DIN EN IEC 61131-3

DIN

ICS 25.040.40; 35.060; 35.080; 35.240.50

Entwurf

Einsprüche bis 2023-10-11
Vorgesehen als Ersatz für
DIN EN 61131-3 Beiblatt
1:2005-04;
vorgesehen als Ersatz für
DIN EN 61131-3:2014-06

Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen (IEC 65B/1229/CDV:2023); Deutsche und Englische Fassung prEN IEC 61131-3:2023

Programmable controllers –
Part 3: Programming languages
(IEC 65B/1229/CDV:2023);
German and English version prEN IEC 61131-3:2023

Automates programmables –
Partie 3: Langages de programmation
(IEC 65B/1229/CDV:2023);
Version allemande et anglaise prEN IEC 61131-3:2023

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2023-08-11 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@vde.com möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, Merianstraße 28, 63069 Offenbach am Main.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 505 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

AVA Documents Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nationales Vorwort

Die Deutsche Fassung des europäischen Dokuments prEN IEC 61131-3:2023-05 „Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 65B/1229/CDV:2023-05 „Programmable controllers – Part 3: Programming languages“ unverändert in den Entwurf prEN IEC 61131-3:2023-05 übernommen worden.

Da die Deutsche Fassung noch nicht endgültig mit der Englischen und Französischen Fassung abgeglichen ist, ist die englische Originalfassung des IEC-CDV entsprechend der diesbezüglich durch die IEC erteilten Erlaubnis beigefügt. Die Nutzungsbedingungen für den deutschen Text des Norm-Entwurfes gelten gleichermaßen auch für den englischen IEC-Text.

Das internationale Dokument wurde vom IEC/SC 65B „Measurement and control devices“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet und unterliegt dem Copyright der IEC. Das Dokument wurde den nationalen Komitees zur Stellungnahme vorgelegt.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für dieses Dokument ist das nationale Arbeitsgremium DKE/K 962 „SPS“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

Aktuelle Informationen zu diesem Dokument können über die Internetseiten von DKE (www.dke.de) und DIN (www.din.de) durch eine Suche nach der Dokumentennummer aufgerufen werden.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe des Dokuments.

Der Zusammenhang der zitierten Dokumente mit den entsprechenden deutschen Dokumenten ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. IEC ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Änderungen

Gegenüber DIN EN 61131-3:2014-06 und DIN EN 61131-3 Beiblatt 1:2005-04 wurden folgende Änderungen vorgenommen:

a) Die wesentliche Erweiterung ist die Aufnahme von UTF-8-Folgen und den damit verbundenen Funktionen.

b) Anhang B enthält eine vollständige Liste der verglichen mit IEC 61131-3:2013 hinzugefügten, gestrichenen oder als abgelehnt eingestuften Eigenschaften.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen

Inhalt

Seite

1 Anwendungsbereich 9
2 Normative Verweisungen 9
3 Begriffe 9
4 Architekturmodelle 19
4.1 Software-Modell 19
4.2 Kommunikationsmodell 21
4.3 Programmiermodell 22
5 Normerfüllung 25
5.1 Allgemeines 25
5.2 Eigenschaftentabellen 25
5.3 Übereinstimmungserklärung des Herstellers 25
6 Gemeinsame Elemente 27
6.1 Verwendung von Druckbuchstaben 27
6.1.1 Zeichensatz 27
6.1.2 Bezeichner 27
6.1.3 Schlüsselwörter 28
6.1.4 Verwendung von Leerzeichen 28
6.1.5 Kommentare 28
6.2 Pragma 29
6.3 Literale – Externe Darstellung von Daten 29
6.3.1 Allgemeines 29
6.3.2 Numerische Literale und Folgeliterale 29
6.3.3 Zeichenfolge-Literale 30
6.3.4 Dauerliteral 33
6.3.5 Datums- und Uhrzeitliteral 33
6.4 Datentypen 34
6.4.1 Allgemeines 34
6.4.2 Elementare Datentypen (`BOOL`, `INT`, `REAL`, `STRING` usw.) 34
6.4.3 Allgemeine Datentypen 37
6.4.4 Anwenderdefinierte Datentypen 38
6.5 Variablen 53
6.5.1 Deklaration und Initialisierung von Variablen 53
6.5.2 Variablensektionen 55
6.5.3 `ARRAY`-Variablen mit variabler Länge 58
6.5.4 Konstante Variablen 60

AVA Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.5.5 Direkt dargestellte Variablen (%) ... 60
6.5.6 Gepufferte Variablen (RETAIN, NON_RETAIN) ... 62
6.6 Programm-Organisationseinheiten (POEs) ... 64
6.6.1 Gemeinsame Eigenschaften der POEs ... 64
6.6.2 Funktionen ... 78
6.6.3 Funktionsbausteine ... 113
6.6.4 Programme ... 132
6.6.5 Klassen ... 134
6.6.6 Interface ... 154
6.6.7 Objektorientierte Eigenschaften für Funktionsbaustein-Typen ... 164
6.6.8 Polymorphismus ... 170
6.7 Elemente der Ablaufsprache (AS) ... 173
6.7.1 Allgemeines ... 173
6.7.2 Schritte ... 174
6.7.3 Transitionen ... 175
6.7.4 Aktionen ... 178
6.7.5 Ablaufregeln ... 187
6.8 Konfigurationselemente ... 196
6.8.1 Allgemeines ... 196
6.8.2 Tasks ... 202
6.9 Synchronisation der nebenläufigen Ausführung ... 208
6.9.1 Allgemeines ... 208
6.9.2 Mutex ... 209
6.9.3 Objektorientierter Mutex ... 212
6.9.4 Semaphor ... 212
6.9.5 Objektorientierter Semaphor ... 214
6.10 Namensräume ... 214
6.10.1 Allgemeines ... 214
6.10.2 Deklaration ... 215
6.10.3 Verwendung ... 219
6.10.4 Namensraum-Direktive USING ... 220
7 Textsprachen ... 222
7.1 Gemeinsame Elemente ... 222
7.2 Strukturierter Text (ST) ... 223
7.2.1 Allgemeines ... 223
7.2.2 Ausdrücke ... 223
7.2.3 Anweisungen ... 225
8 Grafische Sprachen ... 230
8.1 Gemeinsame Elemente ... 230

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

8.1.1 Allgemeines 230
8.1.2 Darstellung von Variablen und Instanzen 231
8.1.3 Darstellung von Linien und Blöcken 233
8.1.4 Flussrichtung in Netzwerken 233
8.1.5 Auswertung von Netzwerken 234
8.1.6 Elemente der Ausführungssteuerung 235
8.2 Kontaktplan (KOP) 237
8.2.1 Allgemeines 237
8.2.2 Stromschienen 237
8.2.3 Verbindungselemente und Zustände 237
8.2.4 Kontakte 238
8.2.5 Spulen 239
8.2.6 Funktionen und Funktionsbausteine 240
8.2.7 Reihenfolge der Netzwerkauswertung 240
8.3 Funktionsbausteinsprache (FBS) 241
8.3.1 Allgemeines 241
8.3.2 Kombination von Elementen 241
8.3.3 Reihenfolge der Netzwerkauswertung 241
Anhang A (normativ) Formale Spezifikation der Sprachelemente 242
Anhang B (informativ) Liste der wichtigen Änderungen und Erweiterungen der 4. Ausgabe 250
Anhang C (informativ) 252
Literaturhinweise 254

Bilder

Bild 1 – Software-Modell 20
Bild 2 – Kommunikationsmodell 22
Bild 3 – Kombination von SPS-Sprachelementen 24
Bild 4 – Übereinstimmungserklärung des Herstellers (Beispiel) 26
Bild 5 – Hierarchie der allgemeinen Datentypen 38
Bild 6 – Initialisierung durch Literale und konstante Ausdrücke (Regeln) 39
Bild 7 – Schlüsselwörter für die Variablendeklaration (Zusammenfassung) 56
Bild 8 – Verwendung von `VAR_GLOBAL`, `VAR_EXTERNAL` und `CONSTANT` (Regeln) 57
Bild 9 – Bedingungen für den Anfangswert einer Variable (Regeln) 63
Bild 10 – Formale und nicht formale Darstellung eines Aufrufs (Beispiele) 69
Bild 11 – Regeln für die Datentypkonvertierung – implizit und/oder explizit (Zusammenfassung) 73
Bild 12 – Überblick über implizite Typkonvertierungen 74
Bild 13 – Datentypkonvertierung von Folgetypen zu Byte-Feldern 96
Bild 14 – Datentypkonvertierung von Byte-Feldern zu Folgetypen 96
Bild 15 – Verwendung der Eingangs- und Ausgangsparameter für Funktionsbausteine (Regeln) 123

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Bild 16 – Verwendung der Eingangs- und Ausgangsparameter für Funktionsbausteine (Veranschaulichung der Regeln) ... 124
Bild 17 – Standardfunktionsbausteine für Zeitgeber – Zeitdiagramme (Regeln) ... 130
Bild 18 – Empfohlene Implementierung in Pseudokode für Zeitgeber ... 132
Bild 19 – Überblick über die Vererbung und die Interface-Implementierung ... 135
Bild 20 – Vererbung von Klassen (Veranschaulichung) ... 144
Bild 21 – Interface mit abgeleiteten Klassen (Veranschaulichung) ... 156
Bild 22 – Vererbung von Interface und Klasse (Veranschaulichung) ... 161
Bild 23 – Funktionsbaustein-Typen mit optionalem Rumpf und optionalen Methoden (Veranschaulichung) ... 167
Bild 24 – Vererbung des Funktionsbaustein-Rumpfs mit `SUPER()` (Beispiel) ... 169
Bild 25 – Funktionsbaustein `ACTION_CONTROL` – Externe Schnittstelle (Zusammenfassung) ... 183
Bild 26 – Rumpf des Funktionsbausteins `ACTION_CONTROL` (Zusammenfassung) ... 185
Bild 27 – Aktionssteuerung (Beispiel) ... 187
Bild 28 – AS-Ablauf (Regeln) ... 194
Bild 29 – AS-Fehler (Beispiel) ... 195
Bild 30 – Konfiguration (Beispiel) ... 197
Bild 31 – Deklaration von `CONFIGURATION` und `RESOURCE` (Beispiel) ... 201
Bild 32 – Referenzimplementierung von Mutex-Funktionen ... 211
Bild 33 – Referenzimplementierung des objektorientierten Mutex ... 212
Bild 34 – Referenzimplementierung von Semaphor-Funktionen ... 213
Bild 35 – Referenzimplementierung des objektorientierten Semaphor ... 214
Bild 36 – Zugriffsmöglichkeiten bei Verwendung von Namensräumen (Regeln) ... 217
Bild 37 – Gemeinsame textuelle Elemente (Zusammenfassung) ... 223
Bild 38 – Zusammenhang zwischen ISO/IEC 10646 und IEC 61131-3 ... 252
Bild 39 – Beispiel für kodierte Folgen im Speicher ... 253

Tabellen

Tabelle 1 – Zeichensatz ... 27
Tabelle 2 – Bezeichner ... 27
Tabelle 3 – Kommentare ... 28
Tabelle 4 – Pragma ... 29
Tabelle 5 – Numerische Literale ... 30
Tabelle 6 – Zeichenfolge-Literale ... 31
Tabelle 7 – 2-Zeichen-Kombinationen in Zeichenfolgen ... 32
Tabelle 8 – Dauerliterale ... 33
Tabelle 9 – Datums- und Uhrzeitliteral ... 34
Tabelle 10 – Elementare Datentypen ... 34
Tabelle 11 – Deklaration von anwenderdefinierten Datentypen und Initialisierung ... 40
Tabelle 12 – Referenzierungsoperationen ... 52

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Tabelle 13 – Deklaration von Variablen 54
Tabelle 14 – Initialisierung von Variablen 55
Tabelle 15 – ARRAY-Variablen mit variabler Länge 58
Tabelle 16 – Direkt dargestellte Variablen 61
Tabelle 17 – Teilweiser Zugriff auf ANY_BIT-Variablen (ausgenommen BOOL) 66
Tabelle 18 – Grafische Ausführungssteuerung mit EN und ENO 71
Tabelle 19 – Funktionsdeklaration 79
Tabelle 20 – Funktionsaufruf 81
Tabelle 21 – Typisierte und überladene Funktionen 83
Tabelle 22 – Funktionen zur Datentypkonvertierung 85
Tabelle 23 – Datentypkonvertierung von numerischen Datentypen 87
Tabelle 24 – Datentypkonvertierung von Bit-Datentypen 90
Tabelle 25 – Datentypkonvertierung zwischen Bit-Typen und numerischen Typen 91
Tabelle 26 – Datentypkonvertierung von Datums- und Uhrzeittypen 93
Tabelle 27 – Datentypkonvertierung von Zeichentypen 94
Tabelle 28 – Numerische und arithmetische Funktionen 98
Tabelle 29 – Arithmetische Funktionen 99
Tabelle 30 – Bitschiebe-Funktionen 100
Tabelle 31 – Bitweise boolesche Funktionen 100
Tabelle 32 – Auswahlfunktionen 101
Tabelle 33 – Vergleichsfunktionen 102
Tabelle 34 – Funktionen für Zeichenfolgen 103
Tabelle 35 – Numerische Funktionen mit Datentypen für Zeit und Dauer 105
Tabelle 36 – Zusätzliche Funktionen für die Zeitdatentypen CONCAT und SPLIT 107
Tabelle 37 – Funktionen zur Konvertierung der Bytereihenfolge 111
Tabelle 38 – Funktionen für Aufzählungs- und Benannte-Werte-Datentypen 111
Tabelle 39 – Validierungsfunktionen 112
Tabelle 40 – Funktion ASSERT 112
Tabelle 41 – Deklaration des Funktionsbaustein-Typs 114
Tabelle 42 – Deklaration der Funktionsbaustein-Instanz 118
Tabelle 43 – Funktionsbaustein-Aufruf 120
Tabelle 44 – Bistabile Standardfunktionsbausteine 126
Tabelle 45 – Standardfunktionsbausteine zur Flankenerkennung 127
Tabelle 46 – Standardfunktionsbausteine für Zähler 128
Tabelle 47 – Standardfunktionsbausteine für Zeitgeber 129
Tabelle 48 – Programmdeklaration 133
Tabelle 49 – Klasse 136
Tabelle 50 – Deklaration von Klasseninstanzen 138
Tabelle 51 – Textueller Aufruf von Methoden – Formale und nicht formale Parameterliste 141

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Tabelle 52 – Interface 155
Tabelle 53 – Zuweisungsversuch 164
Tabelle 54 – Objektorientierter Funktionsbaustein-Typ 164
Tabelle 55 – AS-Schritt 175
Tabelle 56 – Transition und Transitionsbedingung der AS 176
Tabelle 57 – AS-Deklaration von Aktionen 179
Tabelle 58 – Verknüpfung Schritt/Aktion 180
Tabelle 59 – Aktionsblock 181
Tabelle 60 – Aktionsbestimmungszeichen 181
Tabelle 61 – Eigenschaften der Aktionssteuerung 187
Tabelle 62 – Kettenablauf – grafisch 189
Tabelle 63 – Deklaration von Konfigurationen und Ressourcen 199
Tabelle 64 – Task 203
Tabelle 65 – Mutex-Typ und -Operationen 210
Tabelle 66 – Semaphor-Typ und -Operationen 212
Tabelle 67 – Namensraum 218
Tabelle 68 – Möglichkeiten der geschachtelten Namensraumdeklaration 219
Tabelle 69 – Namensraum-Direktive `USING` 222
Tabelle 70 – Operatoren der Sprache ST 224
Tabelle 71 – Anweisungen der Sprache ST 225
Tabelle 72 – Grafische Elemente der Ausführungssteuerung 236
Tabelle 73 – Stromschienen und Verbindungselemente 238
Tabelle 74 – Kontakte 238
Tabelle 75 – Spulen 240

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

1 Anwendungsbereich

Dieser Teil von IEC 61131 legt die Syntax und Semantik von Programmiersprachen für speicherprogrammierbare Steuerungen fest, wie sie in IEC 61131-1 definiert sind.

Dieser Teil von IEC 61131 legt die Syntax und Semantik einer vereinheitlichten Reihe von Programmiersprachen für speicherprogrammierbare Steuerungen (SPS) fest. Diese Reihe umfasst die Textsprache „Strukturierter Text (ST)“ und die grafischen Sprachen „Kontaktplan (KOP)“ und „Funktionsbausteinsprache (FBS)“.

Ein zusätzlicher Satz von grafischen und äquivalenten textuellen Elementen, der „Ablaufsprache (AS)“ genannt wird, wird zur Strukturierung des internen Aufbaus von Programmen und Funktionsbausteinen definiert. Außerdem werden Konfigurationselemente definiert, die der Installation von SPS-Programmen in die SPS-Systeme dienen.

Zusätzlich werden Eigenschaften definiert, die die Kommunikation zwischen speicherprogrammierbaren Steuerungen und anderen Komponenten von automatisierten Systemen ermöglichen.

