus, die zwischen zwei Aufrufen dieser F-Ablaufgruppe vergehen darf (maximal 20000 ms).
- Wenn aus einem DB für F-Ablaufgruppenkommunikation Variablen gelesen werden, dessen Ablaufgruppe nicht bearbeitet wird (Main-Safety-Block der F-Ablaufgruppe wird nicht aufgerufen).
- Das Editieren der Startwerte in Instanz-DBs von F-FBs ist online und offline nicht zulässig und kann zum STOP der F-CPU führen.
- Der Main-Safety-Block darf keine Parameter beinhalten, da sie nicht versorgt werden können.
- Ausgänge von F-FCs müssen immer initialisiert werden.

5.16 Migration von Sicherheitsprogrammen

Informationen zur Migration von Sicherheitsprogrammen finden sie unter:

https://support.industry.siemens.com/cs/ww/de/view/109475826

5.17 Allgemeine Empfehlungen für Safety

Allgemein gelten folgende Empfehlungen im Umgang mit STEP 7 Safety und F-Baugruppen.

- Verwenden Sie wenn möglich immer F-Steuerungen. Somit ist eine spätere Erweiterung von Sicherheitsfunktionen einfach zu realisieren.
- Verwenden Sie immer ein Passwort für das Sicherheitsprogramm um nicht autorisierte Änderungen zu unterbinden. Das Passwort wird im "Safety Administrator"-Editor eingestellt.

6 Visualisierung automatisch anhand des Anwenderprogramms generieren

6.1 Einleitung

Ab TIA Portal V14 können Sie mithilfe des Optionspakets SiVArc (SIMATIC Visualization Architect) aus einer Visualisierungsbibliothek und dem Anwenderprogramm in der Steuerung die Visualisierung Ihrer Anlage automatisch generieren.

In diesem Kapitel optimieren Sie Ihr Anwenderprogramm für den Einsatz mit SiVArc.

Vorteile der Generierung mit SiVArc

- Automatische Generierung der Visualisierung mit Prozessanschluss
- Vereinheitlichen von Bedienoberflächen
- Einfache und durchgängige Anpassungen an Bedienbildern

Voraussetzungen

Die Grundvoraussetzung für den Einsatz von SiVArc ist ein hoher **Standardisierungsgrad** Ihrer Anlage. Eine Modularisierung der Anlage in einzelne Funktionsgruppen hat den Vorteil, dass SiVArc anhand dieser Funktionsgruppen das Bedienbild aus vorhandenen Bibliotheksbildern generieren und verschalten kann. Eine Standardisierung der Schnittstellen trägt zum effizienten Arbeiten und zur automatischen Generierung der Visualisierung bei.

Die Einhaltung der allgemeinen Empfehlungen dieses Programmierleitfadens hilft Ihnen dabei.

Hinweis

SiVArc ist ein Schnittstellenthema zwischen HMI und Steuerung. In diesem Kapitel wird SiVArc von der Steuerungsseite betrachtet.

Einen Einblick in den Funktionsumfang von SiVArc erhalten Sie unter den folgenden Links:

Anwendungsbeispiel "SiVArc Getting Started"
https://support.industry.siemens.com/cs/ww/de/view/109740350

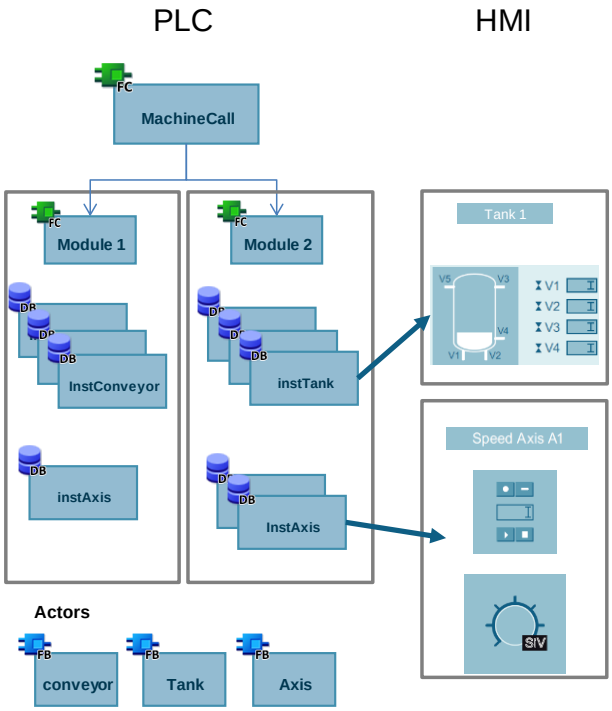
Handbuch SiVArc
https://support.industry.siemens.com/cs/ww/de/view/109755214