2 Normative Verweisungen

Die folgenden Dokumente werden im Text in solcher Weise in Bezug genommen, dass einige Teile davon oder ihr gesamter Inhalt Anforderungen des vorliegenden Dokuments darstellen. Bei datierten Verweisungen gilt nur die in Bezug genommene Ausgabe. Bei undatierten Verweisungen gilt die letzte Ausgabe des in Bezug genommenen Dokuments (einschließlich aller Änderungen).

IEC 61131-1, *Programmable controllers – Part 1: General information*

IEC 61131-5, *Programmable controllers – Part 5: Communications*

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)*

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

3 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC 61131-1 und die folgenden Begriffe.

ISO und IEC stellen terminologische Datenbanken für die Verwendung in der Normung unter den folgenden Adressen bereit:

– IEC Electropedia: verfügbar unter http://www.electropedia.org/

– ISO Online Browsing Platform: verfügbar unter http://www.iso.org/obp

3.1
Zugriffspfad
en **access path**
Verknüpfung eines symbolischen Namens mit einer Variablen zum Zweck der offenen Kommunikation

3.2
Aktion
en **action**
boolesche Variable oder eine Sammlung von Operationen, die zusammen mit einer zugehörigen Kontrollstruktur ausgeführt wird

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.3
Aktionsblock
en **action block**
grafisches Sprachelement, das eine boolesche Eingangsvariable benutzt, um den Wert einer booleschen Ausgangsvariablen oder die Freigabebedingung für eine Aktion zu bestimmen, und zwar nach einer vorbestimmten Kontrollstruktur

3.4
Aggregat
en **aggregate**
strukturierte Sammlung von Datenobjekten, die einen Datentyp bilden

3.5
Feld
en **array**
Aggregat, das aus Datenobjekten mit identischen Attributen besteht, von denen jedes eindeutig durch Indizieren referenziert werden kann

3.6
Zuweisung
en **assignment**
Mechanismus, um einer Variablen oder einem Aggregat einen Wert zu geben

3.7
Basistyp
en **base type**
Datentyp, Funktionsbaustein-Typ oder Klasse, von dem/der weitere Typen vererbt/abgeleitet sind

3.8
basisbezogene Zahl
en **based number**
Zahl, die mit einer festgelegten Basis ungleich zehn dargestellt ist

3.9
binär kodierte Dezimalzahl
BCD
en **binary coded decimal**
Kodierung von Dezimalzahlen, bei der jede Ziffer durch ihre eigene Binärfolge dargestellt ist

3.10
bistabiler Funktionsbaustein
en **bistable function block**
Funktionsbaustein mit zwei stabilen Zuständen, gesteuert durch einen oder mehr Eingänge

3.11
Bitfolge
en **bit string**
Datenelement, bestehend aus einem oder mehreren Bits

3.12
Bitfolge-Literal
en **bit string literal**
Literal, das direkt einen Bitfolge-Wert vom Datentyp `BOOL`, `BYTE`, `WORD`, `DWORD` oder `LWORD` darstellt

3.13
Rumpf
en **body**
Menge von Operationen einer Programm-Organisationseinheit

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.14
Aufruf
en **call**
Sprachkonstrukt, das die Ausführung einer Funktion, eines Funktionsbausteins oder einer Methode bewirkt

3.15
Zeichenfolge
en **character string**
Aggregat, das aus einer geordneten Folge von Zeichen besteht

3.16
Zeichenfolge-Literal
en **character string literal**
Literal, das direkt einen Zeichen- oder Zeichenfolge-Wert vom Datentyp `CHAR`, `WCHAR`, `UCHAR`, `STRING`, `WSTRING` oder `USTRING` darstellt

3.17
Klasse
en **class**
Programm-Organisationseinheit, bestehend aus:

- der Definition einer Datenstruktur
- einer Menge von Methoden und Propertys, die auf der Datenstruktur ausgeführt werden

3.18
Kommentar
en **comment**
Sprachkonstrukt zur Einbindung von Text ohne Auswirkung auf die Ausführung des Programmkodes

3.19
Konfiguration
en **configuration**
Sprachelement, das einem oder mehreren SPS-Systemen entspricht

3.20
Konstante
en **constant**
Sprachelement, das ein Datenelement mit einem festen Wert deklariert

3.21
Zähler-Baustein
en **counter function block**
Funktionsbaustein, der einen Wert für die Anzahl der erkannten Wechsel an einem oder mehreren festgelegten Eingängen akkumuliert

3.22
Datentyp
en **data type**
Sprachelement, das eine Menge von Werten zusammen mit einer Menge von zulässigen Operationen ermöglicht

3.23
Datum und Uhrzeit
en **date and time**
Datum im Jahr und Uhrzeit am Tag, dargestellt als ein einzelnes Sprachelement

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.24
Deklaration
en **declaration**
Mechanismus zur Festlegung der Definition eines Sprachelements

3.25
Begrenzungszeichen
en **delimiter**
Zeichen oder Zeichenkombination, das/die der Trennung von Programmsprachelementen dient

3.26
abgeleitete Klasse
en **derived class**
Klasse, die eine andere Klasse durch Vererbung erweitert; abgeleitete Klassen werden auch als „erweiterte Klassen“ oder „Kind-Klassen“ bezeichnet

3.27
direkt abgeleiteter Datentyp
en **directly derived data type**
Datentyp, der als Alias für einen anderen Datentyp deklariert wird

3.28
abgeleiteter Funktionsbaustein-Typ
en **derived function block type**
Funktionsbaustein-Typ, der durch Vererbung von einem anderen Funktionsbaustein-Typ oder Klassentyp erzeugt wird

3.29
direkte Darstellung
en **direct representation**
Mittel zur Darstellung einer Variablen in einem SPS-Programm, von der eine durch die Implementierung festgelegte Entsprechung zu einem physikalischen oder logischen Speicherort direkt bestimmt werden kann

3.30
Doppelwort
en **double word**
Datenelement, das 32 Bits enthält

3.31
dynamisches Binden
en **dynamic binding**
Situation, in der die Implementierung einer Methode entsprechend dem tatsächlichen Typ der Instanz während der Laufzeit ermittelt wird

3.32
Auswertung
en **evaluation**
Vorgang der Ermittlung eines Wertes für einen Ausdruck oder eine Funktion oder für die Ausgänge eines Netzwerks oder einer Funktionsbaustein-Instanz während der Programmausführung

3.33
Element zur Ausführungssteuerung
en **execution control element**
Sprachelement, das den Fluss der Programmausführung steuert

3.34
fallende Flanke
en **falling edge**
Wechsel einer booleschen Variablen von `TRUE (1)` nach `FALSE (0)`

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.35
Funktion
en **function**
Programm-Organisationseinheit, die den Zustand nicht puffert und die, wenn sie ausgeführt wird, typischerweise ein Ergebnis und möglicherweise zusätzliche Ausgangsvariablen liefert

3.36
Funktionsbaustein-Instanz
en **function block instance**
Instanz eines Funktionsbaustein-Typs

3.37
Funktionsbaustein-Typ
en **function block type**
Sprachelement, bestehend aus:

– der Definition einer Datenstruktur, gegliedert in Eingangs-, Ausgangs- und interne Variablen, und

– einer Menge von Operationen oder einer Menge von Methoden und Propertys, die mit den Elementen der Datenstruktur durchgeführt werden, wenn eine Instanz des Funktionsbaustein-Typs aufgerufen wird

3.38
Funktionsbausteinsprache
en **function block diagram**
Netzwerk, in dem die Knoten Funktionsbaustein-Instanzen, grafisch dargestellte Funktionen oder Methodenaufrufe, Variablen, Literale und Marken sind

3.39
allgemeiner Datentyp
en **generic data type**
Datentyp, der mehr als einen Typ von Daten darstellt

3.40
globale Variable
en **global variable**
Variable, deren Anwendungsbereich global ist

3.41
hierarchische Adressierung
en **hierarchical addressing**
direkte Darstellung eines Datenelements als ein Teil einer physikalischen oder logischen Hierarchie

BEISPIEL Ein Anschlusspunkt in einer Baugruppe, die in einem Baugruppenträger steckt, der wiederum in einem Schrank eingebaut ist usw.

3.42
Bezeichner
en **identifier**
Kombination von Buchstaben, Zahlen und Unterstrich-Zeichen, die mit einem Buchstaben oder einem Unterstrich beginnt und ein Sprachelement bezeichnet

3.43
Implementierung
en **implementation**
Produktversion einer SPS oder eines Programmier- und Diagnosewerkzeugs, die/das vom Hersteller zur Verfügung gestellt wird

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.44
Hersteller
en **implementer**
Lieferant der SPS oder des Programmier- und Diagnosewerkzeugs, die/das dem Anwender zum Programmieren einer SPS-Anwendung zur Verfügung gestellt wird

3.45
Vererbung
en **inheritance**
Erzeugung einer neuen Klasse, eines neuen Funktionsbaustein-Typs oder eines neuen Interface, basierend auf einer vorhandenen Klasse, einem vorhandenen Funktionsbaustein-Typ bzw. Interface

3.46
Anfangswert
en **initial value**
Wert, der einer Variablen zu Beginn ihrer Lebensdauer zugewiesen wird

3.47
Ein-Aus-Variable
en **in-out variable**
Variable, die benutzt wird, um einer Programm-Organisationseinheit einen Wert zu übergeben sowie einen Wert von der Programm-Organisationseinheit zurückzugeben

3.48
Eingangsvariable
en **input variable**
Variable, die benutzt wird, um einer Programm-Organisationseinheit oder einer Methode einen Wert zu übergeben

3.49
Instanz
en **instance**
individuelles, benanntes Exemplar einer Datenstruktur, verknüpft mit einem Funktionsbaustein-Typ, einer Klasse oder einem Programmtyp

3.50
Instanzname
en **instance name**
Bezeichner, der mit einer festgelegten Instanz verknüpft ist

3.51
Instanziierung
en **instantiation**
Erzeugung einer Instanz

3.52
ganzzahliges Literal
en **integer literal**
Literal, das direkt den Wert einer ganzen Zahl darstellt

3.53
Interface
en **interface**
Sprachelement im Zusammenhang mit der objektorientierten Programmierung, das eine Menge von Methoden- und Property-Prototypen enthält

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.54
Schlüsselwort
en **keyword**
lexikalische Einheit, die für die Verwendung in einer Programmiersprache reserviert ist

3.55
Marke
en **label**
Bezeichner, der mit einem Netzwerk verknüpft ist (KOP/FBS)

3.56
Sprachelement
en **language element**
Element, das durch ein Symbol auf der linken Seite der Produktionsregeln in der formalen Spezifikation identifiziert wird

3.57
Literal
en **literal**
lexikalische Einheit, die direkt einen Wert darstellt

3.58
logischer Speicherort
en **logical location**
Speicherort einer hierarchisch adressierten Variablen in einem Schema, das eine Relation zu der physikalischen Struktur der Eingänge, der Ausgänge und des Speichers der SPS haben kann, aber nicht muss

3.59
Methode
en **method**
gleich einer Funktion, mit dem Unterschied, dass sie für eine Instanz aufgerufen wird und ihre Operationen auf diese Instanz zugreifen können. Methoden können nur im Anwendungsbereich eines Funktionsbaustein-Typs oder eines Klassentyps definiert werden.

3.60
Methodenprototyp
en **method prototype**
Sprachelement, das nur den Namen und die Signatur einer Methode enthält

3.61
bezeichnetes Element
en **named element**
Element einer Struktur, das durch seinen zugehörigen Bezeichner benannt ist

3.62
Netzwerk
en **network**
Anordnung von Knoten und verbindenden Zweigen

3.63
numerisches Literal
en **numeric literal**
Literal, das direkt einen numerischen Wert darstellt, d. h. ein ganzzahliges Literal oder ein reelles Literal

3.64
Operation
en **operation**
Sprachelement, das eine elementare Funktionalität darstellt, die zu einer Programm-Organisationseinheit oder Methode gehört

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.65
Operand
en **operand**
Sprachelement, auf dem eine Operation durchgeführt wird

3.66
Operator
en **operator**
Symbol, das die Aktion darstellt, die in einer Operation auszuführen ist

3.67
override
en **override**
Schlüsselwort, das mit einer Methode in einer abgeleiteten Klasse oder in einem Funktionsbaustein-Typ für eine Methode mit dem gleichen Namen und der gleichen Signatur wie die Methode der Basis-Klasse oder des Funktionsbaustein-Typs mit einem neuen Methodenrumpf verwendet wird

3.68
Ausgangsvariable
en **output variable**
Variable, die benutzt wird, um einen Wert von einer Funktion, einem Funktionsbaustein, einem Programm oder einer Methode zurückzugeben

3.69
Parameter
en **parameter**
Variable, die verwendet wird, um einer Funktion, einem Funktionsbaustein, einem Programm oder einer Methode einen Wert (als Eingangs- oder Ein-Aus-Parameter) zu übergeben, oder Variable, die verwendet wird, um einen Wert von einer Funktion, einem Funktionsbaustein, einem Programm oder einer Methode (als Ausgangs- oder Ein-Aus-Parameter) zurückzugeben

3.70
Referenz
en **reference**
anwenderdefinierte Daten eines bestimmten Typs, die die Adresszuordnung zu einer Variablen oder zu einer Funktionsbaustein-Instanz oder Klasse enthalten

3.71
Stromfluss
en **power flow**
symbolischer Fluss des elektrischen Stroms in einem Kontaktplan, der das fortschreitende logische Berechnen eines Algorithmus beschreibt

3.72
Pragma
en **pragma**
Sprachkonstrukt zur Einbindung von Text in eine Programm-Organisationseinheit, das die Vorbereitung des Programms zur Ausführung beeinflussen darf

3.73
programmieren
en **program**
Entwerfen, Schreiben und Testen von Anwenderprogrammen

3.74
Programm-Organisationseinheit
en **program organization unit**
Funktion, Funktionsbaustein, Klasse oder Programm

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.75
Property
en **property**
Sprachkonstrukt, das typischerweise wie eine Mitgliedsvariable verwendet werden kann und durch spezielle Methoden zur Ermittlung oder Einstellung seines Wertes definiert wird

3.76
Property-Prototyp
en **property prototype**
Sprachkonstrukt, das nur den Namen und den Typ einer Property enthält

3.77
reelles Literal
en **real literal**
Literal, das einen Wert vom Typ `REAL` oder `LREAL` direkt darstellt

3.78
Ressource
en **resource**
Sprachelement, das einer Signalverarbeitungsfunktion und ihrer Mensch-Maschine-Schnittstelle sowie ihren Funktionen als Sensor- und Aktor-Schnittstelle entspricht

3.79
Ergebnis
en **result**
Wert, der als ein Ausgang einer Funktion oder Methode zurückgegeben wird

3.80
Rücksprung
en **return**
Sprachkonstrukt innerhalb einer Programm-Organisationseinheit, das ein Ende der Ausführungsfolgen in der Einheit bezeichnet

3.81
steigende Flanke
en **rising edge**
Wechsel einer booleschen Variablen von `FALSE (0)` nach `TRUE (1)`

3.82
Anwendungsbereich
en **scope**
Menge von Programm-Organisationseinheiten, in der eine Deklaration oder eine Marke gültig ist

3.83
Semantik
en **semantics**
Beziehungen zwischen den symbolischen Elementen einer Programmiersprache und ihrer Bedeutung, Auslegung und Anwendung

3.84
semigrafische Darstellung
en **semigraphic representation**
Darstellung von grafischen Informationen mittels eines begrenzten Zeichensatzes

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.85
Signatur
en **signature**
Menge von Informationen, die ein `FUNCTION`-, `FUNCTION BLOCK`- oder `METHOD`-Interface definiert und aus einer geordneten Liste von Parametern und dem Rückgabetyp (sofern vorhanden) besteht, wobei jeder Parameter durch seinen Namen, seinen Typ und seine Variablenart (Eingangs-, Ausgangs- oder Ein-Aus-Variable) definiert wird

3.86
Einzelelement-Variable
en **single-element variable**
Variable, die ein einzelnes Datenelement darstellt

3.87
statische Variable
en **static variable**
Variable, deren Wert von einem Aufruf bis zum nächsten gespeichert wird

3.88
Schritt
en **step**
Situation, in der das Verhalten einer Programm-Organisationseinheit in Bezug auf ihre Ein- und Ausgänge einer Menge von Regeln folgt, die durch die zugehörigen Aktionen des Schritts definiert sind

3.89
strukturierter Datentyp
en **structured data type**
zusammengesetzter Datentyp, der unter Verwendung einer Deklaration `STRUCT` deklariert wurde

3.90
Indizieren
en **subscripting**
Mechanismus zur Referenzierung eines Feldelements mittels einer Feldreferenz und eines oder mehrerer Ausdrücke, die die Position des Elements angeben, wenn sie ausgewertet werden

3.91
Task
en **task**
Element zur Ausführungssteuerung, das für eine periodische oder getriggerte Ausführung einer Gruppe von zugeordneten Programm-Organisationseinheiten sorgt

3.92
Zeitliteral
en **time literal**
Literal, das Daten vom Typ `TIME`, `LTIME`, `DATE`, `LDATE`, `TIME_OF_DAY`, `LTIME_OF_DAY`, `DATE_AND_TIME` oder `LDATE_AND_TIME` darstellt

3.93
Transition
en **transition**
Bedingung, bei der die Steuerung von einem oder mehreren Vorgängerschritten entlang einer gerichteten Verbindung zu einem oder mehreren Nachfolgerschritten übergeht

3.94
vorzeichenlose ganze Zahl
en **unsigned integer**
ganze Zahl, die nur einen Wert größer oder gleich Null haben kann und die positive sowie Nullwerte enthalten kann

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

3.95
vorzeichenloses ganzzahliges Literal
en **unsigned integer literal**
ganzzahliges Literal, das kein führendes Pluszeichen (+) oder Minuszeichen (−) enthält und das eine vorzeichenlose ganze Zahl darstellt

3.96
anwenderdefinierter Datentyp
en **user-defined data type**
Datentyp, der vom Anwender definiert wird

BEISPIEL Aufzählung, Feld oder Struktur.

3.97
Variable
en **variable**
Softwareelement, das nacheinander verschiedene Werte annehmen kann

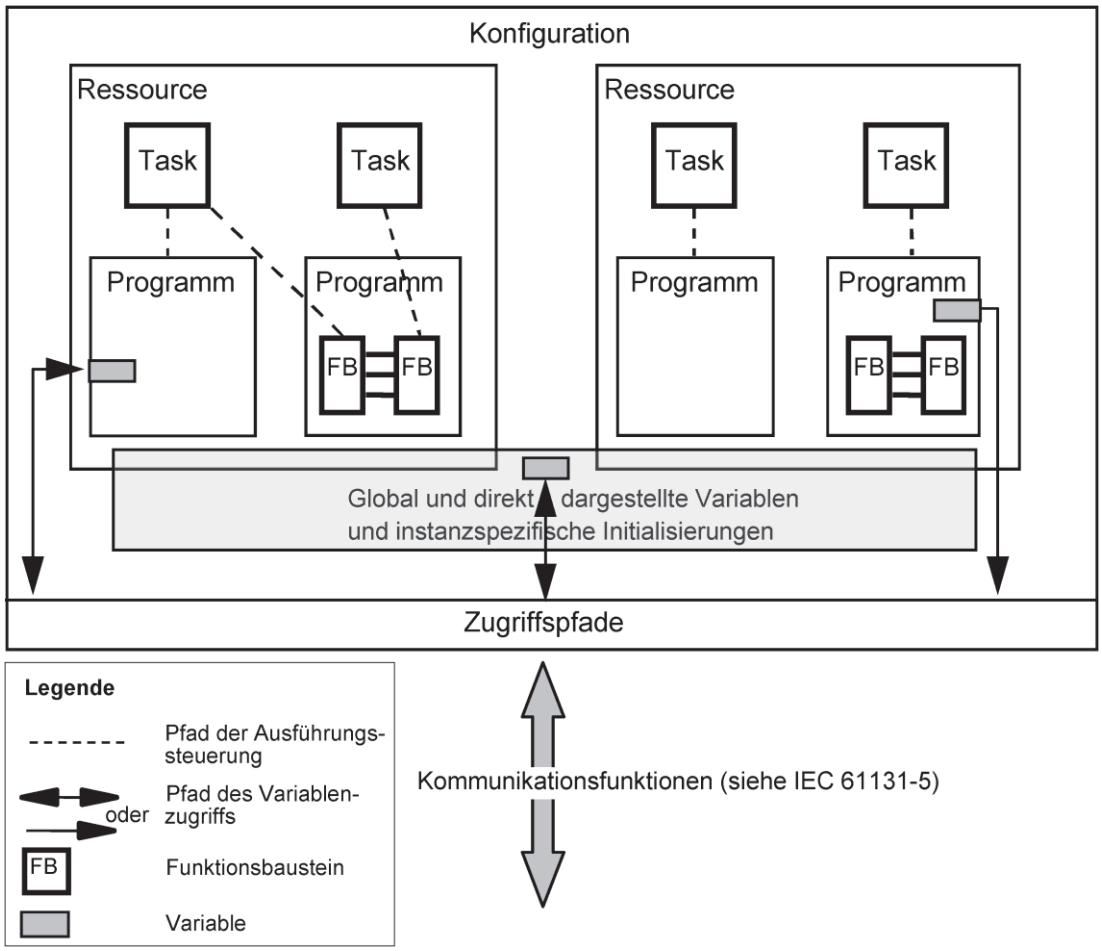
4 Architekturmodelle

4.1 Software-Modell

Die grundlegenden Hochsprachenelemente und ihre Beziehungen untereinander werden in Bild 1 veranschaulicht.

Sie bestehen aus Elementen, die mit den Sprachen programmiert werden, die in dieser Norm definiert sind, also aus Programmen, Funktionsbaustein-Typen, Klassen, Funktionen und Konfigurationselementen wie Konfigurationen, Ressourcen, Tasks, globalen Variablen, Zugriffspfaden und instanzspezifischen Initialisierungen, die der Installation von SPS-Programmen in SPS-Systemen dienen.

Documents
Document Sharing Hub



ANMERKUNG 1 Bild 1 dient nur zur Veranschaulichung. Die grafische Darstellung ist nicht normativ.

ANMERKUNG 2 In einer Konfiguration mit einer einzigen Ressource muss die Ressource nicht explizit dargestellt werden.

ANMERKUNG 3 Die Verwendung von Zugriffspfaden zur Bereitstellung von Daten für die Übertragung wird abgelehnt.

ANMERKUNG 4 Die direkte Zuordnung von Tasks zu Funktionsbaustein-Instanzen wird abgelehnt.

Bild 1 – Software-Modell

Eine Konfiguration ist ein Sprachelement, das einem SPS-System entspricht, wie es in IEC 61131-1 definiert ist. Eine Ressource entspricht einer Signalverarbeitungsfunktion und ihrer Mensch-Maschine-Schnittstelle (HMI, en: human-machine interface) und den Funktionen der Sensor- und Aktor-Schnittstelle (sofern vorhanden), wie sie in IEC 61131-1 definiert werden.

Eine Konfiguration enthält eine oder mehrere Ressourcen, von denen jede ein oder mehrere Programme enthält, die unter der Steuerung von null oder mehr Tasks ausgeführt werden.

Ein Programm darf null oder mehr Funktionsbaustein-Instanzen oder andere Sprachelemente enthalten, wie sie in diesem Teil von IEC 61131 definiert werden.

Eine Task ist in der Lage, die Ausführung einer Menge von Programmen und Funktionsbaustein-Instanzen z. B. periodisch anzustoßen.

Konfigurationen und Ressourcen können mithilfe der Bedienerschnittstellenfunktion, der Programmier-, Test- und Überwachungsfunktionen oder der Betriebssystemfunktion, die in IEC 61131-1 definiert sind, gestartet oder gestoppt werden. Das Starten einer Konfiguration muss die Initialisierung ihrer globalen Variablen

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

bewirken, gefolgt vom Starten aller Ressourcen in der Konfiguration. Das Starten einer Ressource muss die Initialisierung aller Variablen in der Ressource bewirken, gefolgt von der Freigabe aller Tasks in der Ressource. Das Stoppen einer Ressource muss das Sperren aller ihrer Tasks bewirken, während das Stoppen einer Konfiguration das Stoppen aller ihrer Ressourcen bewirken muss.

Die Mechanismen zur Steuerung von Tasks werden in 6.8.2 definiert, während die Mechanismen zum Starten und Stoppen von Konfigurationen und Ressourcen über Kommunikationsfunktionen in IEC 61131-5 definiert werden.

Programme, Ressourcen, globale Variablen, Zugriffspfade (und ihre zugehörigen Zugriffsberechtigungen) und Konfigurationen können durch die in IEC 61131-1 definierte Kommunikationsfunktion geladen und gelöscht werden. Das Laden und Löschen einer Konfiguration oder Ressource muss gleichbedeutend mit dem Laden und Löschen aller Elemente sein, die sie enthält.

Die Abbildung der Sprachelemente auf die Kommunikationsobjekte muss so erfolgen, wie es in IEC 61131-5 definiert ist.

4.2 Kommunikationsmodell

Bild 2 veranschaulicht die Wege, über die die Werte von Variablen zwischen Software-Elementen ausgetauscht werden können.

Wie in Bild 2a) gezeigt, können die Variablenwerte innerhalb eines Programms direkt durch die Verbindung des Ausgangs des einen Programmelements mit dem Eingang des anderen ausgetauscht werden. Diese Verbindung wird in den grafischen Sprachen explizit gezeigt und ist in den Textsprachen implizit angegeben.

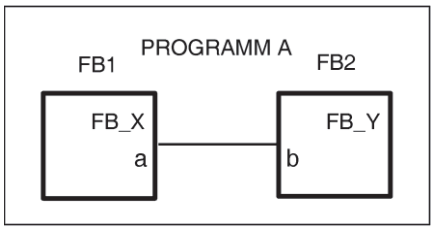
Variablenwerte können zwischen Programmen derselben Konfiguration über globale Variablen ausgetauscht werden, wie es die Variable x in Bild 2b) veranschaulicht. Diese Variablen müssen als `GLOBAL` in der Konfiguration und als `EXTERNAL` im Programm deklariert werden.

Wie in Bild 2c) veranschaulicht, können die Werte von Variablen zwischen verschiedenen Teilen eines Programms, zwischen Programmen in derselben oder verschiedenen Konfigurationen oder zwischen einem SPS-Programm und einem Nicht-SPS-System mithilfe der Kommunikationsbausteine ausgetauscht werden, die in IEC 61131-5 definiert sind.

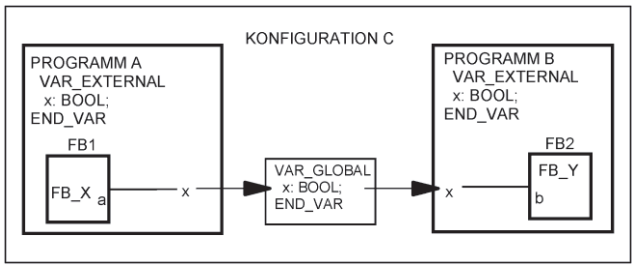
Außerdem können speicherprogrammierbare Steuerungen oder Nicht-SPS-Systeme Daten übertragen, die durch die Zugriffspfade zur Verfügung gestellt werden, wie es in Bild 2d) dargestellt ist, indem die in IEC 61131-5 definierten Mechanismen benutzt werden. Diese Funktionalität wird abgelehnt.

Documents
Document Sharing Hub

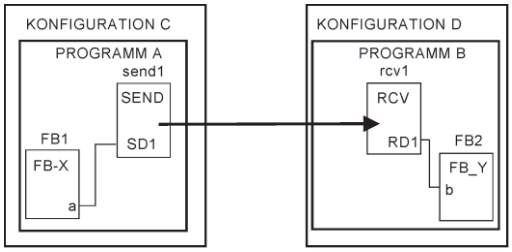
Full document details: E DIN EN IEC 61131-3:2023-09



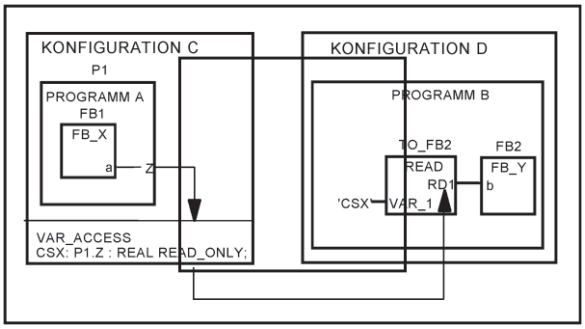
a) **Datenflussverbindung innerhalb eines Programms**



b) **Kommunikation über `GLOBAL`-Variablen**



c) **Funktionsbausteine für die Kommunikation**



d) **Kommunikation über Zugriffspfade**

ANMERKUNG 1 Bild 2 dient nur zur Veranschaulichung. Die grafische Darstellung ist nicht normativ.

ANMERKUNG 2 In diesen Beispielen wird zugrunde gelegt, dass die Konfigurationen C und D jeweils eine einzelne Ressource haben.

ANMERKUNG 3 Die Einzelheiten der Kommunikationsfunktionsbausteine sind in Bild 2 nicht dargestellt.

ANMERKUNG 4 Die Zugriffspfade können auf direkt dargestellte Variablen, globale Variablen oder Eingangs-, Ausgangs- oder interne Variablen von Programmen oder Funktionsbaustein-Instanzen deklariert werden.

ANMERKUNG 5 IEC 61131-5 legt die Mittel fest, mit denen sowohl SPS- als auch Nicht-SPS-Systeme Zugriffspfade zum Lesen und Schreiben von Variablen benutzen können.

ANMERKUNG 6 Die Eigenschaft 9 `VAR_ACCESS` in Tabelle 48 wird abgelehnt und ist in der nächsten Ausgabe dieses Teils von IEC 61131 möglicherweise nicht mehr enthalten.

Bild 2 – Kommunikationsmodell

4.3 Programmiermodell

Bild 3 veranschaulicht die wesentlichen Beziehungen zwischen den häufigsten SPS-Sprachelementen. Die Kombinationen dieser Elemente müssen den folgenden Regeln gehorchen:

1) Datentypen müssen mit Standard- oder anwenderdefinierten Datentypen deklariert werden.
2) Funktionen können mit Standard- oder anwenderdefinierten Datentypen und Funktionen deklariert werden.
3) Für diese Deklaration müssen die Mechanismen benutzt werden, die für die Sprache ST, KOP oder FBS definiert sind.
4) Funktionsbaustein-Typen können mit Standard- oder anwenderdefinierten Datentypen, Funktionen und Funktionsbaustein-Typen deklariert werden.
5) Diese Deklarationen müssen die Mechanismen benutzen, die für die Sprache ST, KOP oder FBS definiert sind, und können Elemente der Ablaufsprache (AS) enthalten.
6) Optional dürfen objektorientierte Funktionsbaustein-Typen oder Klassen definiert werden, die Methoden enthalten können. Objektorientierte Funktionsbaustein-Typen und Klassen dürfen auch abgeleitet werden und können Interfaces implementieren (nicht dargestellt).

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

7) Ein Programm muss deklariert werden, indem Standard- oder anwenderdefinierte Datentypen, Funktionen, Funktionsbausteine und Klassen benutzt werden.

8) Diese Deklaration muss die Mechanismen benutzen, die für die Sprache ST, KOP oder FBS definiert sind, und kann Elemente der Ablaufsprache (AS) einschließen.

9) Programme können mithilfe von Konfigurationselementen, d. h. globalen Variablen, Ressourcen, Tasks und Zugriffspfaden, zu Konfigurationen kombiniert werden.