SITRAIN-Kurs: SIMATIC Visualization Architect, automatische HMI-Generierung
https://support.industry.siemens.com/cs/ww/de/view/109758628

6.2 Funktionsweise der automatischen Generierung

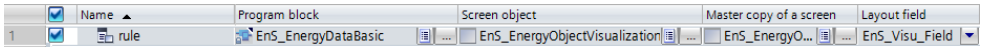
In Ihrem Anwenderprogramm werden standardisierte Bausteine, z. B. für eine Motoransteuerung, aufgerufen. Mithilfe von sogenannten SiVArc-Regeln verknüpfen Sie die aufgerufenen Bausteine mit Visualisierungselementen (Textfelder, IO-Felder, Bildbausteine usw.).

Abbildung 6-1: Beispiel-Aufrufhierarchie in der Steuerung



Anhand der Regeln generiert SiVArc für jeden Aufruf des angegebenen Bausteins das vorgegebene Visualisierungselement auf einer Kopie einer Bildvorlage.

Abbildung 6-2: SiVArc-Regel



Hinweis Sie haben zusätzlich die Möglichkeit, in der Steuerung die Ausführung der Regel einzuschränken oder zu blockieren.

6.3 HMI-Generat steuern

Sie haben folgende Möglichkeiten, um das HMI-Generat zu steuern:

- Die SiVArc-Generierung für bestimmte Aufrufe deaktivieren
- Die SiVArc-Generierung nach Funktionen oder Anlagenort sortieren
- Der SiVArc-Generierung weitere Eigenschaften hinzufügen

In SiVArc-Regeln können Sie unter anderem folgende Informationen aus der Steuerung verwenden:

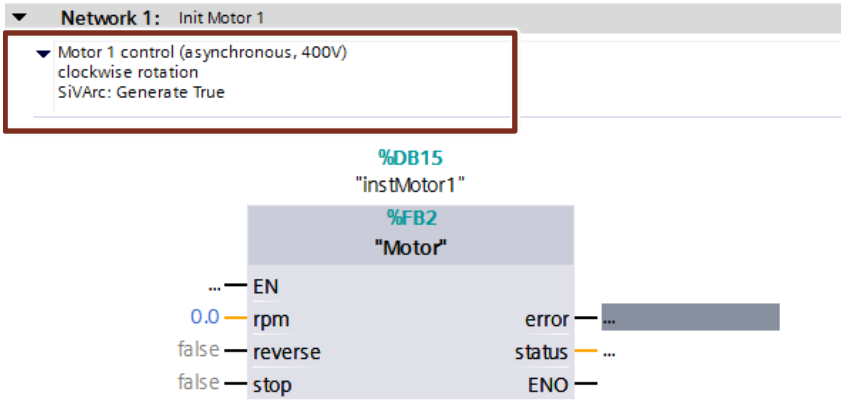
- Netzwerkkommentare
- SiVArc-Variablen

Hierüber haben Sie die Möglichkeiten in der Steuerung zusätzlich Informationen für SiVArc bereitzustellen, die Sie in den SiVArc-Regeln als Bedingung verwenden können.

6.3.1 Netzwerkkommentare zum Steuern verwenden

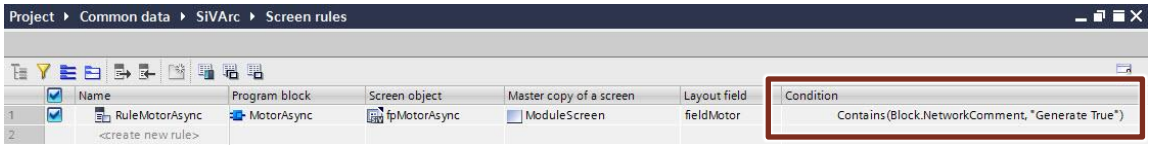
Sie können in den Netzwerkkommentaren Informationen zum Steuern des Generats ergänzen, nach denen die SiVArc-Regeln bei der Generierung suchen:

Abbildung 6-3: Netzwerkkommentar mit SiVArc-Ergänzung



Im Regeleditor verwenden Sie dann in der Spalte "Bedingung" die Funktion "Contains" und suchen nach den Informationen, z. B. "Contains(Block.NetworkComment, "String")".