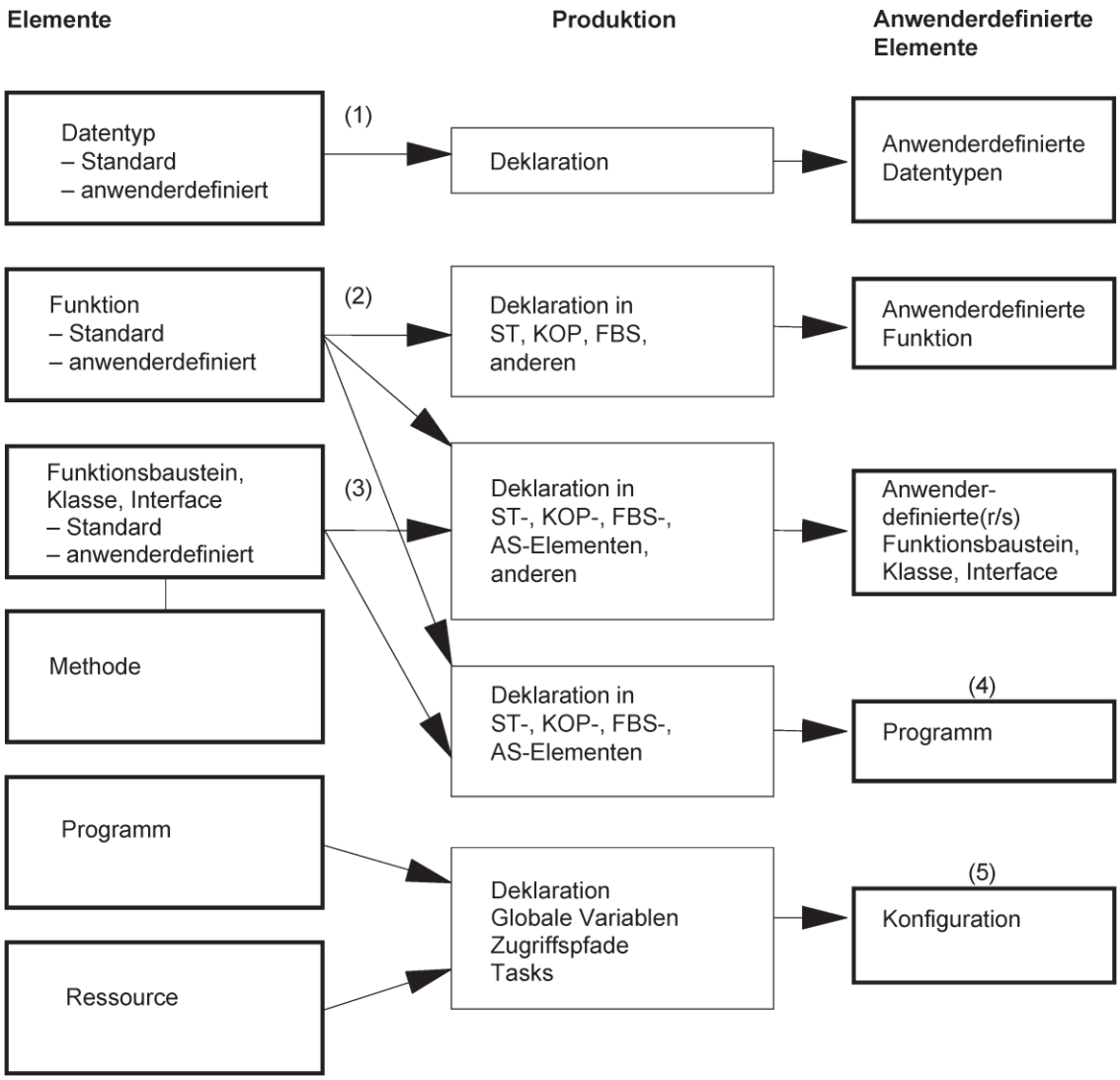
Zum Programmieren einer Funktion, eines Funktionsbaustein-Typs, eines Programms oder einer Methode darf auch eine andere Programmiersprache als die in dieser Norm definierten verwendet werden.

Es muss als Fehler angesehen werden, wenn zwei Elemente denselben vollständig qualifizierten Namen haben (siehe 6.9). Ein Element darf ein Datentyp, eine Funktion, ein Funktionsbaustein, eine Klasse, eine globale Variable oder ein Namensraum sein.

ANMERKUNG Der vollständig qualifizierte Name ist identisch mit dem Namen, wenn Namensräume nicht unterstützt werden.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09



KOP: Kontaktplan

FBS: Funktionsbausteinsprache

ST: Strukturierter Text

Andere: Andere Programmiersprachen

ANMERKUNG 1 Die eingeklammerten Zahlen (1) bis (5) verweisen auf die entsprechenden Absätze 1) bis 5) oben.

ANMERKUNG 2 Datentypen werden in allen Produktionen benutzt. Die entsprechenden Verbindungen sind der Klarheit wegen in diesem Bild weggelassen.

Bild 3 – Kombination von SPS-Sprachelementen

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

5 Normerfüllung

5.1 Allgemeines

Ein SPS-Programmier- und -Diagnosewerkzeug (PADT, en: programming and debugging tool) nach IEC 61131-1, das den Anspruch erhebt, vollständig oder teilweise die Anforderungen dieses Teils von IEC 61131 zu erfüllen, muss sich genau so verhalten, wie nachstehend beschrieben.

a) Es muss eine Untermenge der Eigenschaften sowie die nachstehend definierte entsprechende Übereinstimmungserklärung des Herstellers zur Verfügung stellen.

b) Es darf nicht die Verwendung von Ersatz- oder zusätzlichen Sprachelementen erfordern, um eine der Eigenschaften umzusetzen.

c) Es muss ein Dokument zur Verfügung stellen, das alle herstellerspezifischen Erweiterungen festlegt. Dies sind alle Eigenschaften, die vom System akzeptiert werden, die aber nicht zugelassen oder nicht festgelegt sind.

d) Es muss ein Dokument zur Verfügung stellen, das alle herstellerspezifischen Abhängigkeiten festlegt. Dies umfasst die Implementierungsabhängigkeiten, die explizit in diesem Teil von IEC 61131 genannt werden, sowie die Grenzparameter wie maximale Länge, Anzahl, Größe und Wertebereich, die hier nicht explizit festgelegt werden.

e) Es muss ein Dokument zur Verfügung stellen, in dem alle Fehler aufgeführt werden, die erkannt werden können und von der Implementierung gemeldet werden. Dies umfasst die Fehler, die in diesem Teil explizit genannt werden, und die Fehler, die während der Aufbereitung des Programms für die Ausführung und während der Ausführung des Programms erkannt werden können.

   ANMERKUNG Fehler, die während der Programmausführung auftreten, werden in diesem Teil von IEC 61131 nur teilweise festgelegt.

f) Es darf keinen der in dieser Norm definierten Datentyp-Standardnamen, Funktions- oder Funktionsbausteinnamen für implementierungsspezifische Eigenschaften verwenden, deren Funktionalität sich von der in diesem Teil von IEC 61131 beschriebenen Funktionalität unterscheidet.

5.2 Eigenschaftentabellen

Die Tabellen in diesem Teil von IEC 61131 enthalten die in diesem Teil von IEC 61131 definierten Eigenschaften und haben alle den gleichen grundlegenden Aufbau. Die erste Spalte enthält die Nummer der Eigenschaft die zweite Spalte enthält eine Beschreibung der Eigenschaft und die weiteren Spalten können Beispiele oder weitere Informationen enthalten. Der gleiche Tabellenaufbau wird vorzugsweise auch in der Übereinstimmungserklärung des Herstellers verwendet.

5.3 Übereinstimmungserklärung des Herstellers

Der Hersteller darf eine beliebige konsistente Untermenge der Eigenschaften, die in den Eigenschaftentabellen aufgeführt sind, definieren und muss die zur Verfügung gestellte Untermenge in der sogenannten „Übereinstimmungserklärung des Herstellers“ ausweisen.

Die Übereinstimmungserklärung des Herstellers muss in der Dokumentation enthalten sein, die dem System beigefügt ist, oder sie muss vom System selbst ausgestellt werden.

Die Übereinstimmungserklärung des Herstellers muss die folgenden Informationen enthalten. Bild 4 zeigt ein Beispiel.

- Die allgemeinen Informationen müssen den Namen und die Adresse des Herstellers, den Namen und die Version des Produkts, den Typ und die Version der Steuerung und das Ausstellungsdatum umfassen.
- Für jede implementierte Eigenschaft muss (müssen) die Nummer der entsprechenden Eigenschaftentabelle, die Eigenschaftennummer und die anwendbaren Programmiersprachen angegeben werden.

  Optional dürfen die Überschrift und die Unterüberschrift der Eigenschaftentabelle, die Beschreibung der Eigenschaft, Beispiele, Herstellerbemerkungen und andere relevante Informationen angegeben werden.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nicht implementierte Tabellen und Eigenschaften müssen in der Übereinstimmungserklärung des Herstellers nicht ausgewiesen werden.

IEC 61131-3 SPS-Programmiersprachen

Hersteller: Herstellername, Adresse, usw.

Produkt: Produktname, Version usw. Steuerungstypspezifische Untermenge usw.

Datum: 2022-05-01

Dieses Produkt erfüllt die Anforderungen der Norm für die folgenden Spracheigenschaften:

Eigenschaft Nr.	Tabellennummer und Titel / Beschreibung der Eigenschaft		Normgerecht implementiert in der Sprache (✓)				Anmerkung des Herstellers
			KOP	FBS	ST		
	Tabelle 1 – Zeichensatz						
1	ISO/IEC 10646:2012, Information technology – Universal Coded Character SET (UCS)		✓	✓			
2a	Kleinbuchstaben:	a, b, c …	✓	✓	✓		Kein „ß, ü, ä, ö“
2b	Nummernzeichen:	# Siehe Tabelle 5	✓				
2c	Dollar-Zeichen:	$ Siehe Tabelle 6		✓			
	Tabelle 2 – Bezeichner						
1	Großbuchstaben und Zahlen:	`IW215`					
2	Groß- und Kleinbuchstaben, Zahlen, eingebettete Unterstriche						
3	Groß- und Kleinbuchstaben, Zahlen, führende und eingebettete Unterstriche						
	Tabelle 3 – Kommentare						
1	Einzeiliger Kommentar:	//…					
2a	Mehrzeiliger Kommentar:	(* … *)					
2b	Mehrzeiliger Kommentar:	/* … */					
3a	Geschachtelter Kommentar:	(* ..(* .. *) ..*)					
3b	Geschachtelter Kommentar:	/* .. /* .. */ .. */					
	Tabelle 4 – Pragma						
1	Pragma mit geschweiften Klammern:	{ … }					
	Tabelle 5 – Numerische Literale						
1	Ganzzahliges Literal:	`-12`					
2	Reelles Literal:	`-12.0`					
3	Reelles Literal mit Exponent:	`-1.34E-12`					
4	Binäres Literal:	`2#1111_1111`					
5	Hexadezimales Literal:	`16#FF`					
6	Boolesche Null und Eins						
7	Boolesches `FALSE` und `TRUE`						
8	Typisiertes Literal:	`INT#-123`					
	Usw.						

Bild 4 – Übereinstimmungserklärung des Herstellers (Beispiel)

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6 Gemeinsame Elemente

6.1 Verwendung von Druckbuchstaben

6.1.1 Zeichensatz

Tabelle 1 enthält den Zeichensatz der textuellen Sprachen und textuellen Elemente von grafischen Sprachen. Die Zeichen werden nach ISO/IEC 10646 dargestellt.

Tabelle 1 – Zeichensatz

Nr.	Beschreibung		
1	ISO/IEC 10646		
2a	Kleinbuchstaben[a]:		a, b, c
2b	Nummernzeichen:	#	Siehe Tabelle 5
2c	Dollar-Zeichen:	$	Siehe Tabelle 6

[a] Wenn Kleinbuchstaben unterstützt werden, darf die Groß-/Kleinschreibung in den Sprachelementen nicht relevant sein, ausgenommen innerhalb von Kommentaren nach 6.1.5, Folgeliteralen nach 6.3.3 und Variablen vom Typ `STRING`, `WSTRING` und `USTRING` nach 6.3.3.

6.1.2 Bezeichner

Ein Bezeichner ist eine Folge von Buchstaben, Ziffern und Unterstrichen, die mit einem Buchstaben oder Unterstrich-Zeichen beginnen muss.

Die Groß-/Kleinschreibung von Buchstaben darf in Bezeichnern nicht relevant sein, z. B. müssen die Bezeichner `abcd`, `ABCD` und `aBCd` gleich interpretiert werden.

Das Unterstrich-Zeichen muss in Bezeichnern relevant für die Interpretation sein, z. B. müssen `A_BCD` und `AB_CD` als unterschiedliche Bezeichner interpretiert werden. Mehrere führende und mehrere eingebettete Unterstriche sind nicht zulässig, z. B. sind die Zeichensequenzen `__LIM_SW5 and LIM__SW5` keine gültigen Bezeichner. Angehängte Unterstriche sind nicht zulässig, z. B. ist die Zeichensequenz `LIM_SW5_` kein gültiger Bezeichner.

Zur Eindeutigkeit müssen in allen Systemen, die die Verwendung von Bezeichnern unterstützen, mindestens sechs Zeichen unterstützt werden, z. B. muss `ABCDE1` in allen diesen Systemen als verschieden von `ABCDE2` interpretiert werden. Die maximale Anzahl von Zeichen, die in einem Bezeichner zulässig ist, ist herstellerspezifisch.

Tabelle 2 enthält die Eigenschaften von Bezeichnern sowie Beispiele.

Tabelle 2 – Bezeichner

Nr.	Beschreibung	Beispiele
1	Großbuchstaben und Zahlen: `IW215`	`IW215 IW215Z QX75 IDENT`
2	Groß- und Kleinbuchstaben, Zahlen, eingebettete Unterstriche	Alle oben aufgeführten zzgl.: `LIM_SW_5 LimSw5 abcd ab_Cd`
3	Groß- und Kleinbuchstaben, Zahlen, führende und eingebettete Unterstriche	Alle oben aufgeführten zzgl.: `_MAIN _12V7`

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.1.3 Schlüsselwörter

Schlüsselwörter sind eindeutige Kombinationen von Zeichen, die als individuelle syntaktische Elemente angewendet werden. Schlüsselwörter dürfen keine eingebetteten Leerzeichen enthalten. Die Groß-/Kleinschreibung der Zeichen darf in Schlüsselwörtern nicht relevant sein, z. B. sind die Schlüsselwörter `FOR` und `for` syntaktisch gleichbedeutend. Sie dürfen für keinen anderen Zweck genutzt werden, z. B. als Variablennamen oder Erweiterungen.

6.1.4 Verwendung von Leerzeichen

Dem Anwender muss es erlaubt sein, ein oder mehrere Leerzeichen überall im Text des SPS-Programms einzufügen, ausgenommen innerhalb von Schlüsselwörtern, Literalen, Aufzählungswerten, Bezeichnern, direkt dargestellten Variablen oder Kombinationen von Begrenzungszeichen, z. B. für Kommentare. Ein Leerzeichen ist sowohl als SPACE-Zeichen mit dem Kode-Wert 32 dezimal als auch als nicht druckbare Zeichen wie Tabulator, neue Zeile usw., für die keine Kodierung in ISO/IEC 10646 angegeben ist, definiert.

6.1.5 Kommentare

Es gibt verschiedene Arten von Anwenderkommentaren; diese sind in Tabelle 3 aufgeführt:

1) Einzeilige Kommentare beginnen mit der Zeichenkombination `//` und enden beim nächsten Zeilenvorschub, bei neuer Zeile, bei Seitenvorschub oder „Wagenrücklauf“.
2) In einzeiligen Kommentaren haben die Zeichenkombinationen `(*` und `*)` oder `/* and */` keine besondere Bedeutung.
3) Mehrzeilige Kommentare müssen am Anfang und Ende von der Sonderzeichenkombination `(*` bzw. `*)` eingeschlossen sein.

   Ein alternativer mehrzeiliger Kommentar mit den Zeichenkombinationen `/*` und `*/` darf ebenfalls zur Verfügung gestellt werden.

   In mehrzeiligen Kommentaren hat die Sonderzeichenkombination `//` keine spezielle Bedeutung.

Kommentare müssen überall im Programm erlaubt sein, wo Leerzeichen erlaubt sind, ausgenommen innerhalb von Zeichenfolgen-Literalen.

Kommentare dürfen keine syntaktische oder semantische Bedeutung in einer der Sprachen haben, die in dieser Norm definiert werden. Sie werden wie ein Leerzeichen behandelt.

Geschachtelte Kommentare werden folgendermaßen verwendet:

- Paare von (`*`, `*`), z. B. `(* ... (* GESCHACHTELT *)... *)` oder
- Paare von /`*`, `*`/, z. B. `/* ... /* GESCHACHTELT */... */`.

Tabelle 3 – Kommentare

Nr.	Beschreibung	Beispiele

1	Einzeiliger Kommentar mit // …	`X := 13; // Kommentar in einer Zeile`
		`// Ein einzeiliger Kommentar kann am Anfang der Zeile`
		`// beginnen.`

2a	Mehrzeiliger Kommentar mit (* … *)	`(* Kommentar *)`
		`(**************************`
		`Ein eingerahmter Kommentar über drei Zeilen`
		`**************************)`

2b	Mehrzeiliger Kommentar mit /* … */	`/* Kommentar in einer` `oder mehr Zeilen */`
3a	Geschachtelter Kommentar mit (* .. (* .. *) ..*)	`(* (* GESCHACHTELT *) *)`
3b	Geschachtelter Kommentar mit /* .. /* .. */ .. */	`/* /* GESCHACHTELT */ */`

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.2 Pragma

Wie in Tabelle 4 veranschaulicht, müssen Pragmas am Beginn und Ende durch geschweifte Klammern `{` und `}` eingeschlossen werden. Syntax und Semantik der einzelnen Pragma-Konstrukte sind herstellerspezifisch. Pragmas müssen überall im Programm erlaubt sein, wo Leerzeichen erlaubt sind, ausgenommen innerhalb von Zeichenfolgen-Literalen.

Tabelle 4 – Pragma

Nr.	Beschreibung	Beispiele

1	Pragma
	mit { … }
	geschweiften Klammern
		`{VERSION 2.0}`
		`{AUTHOR JHC}`
		`{x := 256, y := 384}`

6.3 Literale – Externe Darstellung von Daten

6.3.1 Allgemeines

Externe Darstellungen von Daten bei den verschiedenen SPS-Programmiersprachen müssen die numerischen Literale, Zeichenfolge-Literale und Zeitliterale umfassen.