Abbildung 6-4: Netzwerkkommentar in Regel verwenden



Mit dieser Funktion haben Sie einen freien Gestaltungsrahmen, um die Ausführung von Regeln einzuschränken oder eine Regel nur für einige Netzwerke auszuführen.

Empfehlung

Kennzeichnen Sie die Informationen für die SiVArc-Generierung in den Netzwerkkommentaren eindeutig, z. B. "SiVArc: Generate True".

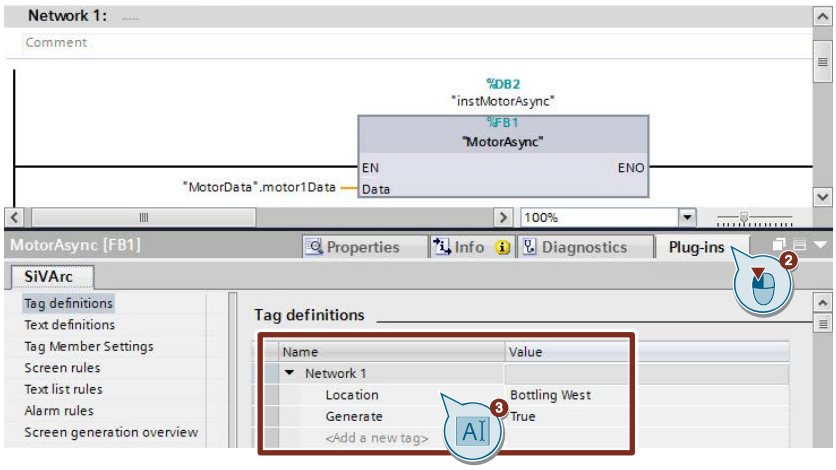
6.3.2 SiVArc-Variablen zum Steuern verwenden

Sie können spezielle SiVArc-Variablen für jedes Netzwerk im Baustein definieren und diese Variablen in den SiVArc-Regeln verwenden.

Um SiVArc-Tags zu erstellen, gehen Sie wie folgt vor:

4. Öffnen Sie den Baustein.
5. Wechseln Sie im Inspektorfenster in das Register "Plug-Ins".
6. Geben Sie für jedes Netzwerk in der Spalte "Name" den Namen der Variablen und in der Spalte "Wert" ("Value") einen Wert als String ein.

Abbildung 6-5: SiVArc-Variablen anlegen

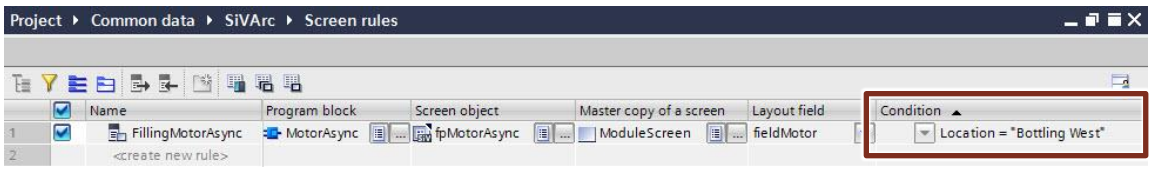


Im Regeleditor können Sie mit dem Ausdruck "Variablenname = "Wert"" die SiVArc-Variablen abfragen und dadurch eine Ausübung der Regel beeinflussen.

Geben Sie dazu im Regeleditor in der Spalte "Bedingung" die SiVArc-Variable ein, z. B. "Location = "Bottling West"".

Die Regel wird in diesem Beispiel nur ausgeführt, wenn in dem zu generierenden Netzwerk die SiVArc-Variable "Location" auf den Wert "Bottling West" gesetzt wurde.