Es ist notwendig, die externe Darstellung für zwei unterschiedliche Typen von zeitbezogenen Daten zu unterstützen:

- Daten für eine Dauer zum Messen und Steuern der verstrichenen Zeit eines Steuerereignisses
- Daten für eine Uhrzeit, die auch Datumsinformationen enthalten dürfen, zum Synchronisieren des Beginns und des Endes eines Steuerungsereignisses mit einer absoluten Zeitreferenz

6.3.2 Numerische Literale und Folgeliterale

Es gibt zwei Arten von numerischen Literalen: ganzzahlige Literale (integer) und reelle Literale (real). Ein numerisches Literal ist als eine Dezimalzahl oder eine basisbezogene Zahl definiert. Die maximale Anzahl von Ziffern für alle Arten von numerischen Literalen muss ausreichend sein, um den gesamten Bereich und die Genauigkeit der Werte aller Datentypen auszudrücken, die durch das Literal in einer gegebenen Implementierung dargestellt werden.

Einzelne Unterstrich-Zeichen „_“, die zwischen den Ziffern eines numerischen Literals eingefügt sind, dürfen nicht relevant sein. Eine andere Anwendung von Unterstrich-Zeichen in numerischen Literalen ist nicht zulässig.

Dezimale Literale müssen in herkömmlicher Dezimal-Schreibweise dargestellt werden. Reelle Literale müssen durch einen Dezimalpunkt gekennzeichnet sein. Ein Exponent gibt die Zehnerpotenz an, mit der die vorausgehende Zahl zu multiplizieren ist, um den dargestellten Wert zu erhalten. Dezimale Literale und ihre Exponenten können ein Vorzeichen „+“ oder „−“ haben.

Literale können auch zur Basis 2 oder 16 dargestellt werden. Die Basis muss in Dezimal-Schreibweise angegeben werden. Für die Basis 16 muss ein erweiterter Satz von Zeichen benutzt werden, der aus den Buchstaben `A` bis `F` besteht, die die übliche Bedeutung der Dezimalzahlen 10 bis 15 haben. Basisbezogene Zahlen dürfen kein Vorzeichen „+“ oder „−“ beinhalten. Sie werden als Bitfolge-Literale interpretiert.

Numerische Literale, die eine positive ganze Zahl darstellen, dürfen als Bitfolge-Literale verwendet werden.

Boolesche Daten müssen durch ganzzahlige Literale mit dem Wert null (`0`) oder eins (`1`) oder durch die Schlüsselwörter `FALSE` oder `TRUE` dargestellt werden.

Tabelle 5 enthält die Eigenschaften von numerischen Literalen sowie Beispiele.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Der Datentyp eines booleschen oder numerischen Literals kann durch Voranstellen eines Typpräfixes festgelegt werden, das sich aus dem Namen eines elementaren Datentyps und dem Zeichen „#“ zusammensetzt. Für Beispiele siehe Eigenschaft 9 in Tabelle 5.

Tabelle 5 – Numerische Literale

Nr.	Beschreibung	Beispiele	Erklärung
1	Ganzzahliges Literal	`-12, 0, 123_4, +986`	
2	Reelles Literal	`0.0, 0.4560, 3.14159_26`	

3	Reelles Literal mit Exponent	`-1.34E-12, -1.34e-12`
		`1.0E+6, 1.0e+6`
		`1.234E6, 1.234e6`	

4	Binäres Literal	
		`2#1111_1111`
		`2#1110_0000`
			Literal zur Basis 2
			255 dezimal
			224 dezimal

5	Eigenschaft 5 (oktale Literale) wurde gestrichen!		

6	Hexadezimales Literal	
		`16#FF` oder `16#ff`
		`16#E0` oder `16#e0`
			Literal zur Basis 16
			255 dezimal
			224 dezimal

7	Boolesche Null und Eins	`0 or 1`	
8	Boolesches `FALSE` und `TRUE`	`FALSE TRUE`	
9	Typisiertes Literal	`INT#-123`	`INT` -Darstellung des Dezimalwerts −123
		`INT#16#7FFF`	`INT` -Darstellung des Dezimalwerts 32767
		`WORD#16#AFF`	`WORD`-Darstellung des Hexadezimalwerts `0AFF`
		`WORD#1234`	`WORD`-Darstellung des Dezimalwerts 1234=16#4D2
		`UINT#16#89AF`	`UINT`-Darstellung des Hexadezimalwerts 89AF
		`BOOL#0`	
		`BOOL#1`	
		`BOOL#FALSE`	
		`BOOL#TRUE`	

6.3.3 Zeichenfolge-Literale

Zeichenfolge-Literale sind entweder Einzelbyte-, Doppelbyte- oder Unicode-Zeichen.

- Ein Einzelbyte-Zeichenfolge-Literal ist eine Sequenz von null oder mehr Zeichen aus den Zeichensätzen Basis-Lateinisch und Lateinisch-1, Ergänzung, (Kodepunkte 0 bis 255) nach ISO/IEC 10646. Es wird durch ein einfaches Anführungszeichen (`'`) eingeleitet und abgeschlossen. In Einzelbyte-Zeichenfolgen muss die 3-Zeichen-Kombination aus einem Dollar-Zeichen (`$`) gefolgt von zwei hexadezimalen Ziffern als die hexadezimale Darstellung des 8-Bit-Zeichenkodes interpretiert werden, wie in Eigenschaft 1g in Tabelle 6 dargestellt.
- Ein Doppelbyte-Zeichenfolge-Literal ist eine Sequenz von null oder mehr Zeichen aus dem Zeichensatz der Basic Multilingual Plane (Kodepunkte 0 bis 65535) nach ISO/IEC 10646. Es wird durch ein doppeltes Anführungszeichen (`"`) eingeleitet und abgeschlossen. In Doppelbyte-Zeichenfolgen muss die 5-Zeichen-Kombination aus einem Dollar-Zeichen „`$`“ gefolgt von vier hexadezimalen Ziffern als die hexadezimale Darstellung des 16-Bit-Zeichenkodes interpretiert werden, wie in Eigenschaft 2h in Tabelle 6 dargestellt.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

- Ein Unicode-Zeichenfolge-Literal ist eine Sequenz von null oder mehr Zeichen aus dem gesamten ISO/IEC-10646-Zeichensatz. Es wird durch die Zeichen `U#` und das einfache Anführungszeichen (`'`) eingeleitet und durch das einfache Anführungszeichen (`'`) abgeschlossen. Anstelle von `U#` darf der Hersteller die Präfixzeichen `USTRING#` verwenden. In diesen Literalen muss die 7-Zeichen-Kombination aus einem Dollar-Zeichen „`$`“ gefolgt von sechs hexadezimalen Ziffern als die hexadezimale Darstellung des 24-Bit-Zeichenkodes interpretiert werden, wie in Eigenschaft 5e in Tabelle 6 dargestellt.

ANMERKUNG Zusammenhang zwischen ISO/IEC 10646 und Unicode:

Wenngleich die Zeichenkodes und Kodierformate zwischen Unicode und ISO/IEC 10646 aufeinander abgestimmt sind, stellt die Unicode-Norm zusätzliche Anforderungen an Implementierungen, um sicherzustellen, dass sie die Zeichen über Plattformen und Anwendungen hinweg einheitlich behandeln. Zu diesem Zweck stellt der Unicode eine beträchtliche Menge von funktionalen Zeichenspezifikationen, Zeichendaten, Algorithmen und umfangreichem Hintergrundmaterial bereit, die in ISO/IEC 10646 nicht enthalten sind.

In allen Zeichenfolge-Literalen müssen Zeichenkombinationen aus „`${`“ gefolgt von hexadezimalen Ziffern und „`}`“ als die hexadezimale Darstellung des Zeichenkodes interpretiert werden, wie in Eigenschaft 6 in Tabelle 6 dargestellt. Es ist ein Fehler, wenn das Escape-Zeichen nicht mit dem Typ des Zeichenfolge-Literals dargestellt werden kann.

Tabelle 6 – Zeichenfolge-Literale

Nr.	Beschreibung	Beispiele
	Einzelbyte-Zeichen oder -Zeichenfolgen mit ‘ ‘	
1a	Leere Zeichenfolge (Länge null)	`''`
1b	Zeichenfolge der Länge eins oder Zeichen `CHAR`, das ein einzelnes Zeichen enthält	`'A'`
1c	Zeichenfolge der Länge eins oder Zeichen `CHAR`, das ein Leerzeichen enthält	`' '`
1d	Zeichenfolge der Länge eins oder Zeichen `CHAR`, das ein einfaches Anführungszeichen enthält	`'$''`
1e	Zeichenfolge der Länge eins oder Zeichen `CHAR`, das ein doppeltes Anführungszeichen enthält	`'"'`
1f	Unterstützung der 2-Zeichen-Kombinationen von Tabelle 7	`'$R$L'`
1g	Unterstützung der Zeichendarstellung mit „$“ und zwei hexadezimalen Ziffern	`'$0A'`
	Doppelbyte-Zeichen oder -Zeichenfolge mit `""` (ANMERKUNG 1)	
2a	Leere Zeichenfolge (Länge null)	`""`
2b	Zeichenfolge der Länge eins oder Zeichen `WCHAR`, das ein einzelnes Zeichen enthält	`"A"`
2c	Zeichenfolge der Länge eins oder Zeichen `WCHAR`, das ein Leerzeichen enthält	`" "`
2d	Zeichenfolge der Länge eins oder Zeichen `WCHAR`, das ein einfaches Anführungszeichen enthält	`"'"`
2e	Zeichenfolge der Länge eins oder Zeichen `WCHAR`, das ein doppeltes Anführungszeichen enthält	`"$""`
2f	Unterstützung der 2-Zeichen-Kombinationen von Tabelle 7	`"$R$L"`
2h	Unterstützung der Zeichendarstellung mit „$“ und vier hexadezimalen Ziffern	`"$00C4"`
	Typisierte Einzelbyte-Zeichen oder -Zeichenfolge-Literale mit #	
3a	Typisierte Zeichenfolge	`STRING#'OK'`
3b	Typisiertes Zeichen	`CHAR#'X'`
	Typisiertes Doppelbyte-Zeichenfolge-Literal mit # (ANMERKUNG 1)	
4a	Typisierte Doppelbyte-Zeichenfolge (mit doppelten Anführungszeichen)	`WSTRING#"OK"`
4b	Typisiertes Doppelbyte-Zeichen (mit doppelten Anführungszeichen)	`WCHAR#"X"`
4c	Typisierte Doppelbyte-Zeichenfolge (mit einfachen Anführungszeichen)	`WSTRING#'OK'`
4d	Typisiertes Doppelbyte-Zeichen (mit einfachen Anführungszeichen)	`WCHAR#'X'`

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nr.	Beschreibung	Beispiele
	Zeichenfolge-Literale, die den gesamten Unicode-Zeichensatz unterstützen	
5a	Unicode-Zeichen (mit Präfix `U#` und einfachen Anführungszeichen)	`U#'X'`
5b	Unicode-Zeichenfolge (mit Präfix `U#` und einfachen Anführungszeichen)	`U#'OK'`
5c	Unicode-Zeichen (mit Präfix `UCHAR#` und einfachen Anführungszeichen)	`UCHAR#'X'`
5d	Unicode-Zeichenfolge (mit Präfix `USTRING#` und einfachen Anführungszeichen)	`USTRING#'OK'`
5e	Unterstützung der Zeichendarstellung mit „$“ und sechs hexadezimalen Ziffern in einem Literal nach 5a bis 5d (siehe ANMERKUNG 2)	`U#'OK$01F600'`

6	Unterstützung der Zeichendarstellung mit „$“ und hexadezimalen Ziffern zwischen „{ }“	`'${9}'` → `'$t'`
		`"${100}"` → `"Ā"`
		`U#'${1F600}'`→ `'😀'`

ANMERKUNG 1 Wenn eine bestimmte Implementierung die Eigenschaft 4, jedoch nicht die Eigenschaft 2 unterstützt, darf der Hersteller eine herstellerspezifische Syntax und Semantik für die Verwendung der doppelten Anführungszeichen festlegen. ANMERKUNG 2 Jeder gültige ISO/IEC-10646-Kodepunkt kann mit sechs hexadezimalen Ziffern dargestellt werden, wenngleich seine UTF-8-Kodierung bis zu 4 Byte beanspruchen kann (siehe Anhang C).		

2-Zeichen-Kombinationen, die mit dem Dollar-Zeichen beginnen, müssen so interpretiert werden, wie in Tabelle 7 dargestellt, wenn sie in Zeichenfolgen auftreten.

Tabelle 7 – 2-Zeichen-Kombinationen in Zeichenfolgen

Nr.	Beschreibung	Kombinationen
1	Dollar-Zeichen	$$
2	Einzelnes Anführungszeichen	$'
3	Zeilenvorschub (en: line feed)	$L oder $l
4	Neue Zeile (en: newline)	$N oder $n
5	Seitenvorschub (en: form feed, page)	$P oder $p
6	Wagenrücklauf (en: carriage return)	$R oder $r
7	Tabulator	$T oder $t
8	Doppeltes Anführungszeichen	$"
ANMERKUNG 1 Das Zeichen „neue Zeile“ bietet ein implementierungsunabhängiges Mittel, um das Ende einer Datenzeile zu definieren; beim Drucken bewirkt es das Beenden einer Datenzeile und das Fortsetzen des Druckvorgangs am Anfang der nächsten Zeile. ANMERKUNG 2 Die Kombination $' ist nur innerhalb von Zeichenfolge-Literalen in einfachen Anführungszeichen zulässig. ANMERKUNG 3 Die Kombination $" ist nur innerhalb von Zeichenfolge-Literalen in doppelten Anführungszeichen zulässig.		

ANMERKUNG Hersteller dürfen zusätzliche herstellerspezifische Literale (z. B. zusätzliche UTF-8-Literale) festlegen, die Einzelbyte-Zeichenfolgen ergeben, die `STRING`-Variablen zugewiesen werden können. Eine direkte Beziehung zwischen einem Zeichen des herstellerspezifischen Literals und einem Zeichen der Einzelbyte-Zeichenfolge ist nicht sichergestellt.

BEISPIEL Zuordnung eines herstellerspezifischen Literals mit UTF-8-Kodierung zu einer `STRING`-Variable. Die Variablen name1 und name2 sind gleich.

```
VAR

  name1 : STRING := UTF8#'ノウサギ';
  name2 : STRING := '$E3$83$8E$E3$82$A6$E3$82$B5$E3$82$AE';
END_VAR
```


Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.3.4 Dauerliteral

Daten für eine Dauer müssen links durch eines der Schlüsselwort-Präfixe `T#`, `LT#`, `TIME#` und `LTIME#` begrenzt werden. Die Darstellung von Daten für eine Dauer in Tagen, Stunden, Minuten, Sekunden und Bruchteilen einer Sekunde oder einer Kombination hieraus muss wie in Tabelle 8 dargestellt unterstützt werden. Die niedrigstwertige Zeiteinheit kann als reelle Zahl ohne Exponent geschrieben werden.

Die Einheiten von Dauerliteralen können durch Unterstrich-Zeichen getrennt werden.

Der „Überlauf“ der höchstwertigen Einheit eines Dauerliterals ist zulässig, z. B. ist die Angabe `T#25h_15m` erlaubt.

Zeiteinheiten, wie z. B. Sekunden, Millisekunden usw., können in Groß- oder Kleinbuchstaben dargestellt werden.

Wie in Tabelle 8 veranschaulicht, sind sowohl positive als auch negative Werte für die Angabe einer Dauer zulässig.

Tabelle 8 – Dauerliterale

Nr.	Beschreibung	Beispiele
	Abkürzungen für die Dauer	
1a	`d`	Tag
1b	`h`	Stunde
1c	`m`	Minute
1d	`s`	Sekunde
1e	`ms`	Millisekunde
1f	`us`	Mikrosekunde
1g	`ns`	Nanosekunde
	Dauerliterale ohne Unterstriche	
2a	Kurzes Präfix	`T#14ms T#-14ms LT#14.7s T#14.7m T#14.7h t#14.7d t#25h15m lt#5d14h12m18s3.5ms t#12h4m34ms230us400ns`
2b	Langes Präfix	`TIME#14ms TIME#-14ms time#14.7s`
	Dauerliterale mit Unterstrich	
3a	Kurzes Präfix	`t#25h_15m t#5d_14h_12m_18s_3.5ms LTIME#5m_30s_500ms_100.1us`
3b	Langes Präfix	`TIME#25h_15m ltime#5d_14h_12m_18s_3.5ms LTIME#34s_345ns`

6.3.5 Datums- und Uhrzeitliteral

Die vorangestellten Schlüsselwörter (Präfixe) für Datums- und Uhrzeitliterale müssen Tabelle 9 entsprechen.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Tabelle 9 – Datums- und Uhrzeitliteral

Nr.	Beschreibung		Beispiele
1a	Datumsliteral	(langes Präfix)	`DATE#1984-06-25, date#2010-09-22`
1b	Datumsliteral	(kurzes Präfix)	`D#1984-06-25`
2a	Langes Datumsliteral	(langes Präfix)	`LDATE#2012-02-29`
2b	Langes Datumsliteral	(kurzes Präfix)	`LD#1984-06-25`
3a	Uhrzeitliteral	(langes Präfix)	`TIME_OF_DAY#15:36:55.36`
3b	Uhrzeitliteral	(kurzes Präfix)	`TOD#15:36:55.36`
4a	Langes Uhrzeitliteral	(kurzes Präfix)	`LTOD#15:36:55.36`
4b	Langes Uhrzeitliteral	(langes Präfix)	`LTIME_OF_DAY#15:36:55.36`
5a	Datums- und Uhrzeitliteral	(langes Präfix)	`DATE_AND_TIME#1984-06-25-15:36:55.360227400`
5b	Datums- und Uhrzeitliteral	(kurzes Präfix)	`DT#1984-06-25-15:36:55.360_227_400`
6a	Langes Datums- und Uhrzeitliteral	(langes Präfix)	`LDATE_AND_TIME#1984-06-25-15:36:55.360_227_400`
6b	Langes Datums- und Uhrzeitliteral	(kurzes Präfix)	`LDT#1984-06-25-15:36:55.360_227_400`

6.4 Datentypen

6.4.1 Allgemeines

Ein Datentyp ist eine Klassifizierung, die die möglichen Werte, die ausführbaren Operationen und die Art definiert, wie Werte für Literale und Variablen gespeichert werden.

6.4.2 Elementare Datentypen (`BOOL`, `INT`, `REAL`, `STRING` usw.)

6.4.2.1 Festlegung der elementaren Datentypen

Diese Norm legt eine Menge von (vordefinierten) elementaren Datentypen fest.

Die elementaren Datentypen, die Schlüsselwörter für jeden Datentyp, die Anzahl der Bits je Datenelement und der Wertebereich für jeden elementaren Datentyp müssen Tabelle 10 entsprechen.

Tabelle 10 – Elementare Datentypen

Nr.	Beschreibung	Schlüsselwort	Voreingestellter Anfangswert	N (Bits)[a]
1	Boolesche (boolean)	`BOOL`	`0, FALSE`	1[h]
2	Kurze ganze Zahl (short integer)	`SINT`	`0`	8[c]
3	Ganze Zahl (integer)	`INT`	`0`	16[c]
4	Doppelte ganze Zahl (double integer)	`DINT`	`0`	32[c]
5	Lange ganze Zahl (long integer)	`LINT`	`0`	64[c]
6	Vorzeichenlose kurze ganze Zahl (unsigned short integer)	`USINT`	`0`	8[d]
7	Vorzeichenlose ganze Zahl (unsigned integer)	`UINT`	`0`	16[d]
8	Vorzeichenlose doppelte ganze Zahl (unsigned double integer)	`UDINT`	`0`	32[d]
9	Vorzeichenlose lange ganze Zahl (unsigned long integer)	`ULINT`	`0`	64[d]
10	Reelle Zahl (real number)	`REAL`	`0.0`	32[e]

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nr.	Beschreibung	Schlüsselwort	Voreingestellter Anfangswert	N (Bits)[a]
11	Lange reelle Zahl (long real)	LREAL	0.0	64[f]
12a	Dauer (duration)	TIME	T#0s	--[b]
12b	Lange Dauer (long duration)	LTIME	LTIME#0s	64[m, q, s]
13a	Datum (nur) (date (only))	DATE		--[b, r]
13b	Langes Datum (nur) (long date (only))	LDATE	LDATE#1970-01-01	64[n, s]
14a	Uhrzeit (nur) (time of day (only))	TIME_OF_DAY or TOD	TOD#00:00:00	--[b]
14b	Lange Uhrzeit (nur) (long time of day (only))	LTIME_OF_DAY or LTOD	LTOD#00:00:00	64[p, q, s]
15a	Datum und Uhrzeit (date and time of day)	DATE_AND_TIME or DT		--[b, r]
15b	Langes Datum und Uhrzeit (long date and time of day)	LDATE_AND_TIME or LDT	LDT#1970-01-01-00:00:00	64[o, q, s]
16a	Einzelbyte-Zeichenfolge mit variabler Länge	STRING	'' (empty)	8[i, g, k, l]
16b	Doppelbyte-Zeichenfolge mit variabler Länge	WSTRING	"" (empty)	16[i, g, k, l]
16c	UTF-8-kodierte Zeichenfolge mit variabler Länge	USTRING	U#'' (empty)	--[g, k, l, t]
17a	Einzelbyte-Zeichen	CHAR	'$00'	8[g, l]
17b	Doppelbyte-Zeichen	WCHAR	"$0000"	16[g, l]
17c	4-Byte-Zeichen	UCHAR	U#'$000000'	32[g, l]
18	Bitfolge der Länge 8	BYTE	16#00	8[j, g]
19	Bitfolge der Länge 16	WORD	16#0000	16[j, g]
20	Bitfolge der Länge 32	DWORD	16#0000_0000	32[j, g]
21	Bitfolge der Länge 64	LWORD	16#0000_0000_0000_0000	64[j, g]

[a] Einträge in dieser Spalte müssen so interpretiert werden, wie es in den Tabellen-Fußnoten festgelegt ist.

[b] Der Wertebereich und die Genauigkeit der Darstellung in diesen Datentypen sind herstellerspezifisch.

[c] Der Wertebereich für Variablen dieses Datentyps reicht von $-(2^{N-1})$ bis $(2^{N-1})-1$.

[d] Der Wertebereich für Variablen dieses Datentyps reicht von 0 bis $(2^{N})-1$.

[e] Der Wertebereich für Variablen dieses Datentyps muss der Definition in IEC 60559 für das Basis-Gleitpunkt-Format mit einfacher Länge entsprechen. Ergebnisse von arithmetischen Anweisungen mit denormierten Werten, unendlich oder Werten, die keine Zahl sind, sind herstellerspezifisch.

[f] Der Wertebereich für Variablen dieses Datentyps muss der Definition in IEC 60559 für das Basis-Gleitpunkt-Format mit doppelter Länge entsprechen. Ergebnisse von arithmetischen Anweisungen mit denormierten Werten, unendlich oder Werten, die keine Zahl sind, sind herstellerspezifisch.

[g] Ein numerischer Wertebereich ist für diesen Datentyp nicht anwendbar.

[h] Die möglichen Werte von Variablen dieses Datentyps müssen 0 und 1 sein, entsprechend den Schlüsselwörtern FALSE und TRUE.

[i] Der Wert von N gibt die Anzahl der Bits/Zeichen für diesen Datentyp an.

[j] Der Wert N gibt die Anzahl der Bits in der Bitfolge für diesen Datentyp an.

[k] Die maximal zulässige Länge von STRING-, WSTRING- und USTRING-Variablen ist herstellerspezifisch.

[l] Bei CHAR, STRING, WCHAR, WSTRING, UCHAR und USTRING werden den Zeichen Kodepunkte (Zeichenkodierung) nach ISO/IEC 10646 zugewiesen (siehe 6.4.2.2 und Anhang C).

[m] Der Datentyp LTIME ist eine 64-Bit-Ganzzahl mit Vorzeichen und hat die Einheit Nanosekunden.

[n] Der Datentyp LDATE ist eine 64-Bit-Ganzzahl mit Vorzeichen und hat die Einheit Nanosekunden und die Startzeit Mitternacht am 1970-01-01.

[o] Der Datentyp LDT ist eine 64-Bit-Ganzzahl mit Vorzeichen und hat die Einheit Nanosekunden und die Startzeit Mitternacht am 1970-01-01.

[p] Der Datentyp LTOD ist eine 64-Bit-Ganzzahl mit Vorzeichen und hat die Einheit Nanosekunden und die Startzeit Mitternacht TOD#00:00:00..

[q] Die Genauigkeit der Aktualisierung der Werte dieses Zeitformats ist herstellerspezifisch, d. h. der Wert wird in Nanosekunden angegeben, er kann aber jede Mikrosekunde oder Millisekunde aktualisiert werden.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nr.	Beschreibung	Schlüsselwort	Voreingestellter Anfangswert	N (Bits)[a]

[r] Der voreingestellte Anfangswert dieses Datentyps ist herstellerspezifisch.

[s] In den langen Zeitformaten werden Schaltsekunden nicht gezählt, d. h. ein Tag hat stets 86 400 s, ein Nicht-Schaltjahr 365 Tage und ein Schaltjahr 366 Tage. Es wird der gregorianische Kalender verwendet.

[t] Die Zeichen werden entsprechend UTF-8 nach ISO/IEC 10646 binär dargestellt (siehe ANMERKUNG und Anhang C).

ANMERKUNG Ein einzelnes Zeichen belegt je nach dem Kodepunkt des Zeichens 1 bis 4 Kodeeinheiten. Kodepunkte im Bereich `16#00-16#7F` werden mit einer einzelnen Kodeeinheit dargestellt. Kodepunkte im Bereich `16#80-16#7FF` werden mit zwei Kodeeinheiten dargestellt. Kodepunkte im Bereich `16#800-16#FFFF` werden mit drei Kodeeinheiten und Kodepunkte im Bereich `16#10000-16#10FFFF` mit vier Kodeeinheiten dargestellt. Eine Kodeeinheit hat einen Umfang von einem Byte.

6.4.2.2 Elementare Datentypfolgen (`STRING`, `WSTRING`, `USTRING`)

Die Definition von Folge-Datentypen beruht auf ISO/IEC 10646. Anhang C enthält einen allgemeinen Überblick über ISO/IEC 10646.

Die unterstützte maximale Länge von Werten des Typs `STRING`, `WSTRING` und `USTRING` muss herstellerspezifisch sein.

Die explizite maximale Länge n wird in der zugehörigen Deklaration in eckigen Klammern angegeben, „`[n]`“. Die explizite maximale Länge n darf den herstellerspezifischen unterstützten Höchstwert nicht überschreiten.

ANMERKUNG Dies ist die Anzahl von ASCII-Zeichen, die von der Folge dargestellt werden können. Bei dem Typ USTRING geben die unterstützte maximale Länge und die explizite maximale Länge die Kodeeinheiten-Kapazität der Folge an.

Der Zugriff auf einzelne Zeichen einer Folge, die Elemente vom Datentyp `CHAR`, `WCHAR` oder `UCHAR` verwendet, mittels eckiger Klammern und der Position des Zeichens in der Folge, beginnend mit Position 1, muss unterstützt werden.

`U` sei eine Variable vom Typ `USTRING[n]`, die die Sequenz von Zeichen `C1`, `C2`, ..., `Ci`, ..., `Cm` enthält.

Die Zuweisung `U[i]:=X` beschreibt die Ersetzung von `Ci` durch `X`. Wenn die resultierende Zeichensequenz nicht in `U` passt, wird eine minimale Anzahl von Zeichen am Ende der Folge abgeschnitten.

BEISPIEL 1 `STRING, WSTRING und CHAR, WCHAR.`

a) Deklaration

```
VAR
     String1:   STRING[10] := 'ABCD';
     String2:   STRING[10] := '';
     aWStrings: ARRAY [0..1] OF WSTRING := ["1234", "5678"];
     UString1: USTRING[10] := U#'A+Ä-$01F68D'; //5 UCHAR mit 9 Kodeeinheiten: 'A+Ä-🚍'
     Char1:     CHAR;
     WChar1:    WCHAR;
     UChar1:    UCHAR := U#'$01F600'; // Unicode-Zeichen '😀'
END_VAR
```

b) Verwendung von `STRING` und `CHAR`

```
     Char1 := String1[2];         //Ist äquivalent mit Char1 := 'B';
     String1[3] := Char1;         //Ergibt String1 := 'ABBD '
     String1[4] := 'B';           //Ergibt String1 := 'ABBB'
     String1[1] := String1[4];    //Ergibt String1 := 'BBBB'
     String2 := String1[2];  (*Ergibt String2 := 'B' ,
             wenn die implizite Konvertierung CHAR_TO_STRING implementiert wurde*)
```


Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

c) Verwendung von WSTRING und WCHAR

```
WChar1 := aWStrings[1][2];          //Ist äquivalent mit WChar1 := "6";
aWStrings[1][3] := WChar1;          //Ergibt aWStrings[1] := "5668"
aWStrings[1][4] := "6";             //Ergibt aWStrings[1] := "5666"
aWStrings[1][1] := aWStrings[1][4]; //Ergibt String1 := "6666"
aWStrings[0] := aWStrings[1][4];    (* Ergibt aWStrings[0] := "6"; ,
        wenn die implizite Konvertierung WCHAR_TO_WSTRING implementiert wurde *)
```

d) Verwendung von USTRING und UCHAR

```
UChar1 := UString1[4];  //Ergibt UChar1 = U#'-'
UString1[2] := UChar1;  //Ergibt UString1 = U#'A-Ä-🚆'
UString1[5] := U#'$01F600';       //Ergibt UString1 = U#'A-Ä-😀'
UString1[5] := USTRING#'$01F600'; //Äquivalent mit vorangehender Zeile
UString1[1] := USTRING#'$01F68D'; (* Ergibt UString1 = U#'🚆-Ä-'
Damit sie in USTRING[10] passt, wird die Folge von 12 Kodeeinheiten auf
8 Kodeeinheiten gekürzt, indem das letzte Zeichen gestrichen wird *)
UString1 := "ABCÄ"; // Ergibt UString1 = U#'ABCÄ' (implizite Konvertierung)
```

e) Äquivalente Funktionen (siehe 6.6.2.5.11)

```
Char1 := String1[2];
```

ist äquivalent mit

```
Char1 := STRING_TO_CHAR(Mid(IN := String1, L := 1, P := 2));
aWStrings[1][3] := WChar1;
```

ist äquivalent mit

```
REPLACE(IN1 := aWStrings[1], IN2 := WChar1, L := 1, P :=3 );
```

f) Fehlerfälle

```
String1[2] := WChar1; //Ungültige implizite Konvertierung
String1[2] := String2;
   //Erfordert implizite Konvertierung STRING_TO_CHAR , die nicht zulässig ist
Char1    := Uchar1; // Ungültige implizite Konvertierung
UString1 := U#'ä'; Uchar1 := Ustring1[2]; // Zugriff über Folge hinaus
Byte1    := Ustring1[1]; // Ungültige Konvertierung
UString1 := '${1F600}';  // Fehler, da das Zeichen 😀 nicht Teil des
                         // Zeichensatzes eines STRING-Literals ist
```

ANMERKUNG Die Datentypen für einzelne Zeichen (CHAR und WCHAR) können nur ein Zeichen enthalten. Folgen können mehrere Zeichen enthalten; daher können Folgen zusätzliche Verwaltungsinformationen erfordern, die bei einzelnen Zeichen nicht benötigt werden.

BEISPIEL 2

```
TYPE STR10: STRING[10] := 'ABCDEF'; END_TYPE
```

Hier wird der Typ STR10 durch eine maximale Länge von STR10 von 10 Zeichen deklariert, der voreingestellte Anfangswert lautet 'ABCDEF' und die Anfangslänge von Datenelementen des Typs STR10 beträgt 6 Zeichen.

```
VAR
     UString10: USTRING[10] := U#'A+Ä-$01F68D'; //5 UCHAR mit 9 Kodeeinheiten:'A+Ä-🚆'
END_VAR
```

Hier wird die Variable UString10 durch eine maximale Länge von 10 Kodeeinheiten definiert; der voreingestellte Anfangswert lautet 'A+Ä-🚆' und enthält 5 Unicode-Zeichen und 9 Kodeeinheiten.

6.4.3 Allgemeine Datentypen

Zusätzlich zu den elementaren Datentypen, die in Tabelle 10 aufgeführt sind, kann die Hierarchie der allgemeinen (en: generic) Datentypen, die in Bild 5 dargestellt ist, bei der Festlegung von Eingängen und Ausgängen von Standardfunktionen und Funktionsbausteinen angewendet werden. Die allgemeinen Datentypen werden durch das Präfix ANY gekennzeichnet.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Die Verwendung von allgemeinen Datentypen unterliegt den folgenden Regeln:

1) Der allgemeine Typ eines direkt abgeleiteten Typs muss der gleiche sein wie der allgemeine Typ des elementaren Typs, von dem er abgeleitet ist.
2) Der allgemeine Typ eines Unterbereich-Typs muss `ANY_INT`. sein.
3) Der allgemeine Typ von allen anderen abgeleiteten Datentypen, die in Tabelle 11 definiert sind, muss `ANY_DERIVED` sein.

Die Verwendung von allgemeinen Datentypen in anwenderdeklarierten Programm-Organisationseinheiten fällt nicht in den Anwendungsbereich dieser Norm.