Abbildung 6-6: SiVArc-Variable in Regel verwenden



Empfehlung

Verwenden Sie einheitliche Variablennamen zum Steuern des Generats, um die Erstellung der Regeln zu vereinfachen.
Die nachfolgende Tabelle zeigt ein Beispiel für SiVArc-Variablen:

Tabelle 6-1: Beispiel für SiVArc-Variablen

Name	Wert	Bedeutung
Generate	true, false	SiVArc generiert für dieses Netzwerk Elemente.
Location	Bottling West, Mixing usw.	Sie können hiermit Elemente einem HMI-Bild automatisch zuordnen

Hinweis Die in der Spalte "Bedeutung" genannten Auswirkungen müssen Sie selbst mithilfe der Bedingungen im Regeleditor festlegen. Die SiVArc-Variablen selbst haben keinen Einfluss auf das Generat.

6.4 Weitere Empfehlungen

Nur von WinCC unterstützte Zeichen verwenden

Verwenden Sie für die Bezeichnung von Variablen nur Zeichen, die von WinCC unterstützt werden.

Bei der Generierung der Bilder greift SiVArc auf die Bezeichner von Datenbausteinen, Variablen oder Netzwerkkommentaren zu. SiVArc entfernt dabei alle Zeichen, die von WinCC nicht unterstützt werden.
Dadurch kommt es zu Inkonsistenzen im Projekt.

Nicht unterstützte Zeichen sind:

- %,@, ?, ", /, \, <, >, ., :

Aufruf der Bausteine mit Programmiersprache FUP

Damit Sie das Generat über Netzwerkkommentare oder SiVArc-Variablen steuern können, rufen Sie die Bausteine, aus denen Visualisierungselemente generiert werden sollen, in der Programmiersprache FUP auf.

Hinweis Diese Empfehlung gilt bis einschließlich TIA Portal V15.

Ab TIA Portal V15.1 können auch SCL-Bausteine verwendet werden.

7 Die wichtigsten Empfehlungen

- Optimierte Bausteine verwenden
  - Kapitel 2.6 Optimierte Bausteine
- Datentyp VARIANT anstatt ANY verwenden
  - Kapitel 2.8.5 Datentyp VARIANT
- Programm übersichtlich und strukturiert gliedern
  - Kapitel 3.2 Programmbausteine
- Anweisungen als Multiinstanz einsetzen (TON, TOF ..)
  - Kapitel 3.2.5 Multiinstanzen
- Bausteine wiederverwendbar programmieren
  - Kapitel 3.2.9 Wiederverwendbarkeit von Bausteinen
- Symbolisch programmieren
  - Kapitel 3.6 Symbolische Adressierung
- Bei Hantierung von Daten mit ARRAY arbeiten
  - Kapitel 3.6.2 Datentyp ARRAY und indirekte Feldzugriffe
- PLC-Datentypen erstellen
  - Kapitel 3.6.5 Zugriff mit PLC-Datentypen auf E/A-Bereiche
- Bibliotheken zur Ablage von Programmelementen nutzen
  - Kapitel 3.7 Bibliotheken
- Keine Merker, sondern globale Datenbausteine
  - Kapitel 4.2 Keine Merker, sondern globale Datenbausteine

8 Anhang

8.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:
https://support.industry.siemens.com

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote – von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:
www.siemens.de/industry/supportrequest

SITRAIN – Training for Industry

Mit unseren weltweit verfügbaren Trainings für unsere Produkte und Lösungen unterstützen wir Sie praxisnah, mit innovativen Lernmethoden und mit einem kundenspezifisch abgestimmten Konzept.

Mehr zu den angebotenen Trainings und Kursen sowie deren Standorte und Termine erfahren Sie unter:
www.siemens.de/sitrain

Serviceangebot

Unser Serviceangebot umfasst folgendes:

- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:
https://support.industry.siemens.com/cs/sc

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für Apple iOS, Android und Windows Phone verfügbar:
https://support.industry.siemens.com/cs/ww/de/sc/2067

8.2 Links und Literatur

Tabelle 8-1

	Themengebiet
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Downloadseite des Beitrages https://support.industry.siemens.com/cs/ww/de/view/81318674
\3\	Programmierstyleguide für S7-1200 und S7-1500 https://support.industry.siemens.com/cs/ww/de/view/81318674
\4\	Bibliothek mit generellen Funktionen (LGF) für STEP 7 (TIA Portal) und S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/de/view/109479728
\5\	Bibliothek mit PLC-Datentypen (LPD) für STEP 7 (TIA Portal) und S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/de/view/109482396
\6\	TIA Portal - Ein Überblick der wichtigsten Dokumente und Links https://support.industry.siemens.com/cs/ww/de/view/65601780
\7\	STEP 7 (TIA Portal) Handbücher https://support.industry.siemens.com/cs/ww/de/ps/14673/man
\8\	S7-1200 (F) Handbücher https://support.industry.siemens.com/cs/ww/de/ps/13683/man
\9\	S7-1500 (F) Handbücher https://support.industry.siemens.com/cs/ww/de/ps/13716/man
\10\	ET 200SP CPU Handbücher https://support.industry.siemens.com/cs/ww/de/ps/13888/man
\11\	S7-1200 Getting Started https://support.industry.siemens.com/cs/ww/de/view/39644875
\12\	S7-1500 Getting Started https://support.industry.siemens.com/cs/ww/de/view/78027451
\13\	SIMATIC S7-1200 / S7-1500 Vergleichsliste für Programmiersprachen https://support.industry.siemens.com/cs/ww/de/view/86630375

8.3 Änderungshistorie

Tabelle 8-2

Version	Datum	Änderung
V1.0	09/2013	Erste Ausgabe
V1.1	10/2013	Korrekturen in folgenden Kapiteln: 2.6.3 Prozessoroptimale Datenablage bei S7-1500 2.12 Anwenderkonstanten 3.2.2 Funktionen (FC) 3.2.3 Funktionsbausteine (FB) 3.4.3 Lokaler Speicher
V1.2	03/2014	Neue Kapitel: 2.6.4 Konvertierung zwischen optimierten und nicht optimierten Variablen 2.6.6 Kommunikation mit optimierten Daten 2.9.1 MOVE Anweisungen 2.9.2 VARIANT Anweisungen 3.6.5 Zugriff mit PLC-Datentypen auf E/A-Bereiche Erweiterungen in folgenden Kapiteln: 2.2 Begriffe 2.3 Programmiersprachen 2.6 Optimierte Bausteine 2.10 Symbolik und Kommentare 3.2 Programmbausteine 3.5 Remanenz 4.3 Programmieren von "Takt Bits" Diverse Korrekturen in unterschiedlichen Kapiteln
V1.3	09/2014	Neue Kapitel: 2.8.4 Unicode-Datentypen 2.10.2 Kommentarzeilen in Beobachtungstabellen 2.12 Anwenderkonstanten 3.2.10 Autonummerierung von Bausteinen 5 STEP 7 Safety im TIA Portal Erweiterungen in folgenden Kapiteln: 2.7 Bausteineigenschaften 2.8 Neue Datentypen bei S7-1200/1500 2.9 Anweisungen 2.10 Symbolik und Kommentare 3.6.4 Datentyp STRUCT und PLC-Datentypen 3.7 Bibliotheken Diverse Korrekturen in unterschiedlichen Kapiteln

Version	Datum	Änderung
V1.4	11/2015	Neue Kapitel: 2.6.5 Parameterübergabe zwischen Bausteinen mit optimiertem und nicht optimierten Zugriff 3.3.3 Übersicht zur Übergabe von Parametern 3.10.5 Richtige Verwendung von FOR, REPEAT und WHILE-Schleifen 5.12 Optimierung der Übersetzungs- und Programmlaufzeit
V1.5	03/2017	Neue Kapitel: 2.7.3 Bausteinschnittstelle – Bausteinparameter ausblenden (ab V14) 2.9.4 Vergleich von Variablen aus PLC-Datentypen (ab V14) 2.9.5 Mehrfachzuweisung (ab V14) 3.2.6 Instanz als Parameter übergeben (V14) 3.6.3 Formalparameter Array [*] (ab V14) 3.6.7 SCL-Netzwerke in KOP und FUP (ab V14) 3.10.4 Strukturierung mit dem Schlüsselwort REGION (ab V14) 3.10.11 Unnötige IF-Anweisungen Diverse Korrekturen in unterschiedlichen Kapiteln
V1.6	12/2018	Neues Kapitel: 6 Visualisierung automatisch anhand des Anwenderprogramms generieren Anpassung Titelblatt und rechtliche Hinweise
V1.6.1	02/2019	Ergänzungen im Kapitel "Weitere Performance-Empfehlungen"