Allgemeine Datentypen	Gruppen von elementaren Datentypen
ANY	
ANY_DERIVED	
ANY_ELEMENTARY	
ANY_MAGNITUDE	
ANY_DURATION	TIME, LTIME
ANY_NUM	
ANY_REAL	REAL, LREAL
ANY_INT	
ANY_UNSIGNED	USINT, UINT, UDINT, ULINT
ANY_SIGNED	SINT, INT, DINT, LINT
ANY_BIT	BOOL, BYTE, WORD, DWORD, LWORD
ANY_CHARS	
ANY_STRING	STRING, WSTRING, USTRING
ANY_CHAR	CHAR, WCHAR, UCHAR
ANY_DATE	DATE_AND_TIME (DT), LONG_DATE_AND_TIME (LDT), DATE, LDATE, TIME_OF_DAY (TOD), LONG_TIME_OF_DAY (LTOD)

Bild 5 – Hierarchie der allgemeinen Datentypen

6.4.4 Anwenderdefinierte Datentypen

6.4.4.1 Deklaration (`TYPE`)

6.4.4.1.1 Allgemeines

Der Zweck der anwenderdefinierten Datentypen ist die Verwendung in der Deklaration von weiteren Datentypen und in Variablendeklarationen.

Ein anwenderdefinierter Typ kann überall verwendet werden, wo ein Basistyp verwendet werden kann.

Die anwenderdefinierten Datentypen werden mit dem textuellen Konstrukt `TYPE...END_TYPE` deklariert.

Eine anwenderdefinierte Typdeklaration besteht aus:

- dem Namen des Typs;
- dem Zeichen „`:`“ (Doppelpunkt);
- der Deklaration des Typs selbst, wie sie in den folgenden Abschnitten definiert ist.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

BEISPIEL Typdeklaration.

```
TYPE
     myDatatype1: <data type declaration with optional initialization>;
END_TYPE
```

6.4.4.1.2 Initialisierung

Der Anfangswert ist durch den voreingestellten Anfangswert gegeben, sofern kein anwenderdefinierter Wert festgelegt wird.

Die anwenderdefinierte Initialisierung folgt der Typdeklaration und beginnt mit dem Zuweisungsoperator „`:=`“, gefolgt von dem Anfangswert bzw. den Anfangswerten.

Es dürfen Literale (z. B. `−123`, `1.55`, „`abc`“) oder konstante Ausdrücke (z. B. `12*24`) verwendet werden. Die verwendeten Anfangswerte müssen einen kompatiblen Typ haben, d. h. entweder den gleichen Typ oder einen Typ, der mittels der impliziten Typkonvertierung konvertiert werden kann.

Bei der Initialisierung von Datentypen müssen die in Bild 6 angegebenen Regeln befolgt werden.

Allgemeiner Datentyp	Initialisiert durch Literal	Ergebnis
`ANY_UNSIGNED`	Nicht negatives ganzzahliges Literal oder nicht negativer konstanter Ausdruck	Nicht negativer ganzzahliger Wert
`ANY_SIGNED`	Ganzzahliges Literal oder konstanter Ausdruck	Ganzzahliger Wert
`ANY_REAL`	Numerisches Literal oder konstanter Ausdruck	Numerischer Wert
`ANY_BIT`	Vorzeichenloses ganzzahliges Literal oder vorzeichenloser konstanter Ausdruck	Vorzeichenloser ganzzahliger Wert
`ANY_STRING`	Folgeliteral	Folgewert
`ANY_DATE`	Datums- und Uhrzeitliteral	Datums- und Uhrzeitwert
`ANY_DURATION`	Dauerliteral	Zeitwert

Bild 6 – Initialisierung durch Literale und konstante Ausdrücke (Regeln)

Tabelle 11 definiert die Eigenschaften der Deklaration von anwenderdefinierten Datentypen und der Initialisierung.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Tabelle 11 – Deklaration von anwenderdefinierten Datentypen und Initialisierung

Nr.	Beschreibung	Beispiel	Erklärung

1a
1b
	Datentyp Aufzählung
	(en: enumeration)
		`TYPE`
		`  ANALOG_SIGNAL_RANGE:`
		`    (BIPOLAR_10V,`
		`    UNIPOLAR_10V,`
		`    UNIPOLAR_1_5V,`
		`    UNIPOLAR_0_5V,`
		`    UNIPOLAR_4_20_MA,`
		`    UNIPOLAR_0_20_MA)`
		`         := UNIPOLAR_1_5V;`
		`END_TYPE`
			Initialisierung

2a
2b
	Datentyp Benannte Werte
	(en: named values)
		`TYPE`
		` Colors: DWORD`
		`    (Red  := 16#00FF0000,`
		`     Green:= 16#0000FF00,`
		`     Blue := 16#000000FF,`
		`     White:= Red OR Green OR Blue,`
		`     Black:= Red AND Green AND Blue)`
		`     := Green;`
		`END_TYPE`
			Initialisierung

3a
3b
	Datentyp Unterbereich
	(en: subrange)
		`TYPE`
		`  ANALOG_DATA: INT(-4095 .. 4095) := 0;`
		`END_TYPE`	

4a
4b
	Datentyp Feld (en: array)	`TYPE ANALOG_16_INPUT_DATA:`
		`  ARRAY [1..16] OF ANALOG_DATA`
		`       := [8(-4095), 8(4095)];`
		`END_TYPE`
			`ANALOG_DATA` siehe oben.
			
			Initialisierung

5a
5b
	Funktionsbaustein-Typen und Klassen als Feldelemente	`TYPE`
		`  TONs: ARRAY[1..50] OF TON`
		`      := [50((PT :=T#100ms))];`
		`END_TYPE`
			FB `TON` als Feldelement
			Initialisierung

6a
6b
	Strukturierter Datentyp	`TYPE ANALOG_CHANNEL_CONFIGURATION:`
		`  STRUCT`
		`    RANGE:     ANALOG_SIGNAL_RANGE;`
		`    MIN_SCALE: ANALOG_DATA := -4095;`
		`    MAX_SCALE: ANALOG_DATA :=  4095;`
		`  END_STRUCT;`
		`END_TYPE`
			`ANALOG_SIGNAL_RANGE`
			siehe oben

7a
7b
	Funktionsbaustein-Typen und Klassen als Strukturelemente	`TYPE`
		`  Cooler: STRUCT`
		`    Temp: INT;`
		`    Cooling: TOF := (PT := T#100ms);`
		`  END_STRUCT;`
		`END_TYPE`
			FB TOF als Strukturelement

8a
8b
	Strukturierter Datentyp mit relativer Adressierung AT	`TYPE`
		`  Com1_data: STRUCT`
		`   head        AT %B0:        INT;`
		`   length      AT %B2:        USINT := 26;`
		`   flag1       AT %X3.0:      BOOL;`
		`   end         AT %B25:       BYTE;`
		`  END_STRUCT;`
		`END_TYPE`
			Explizite Ablage ohne Überlappung

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Nr.	Beschreibung	Beispiel	Erklärung

9a	Strukturierter Datentyp mit relativer Adressierung AT und OVERLAP	`TYPE`
		`  Com2_data: STRUCT OVERLAP`
		`   head  AT %B0:  INT;`
		`   length  AT %B2:  USINT;`
		`   flag2  AT %X3.3:  BOOL;`
		`   data1  AT %B5:  BYTE;`
		`   data2  AT %B5:  REAL;`
		`   end  AT %B19:  BYTE;`
		`  END_STRUCT;`
		`END_TYPE`
			Explizite Ablage mit Überlappung

10a
10b
	Direkt dargestellte Elemente einer Struktur – teilweise festgelegt mit „ * “	`TYPE`
		`  HW_COMP: STRUCT;`
		`   IN  AT %I*: BOOL;`
		`   OUT_VAR AT %Q*: WORD := 200;`
		`   ITNL_VAR: REAL := 123.0; // Ohne Speicherort`
		`   END_STRUCT;`
		`END_TYPE`
			Weist die Komponenten einer Struktur Eingängen und Ausgängen zu, für die noch keine Speicherorte festgelegt sind.

11a
11b
	Direkt abgeleitete Datentypen	`TYPE`
		`  CNT:  UINT;`
		`  FREQ: REAL := 50.0;`
		`  ANALOG_CHANNEL_CONFIG:`
		`   ANALOG_CHANNEL_CONFIGURATION`
		`  := (MIN_SCALE := 0, MAX_SCALE := 4000);`
		`END_TYPE`
			Initialisierung
			
			Neue Initialisierung

12	Initialisierung unter Verwendung von konstanten Ausdrücken	`TYPE`
		`  PIx2: REAL := 2 * 3.1416;`
		`END_TYPE`
			Verwendet konstanten Ausdruck

13a
13b
	Datentyp Referenz (en: reference)	`TYPE`
		`  ref1 : REF_TO INT;`
		`  ref2 : REF_TO INT := NULL;`
		`  st : STRUCT`
		`    a : INT;`
		`    ref3 : REF_TO INT := REF(a);`
		`  END_STRUCT`
		`END_TYPE`
		
		`VAR_GLOBAL`
		`  g : ARRAY [0..7] OF INT;`
		`END_VAR`
		
		`PROGRAM A`
		`VAR_EXTERNAL`
		`  g : ARRAY [0..7] OF INT;`
		`END_VAR`
		`VAR`
		`  var1 : REF_TO INT := REF(g[3]);`
		`END_VAR`
		`;`
		`END_PROGRAM`
			Initialisierung
			
			Initialisierung
			
			Initialisierung

Die Datentypdeklaration ist ohne Initialisierung (Eigenschaft a) oder mit Initialisierung (Eigenschaft b) möglich. Wenn nur die Eigenschaft (a) unterstützt wird, wird der Datentyp mit dem voreingestellten Anfangswert initialisiert. Wenn die Eigenschaft (b) unterstützt wird, muss der Datentyp mit dem angegebenen Wert initialisiert werden oder mit dem voreingestellten Anfangswert, wenn kein Anfangswert angegeben ist.

Variablen mit direkt dargestellten Elementen einer Struktur, die teilweise mit „ * “ festgelegt sind, dürfen in den Sektionen `VAR_INPUT`, `VAR_IN_OUT` und `VAR_TEMP` nicht verwendet werden.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.4.4.2 Datentyp Aufzählung

6.4.4.2.1 Allgemeines

Die Deklaration eines Aufzählungs-Datentyps (en: enumeration) legt fest, dass der Wert jedes Datenelements dieses Typs nur eines der Elemente annehmen kann, die in der zugehörigen Liste der Bezeichner angegeben sind, wie in Tabelle 11 dargestellt.

Die Aufzählungsliste definiert eine geordnete Menge von aufgezählten Bezeichnern, beginnend mit dem ersten Bezeichner der Liste und endend mit dem letzten. Verschiedene Aufzählungs-Datentypen dürfen die gleichen Bezeichner verwenden. Die maximal zulässige Anzahl aufgezählter Bezeichner ist herstellerspezifisch. Einem aufgezählten Bezeichner darf kein Wert zugewiesen werden.

ANMERKUNG Der Anwender hat keine Möglichkeit, den Wert festzustellen, der einem aufgezählten Bezeichner intern zugewiesen wurde.

Ein Aufzählungs-Datentyp T steht in keiner Beziehung zu irgendeinem anderen Typ, mit Ausnahme von direkt abgeleiteten Datentypen des (transitiven) Basistyps T entsprechend der Definition in 6.4.4.9.

Um eine eindeutige Identifizierung in einem bestimmten Kontext zu ermöglichen, dürfen aufgezählte Literale durch ein Präfix beschrieben werden, das aus ihrem zugehörigen Datentypnamen und dem Zeichen `'#'` (Nummernzeichen) besteht, so wie typisierte Literale. Ein derartiges Präfix darf nicht innerhalb einer Aufzählungsliste verwendet werden.

Es ist ein Fehler, wenn in einem aufgezählten Literal nicht genügend Informationen zur Verfügung stehen, um dessen Wert eindeutig zu bestimmen (siehe Beispiel unten).

ANMERKUNG Ein Ausdruck des Typs der Aufzählung kann nur Werte der Aufzählung liefern.

BEISPIEL Datentyp Aufzählung.

```
TYPE
 Traffic_light:   (Red, Amber, Green);
 Painting_colors: (Red, Yellow, Green, Blue) := Blue;
 Painting_colors2: Painting_colors := Green;   // Painting_colors2 ist direkt
                                                  abgeleiteter
                                               // Datentyp und somit kompatibel mit
                                               // Painting_colors.
END_TYPE

VAR
 My_Traffic_light: Traffic_light := Amber;
END_VAR

IF My_Traffic_light = Traffic_light#Amber THEN ...  // OK
IF My_Traffic_light = Traffic_light#Red   THEN ...  // OK
IF My_Traffic_light = Amber THEN ...                // OK - Amber ist eindeutig
IF My_Traffic_light = Red   THEN ...                // FEHLER - Red ist nicht eindeutig
```

6.4.4.2.2 Initialisierung

Der voreingestellte Anfangswert eines Aufzählungs-Datentyps muss der erste Bezeichner in der zugehörigen Aufzählungsliste sein.

Der Anwender kann den Datentyp mit einem anwenderdefinierten Wert aus der Liste der aufgezählten Werte initialisieren. Diese Initialisierung hat Vorrang.

Wie in Tabelle 11 für `ANALOG_SIGNAL_RANGE`, gezeigt, ist der anwenderdefinierte voreingestellte Anfangswert des Aufzählungs-Datentyps der zugewiesene Wert `UNIPOLAR_1_5V`.

Die anwenderdefinierte Zuweisung des Anfangswerts des Datentyps ist eine Eigenschaft in Tabelle 11.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.4.4.3 Datentyp Benannte Werte

6.4.4.3.1 Allgemeines

Der Datentyp Benannte Werte ähnelt dem Datentyp Aufzählung, legt jedoch auch einen Basistyp und die Kodierung von dessen Werten fest, wie in Tabelle 11 dargestellt.

Die Definition eines Basistyps ist obligatorisch. Der Basistyp muss einer der Typen `ANY_INT` und `ANY_BIT` ohne `BOOL` sein. Anders als bei dem Datentyp Aufzählung kann ein Wert eines Benannte-Werte-Datentyps dort bereitgestellt werden, wo ein Wert seines Basistyps erwartet wird und umgekehrt.

Das Deklarieren benannter Werte begrenzt nicht die Verwendung des Wertebereichs der Variablen dieses Datentyps, d. h. es können weitere Konstante zugewiesen werden oder durch Berechnungen entstehen.

Anders als bei dem Datentyp Aufzählung müssen Operationen auf benannten Werten in gleicher Weise verfügbar sein wie Operationen auf dem Basistyp der benannten Werte.

Um eine eindeutige Identifizierung in einem bestimmten Kontext zu ermöglichen, dürfen benannte Werte durch ein Präfix beschrieben werden, das aus ihrem zugehörigen Datentypnamen und dem Zeichen `'#'` (Nummernzeichen) besteht, so wie typisierte Literale.

Ein derartiges Präfix darf nicht innerhalb einer Deklarationsliste verwendet werden. Es ist ein Fehler, wenn in einem aufgezählten Literal nicht genügend Informationen zur Verfügung stehen, um dessen Wert eindeutig zu bestimmen (siehe Beispiel unten).

BEISPIEL Definition und Verwendung des Datentyps Benannte Werte.

```
TYPE
 Traffic_light:   INT (Red := 1, Amber := 2, Green := 3):= Green;
 Painting_colors: INT (Red := 1, Yellow := 2, Green := 3, Blue := 4):= Blue;
  BaseColors: DWORD (Red := 16#00FF0000, Green := 16#0000FF00, Blue := 16#000000FF);
  Colors: DWORD (White := BaseColors#Red OR BaseColors#Green OR BaseColors#Blue,
                 Black := BaseColors#Red AND BaseColors#Green AND BaseColors#Blue);
END_TYPE

VAR
 My_Traffic_light: Traffic_light;
END_VAR

My_Traffic_light := 27;          // Zuweisung mit einer Konstanten
My_Traffic_light := Amber + 1;   // Zuweisung mit einem Ausdruck
                                 // Anmerkung: Dies ist für Aufzählungswerte nicht möglich
My_Traffic_light := Traffic_light#Red + 1;

IF My_Traffic_light   = 123 THEN ...                    // OK
IF My_Traffic_light   = Traffic_light#Amber THEN ...    // OK
IF My_Traffic_light   = Traffic_light#Red   THEN ...    // OK
IF My_Traffic_light   = Amber THEN ...                  // OK, weil Amber eindeutig ist
 IF  My_Painting_Color = Amber THEN ...         // OK, hat den gleichen Basisdatentyp
 IF My_Traffic_light   = Red THEN ...           // Fehler, weil Red nicht eindeutig ist
 My_Traffic_light := Painting_colors#Red;       // OK, hat den gleichen Basisdatentyp
 My_Traffic_light := Painting_colors#Blue-1;    // OK
 My_Traffic_light := My_Painting_colors;        // OK
```

6.4.4.3.2 Initialisierung

Der voreingestellte Anfangswert eines Benannte-Werte-Datentyps ist das erste Datenelement in der Aufzählungsliste. Im Beispiel oben für `Traffic_light` ist dieses Element `Red`.

Der Anwender kann den Datentyp mit einem anwenderdefinierten Wert initialisieren. Die Initialisierung ist nicht auf benannte Werte beschränkt; es darf jeder Wert innerhalb des Bereichs des Basisdatentyps verwendet werden. Diese Initialisierung hat Vorrang.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

In dem Beispiel ist der anwenderdefinierte Anfangswert des Benannte-Werte-Datentyps für `Traffic_light` `Green`.

Die anwenderdefinierte Zuweisung des Anfangswerts des Datentyps ist eine Eigenschaft in Tabelle 11.

6.4.4.4 Datentyp Unterbereich

6.4.4.4.1 Allgemeines

Ein Unterbereich ist ein Untertyp seines Basistyps, der auf einen anwenderdefinierten Wertebereich begrenzt ist.

Ein Ausdruck des Basistyps kann dort angegeben werden, wo ein Wert des Unterbereich-Typs erwartet wird. Es ist ein Fehler, wenn einem Datenelement eines Unterbereich-Typs ein Wert zugewiesen wird, der kein Wert des Unterbereich-Typs ist, der also nicht in dessen Bereich mit dem festgelegten oberen und unteren Grenzwert liegt.

ANMERKUNG Es ist kein Fehler, wenn ein vorübergehender Wert (z. B. während der Ausdruckauswertung) die Grenzen des Unterbereichs überschreitet, solange die Grenzen des zugehörigen Basistyps nicht überschritten werden. Die Grenzen eines Unterbereichs müssen Literale oder konstante Ausdrücke sein und müssen Werte des Basistyps sein.

BEISPIEL

```
TYPE
       ANALOG_DATA: INT (-4096..4095) := 0;
END_TYPE

PROGRAM prog
VAR
       i : INT;
       a : ANALOG_DATA;
       b AT %I* : ANALOG_DATA;
END_VAR
       [...]
       a := i; // Ist ein Fehler, wenn i den Bereich -4095..4096 überschreitet;
       a := (a*2)/3; // Kein Fehler. a*2 überschreitet ggf. den Unterbereich, aber der
                     // zugewiesene Wert nicht
       a := (a*20)/30; // Fehler, wenn a*20 Bereich -32768..32767 von INT überschreitet
       i := a;         // Kein Fehler

       // Extern bereitgestellte Werte von b dürfen den Bereich überschreiten, ohne
       // dass dies zu einem Fehler führt
       IF NOT LE(-4096,b,4095) THEN //LE(…): kleiner oder gleich
           (*Fehlerbehandlung*)
       END_IF;
END_PROGRAM
```

6.4.4.4.2 Initialisierung

Die voreingestellten Anfangswerte für Datentypen mit Unterbereich müssen den ersten (unteren) Grenzwert des Unterbereichs bilden.

Der Anwender kann den Datentyp mit einem anwenderdefinierten Wert aus dem Unterbereich initialisieren. Diese Initialisierung hat Vorrang.

Wie das Beispiel in Tabelle 11 zeigt, ist der voreingestellte Anfangswert von Elementen vom Typ `ANALOG_DATA` `-4095`, während der voreingestellte Anfangswert mit expliziter Initialisierung (wie deklariert) gleich null ist.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.4.4.5 Datentyp Feld

6.4.4.5.1 Allgemeines

Die Deklaration eines Feld-Datentyps legt den Datentyp von Feldelementen und eine geordnete Folge von Indexbereichen fest. Die Sequenz darf nicht leer sein. Jeder Bereich wird durch eine untere und eine obere Grenze definiert. Jeder Bereich entspricht einer Dimension des Felds.

Es muss als Fehler angesehen werden, wenn eine Dimension mit einem ganzzahligen Wert indiziert wird, der kleiner als die untere oder größer als die obere Grenze ist. Es muss als Fehler angesehen werden, wenn das Ergebnis der Subtraktion der unteren Grenze von der oberen Grenze kleiner als −1 ist. Die Unterstützung eines Ergebnisses von −1 (d. h. ein Feld ohne Elemente) ist herstellerspezifisch.

ANMERKUNG Das ARRAY[1..1] enthält ein Element, das ARRAY[1..0] enthält kein Element.

Zwei Feld-Datentypen, die sich in ihrem Element-Datentyp oder in der geordneten Folge von Indexbereichen unterscheiden, sind verschiedene Datentypen.

Ein Feld ist eine Sammlung von Datenelementen des gleichen Datentyps. Als Typ eines Feldelements können elementare und anwenderdefinierte Datentypen, Funktionsbaustein-Typen und Klassen verwendet werden. Diese Sammlung von Datentypen wird durch einen Index oder mehrere Indizes adressiert, die in eckige Klammern eingeschlossen und durch Kommas getrennt sind. Es muss als Fehler angesehen werden, wenn der Wert eines Index außerhalb des Bereichs liegt, der in der Feld-Deklaration festgelegt ist.

ANMERKUNG Dieser Fehler kann für berechnete Indizes nur während der Laufzeit entdeckt werden.

Die maximale Anzahl der Feldindizes, die maximale Feldgröße und der maximale Bereich der Indexwerte sind herstellerspezifisch.

Die Grenzen des (der) Indexunterbereich(e) müssen Literale oder konstante Ausdrücke sein. Felder mit variabler Länge sind in 6.5.3 definiert.

In der Sprache ST muss ein Index ein Ausdruck sein, der einen Wert liefert, der zu einem der Untertypen des allgemeinen Typs `ANY_INT` gehört.

Die Nennformen von Indizes in den in 8.1.2 definierten grafischen Sprachen sind Einzelelement-Variablen und ganzzahlige Literale.

BEISPIEL

a) Deklaration eines Feldes

```
VAR myANALOG_16: ARRAY [1..16] OF ANALOG_DATA
     := [8(-4095), 8(4095)];// Anwenderdefinierte Anfangswerte
END_VAR
```

b) Die Verwendung von Feldvariablen in der Sprache ST könnte sein:

```
OUTARY[6,SYM]:= INARY[0] + INARY[7] - INARY[i];
```

6.4.4.5.2 Initialisierung

Der voreingestellte Anfangswert jedes Feldelements ist der Anfangswert, der für den Datentyp der Feldelemente definiert wurde.

Der Anwender kann einen Feldtyp mit einem anwenderdefinierten Wert initialisieren. Diese Initialisierung hat Vorrang.

Der anwenderdefinierte Anfangswert eines Feldes wird in Form einer Liste zugewiesen, die runde Klammern verwenden darf, um Wiederholungen auszudrücken.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Bei der Initialisierung der Feld-Datentypen muss der ganz rechts stehende Index eines Feldes am schnellsten laufen, wenn das Feld mit der Liste der Initialisierungswerte aufgefüllt wird.

BEISPIEL Initialisierung eines Feldes.

```
A: ARRAY [0..5] OF INT := [2(1, 2, 3)]
```
entspricht der Initialisierungssequenz `1, 2, 3, 1, 2, 3`.

Wenn die Anzahl der Anfangswerte, die in der Initialisierungsliste angegeben ist, die Anzahl der Feldeinträge überschreitet, müssen die überschüssigen (ganz rechts stehenden) Anfangswerte ignoriert werden. Wenn die Anzahl der Anfangswerte kleiner als die Anzahl der Feldeinträge ist, müssen die restlichen Feldeinträge mit den voreingestellten Anfangswerten des entsprechenden Datentyps aufgefüllt werden. In beiden Fällen muss der Anwender auf diesen Umstand hingewiesen werden, wenn er das Programm für die Ausführung vorbereitet.

Die anwenderdefinierte Zuweisung des Anfangswerts des Datentyps ist eine Eigenschaft in Tabelle 11.

6.4.4.6 Strukturierter Datentyp

6.4.4.6.1 Allgemeines

Die Deklaration eines strukturierten Datentyps (`STRUCT`) legt fest, dass dieser Datentyp eine Sammlung von Unterelementen der festgelegten Typen beinhalten muss, auf die mit den festgelegten Namen zugegriffen werden kann, wie in Tabelle 11 dargestellt.

Ein Element eines strukturierten Datentyps muss durch zwei oder mehr Bezeichner oder Feldzugriffe, die durch einfache Punkte getrennt (**.**) sind, dargestellt werden. Der erste Bezeichner stellt den Namen des strukturierten Elements dar und die nachfolgenden Bezeichner stellen die Sequenz von Elementnamen für den Zugriff auf das bestimmte Datenelement innerhalb der Datenstruktur dar. Als Typ eines Strukturelements können elementare und anwenderdefinierte Datentypen, Funktionsbaustein-Typen und Klassen verwendet werden.

Zum Beispiel enthält ein Element vom Datentyp `ANALOG_CHANNEL_CONFIGURATION`, wie in Tabelle 11 deklariert, ein `RANGE`-Unterelement vom Typ `ANALOG_SIGNAL_RANGE`, ein `MIN_SCALE`-Unterelement vom Typ `ANALOG_DATA` und ein `MAX_SCALE`-Element vom Typ `ANALOG_DATA`.

Die maximale Anzahl von Strukturelementen, die maximale Menge von Daten, die in einer Struktur enthalten sein kann, und die maximale Anzahl von geschachtelten Ebenen der Strukturelementadressierung sind herstellerspezifisch.

Zwei strukturierte Variablen sind bei der Zuweisung nur dann kompatibel, wenn sie den gleichen Datentyp haben.

BEISPIEL Deklaration und Verwendung eines strukturierten Datentyps und einer strukturierten Variable.

a) Deklaration eines strukturierten Datentyps

```
TYPE
  ANALOG_SIGNAL_RANGE: (BIPOLAR_10V, UNIPOLAR_10V);
  ANALOG_DATA: INT (-4095 .. 4095);
  ANALOG_CHANNEL_CONFIGURATION:
    STRUCT
      RANGE:                ANALOG_SIGNAL_RANGE;
      MIN_SCALE, MAX_SCALE: ANALOG_DATA;
    END_STRUCT;
END_TYPE
```

b) Deklaration von strukturierten Variablen

```
VAR
  MODULE_CONFIG:  ANALOG_CHANNEL_CONFIGURATION;
  MODULE_8_CONF:  ARRAY [1..8] OF ANALOG_CHANNEL_CONFIGURATION;
END_VAR
```


Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

c) Verwendung von strukturierten Variablen in der Sprache ST:

```
MODULE_CONFIG.MIN_SCALE := -2047;
MODULE_8_CONF[5].RANGE := BIPOLAR_10V;
```

6.4.4.6.2 Initialisierung

Der voreingestellte Anfangswert eines strukturierten Datentyps setzt sich aus den Anfangswerten seiner Elemente zusammen.

Der voreingestellte Anfangswert eines solchen Elements ist der Anfangswert seines Typs. Es kann ein anwenderdefinierter Anfangswert für ein Element definiert werden, der dann Vorrang vor dem voreingestellten Anfangswert hat.

Es kann ein anwenderdefinierter Anfangswert für einen strukturierten Datentyp festgelegt werden. Der anwenderdefinierte Anfangswert wird angegeben, indem mit der in Anhang A definierten Syntax Werte für die Elemente des Datentyps festgelegt werden.

Es ist nicht notwendig, dass ein anwenderdefinierter Anfangswert für einen strukturierten Datentyp einen Wert für jedes Element dieser Struktur definiert.

Wenn für einen strukturierten Datentyp kein anwenderdefinierter Anfangswert festgelegt wird, so ist der voreingestellte Anfangswert der Anfangswert dieses Datentyps.

Wenn ein anwenderdefinierter Anfangswert für ein Element keinen Wert definiert, so wird der Wert für dieses Element im voreingestellten Anfangswert der Struktur als Anfangswert verwendet.

Diese Regeln gelten rekursiv, wenn der Typ eines Elements eines strukturierten Datentyps selbst ein strukturierter Datentyp ist.

Formal ausgedrückt: `IVD` sei der voreingestellte Anfangswert eines strukturierten Datentyps `S` und `IVU` sei der anwenderdefinierte Anfangswert von `S`. Wenn `C` eine Komponente von `S` ist, die in der Definition von `IVU` enthalten ist, so wird der Anfangswert von `C` definiert als der Wert von `C` in `IVU`. Andernfalls wird der Anfangswert von `C` durch `IVD` definiert.

ANMERKUNG Einfach ausgedrückt: Der Anfangswert eines strukturierten Datentyps ist der voreingestellte Anfangswert des Datentyps, sofern dieser nicht gezielt für eines oder alle seine Elemente überschrieben wird.

Wenn für einen strukturierten Datentyp `S` kein anwenderdefinierter Anfangswert `IVU` definiert ist, so wird der voreingestellte Anfangswert `IVD` für die Initialisierung verwendet. Wenn ein anwenderdefinierter Anfangswert `IVU` definiert ist, der alle Komponenten `C` von `S` abdeckt, so wird dieser Wert für die Initialisierung verwendet. Wenn ein anwenderdefinierter Anfangswert `IVU` definiert ist, der nur einen Teil der Komponenten `C` von `S` abdeckt, so wird er zur Initialisierung dieser Komponenten verwendet. Für die Initialisierung der verbleibenden Komponenten wird der voreingestellte Anfangswert `IVD` verwendet. In diesem Fall überschreibt der anwenderdefinierte Anfangswert `IVU` also nur einen Teil des voreingestellten Anfangswerts `IVD`.

BEISPIEL Definition einer Struktur und von Anfangswerten.

```
TYPE
  T : INT := 1;
  S : STRUCT
    I : T;        // Anfangswert ist 1
    J : T := 20;  // Anfangswert ist 20
    K : T := 30;  // Anfangswert ist 30
    L : T := 40;  // Anfangswert ist 40
  END_STRUCT;

  U :  S := (K := 300);   // Anfangswerte sind (I=1,J=20,K=300,L=40)

  V : U := (L := 4000);   // Anfangswerte sind (I=1,J=20,K=300,L=4000)
END_TYPE
```

Der Anfangswert `1` des Elements `I` ist dessen voreingestellter Anfangswert, nämlich der Anfangswert von `T`.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

Der Anfangswert `20` des Elements `J` ist dessen anwenderdefinierter Anfangswert.

Der Anfangswert `(I:=1, J:=20, K:=30, L:=40)` des strukturierten Typs `S` ist dessen voreingestellter Anfangswert, der sich aus den Anfangswerten seiner Elemente zusammensetzt.

Der Anfangswert `(I:=1, J:=20, K:=300, L:=40)` von `U` setzt sich aus seinem voreingestellten Anfangswert, nämlich dem Anfangswert von `S`, und dem anwenderdefinierten Anfangswert von `U`, nämlich `(K:=300)`, zusammen. Letzterer hat Vorrang.

Der Anfangswert `(I:=1, J:=20, K:=300, L:=4000)` von `V` setzt sich aus seinem voreingestellten Anfangswert, nämlich dem Anfangswert von `U`, und dem anwenderdefinierten Anfangswert von `V`, nämlich `(L:=4000)`, zusammen. Letzterer hat Vorrang.

6.4.4.7 Relativer Speicherort für Elemente von strukturierten Datentypen (AT)

6.4.4.7.1 Allgemeines

Die Speicherorte (Adressen) von Elementen eines strukturierten Datentyps können relativ zum Beginn der Struktur definiert sein.

In diesem Fall müssen auf den Namen jeder Komponente dieser Struktur das Schlüsselwort `AT` und ein relativer Speicherort (en: location) folgen. Die Deklaration darf Lücken in der Speicherbelegung enthalten.

Der relative Speicherort besteht aus einem Prozentzeichen (`%`), dem Bestimmungszeichen für den Speicherort und einem Bit- oder Byte-Speicherort. Der Byte-Speicherort ist ein vorzeichenloses ganzzahliges Literal, das den Byte-Offset angibt. Der Bit-Speicherort besteht aus einem Byte-Offset, gefolgt von einem „.“ (Punkt) und dem Bit-Offset als vorzeichenloses ganzzahliges Literal aus dem Bereich `0` bis `7`. Leerzeichen innerhalb des relativen Speicherorts sind nicht zulässig.

Die Komponenten der Struktur dürfen bei ihrer Speicherbelegung nicht überlappen, es sei denn, das Schlüsselwort `OVERLAP` ist in der Deklaration angegeben.

Das Überlappen von Folgen fällt nicht in den Anwendungsbereich dieser Norm.

ANMERKUNG Das Zählen von Bit-Offsets beginnt mit 0 beim niedrigstwertigen Bit. Das Zählen von Byte-Offsets beginnt am Anfang der Struktur mit dem Byte-Offset 0.

BEISPIEL Relativer Speicherort und Überlappen in einer Struktur.

```
TYPE
  Com1_data: STRUCT
   head     AT %B0:      INT;          // Am Speicherort 0
   length   AT %B2:      USINT := 26;  // Am Speicherort 2
   flag1    AT %X3.0:    BOOL;         // Am Speicherort 3,0
   end      AT %B25:     BYTE;         // Bei 25, mit Lücke
  END_STRUCT;

  Com2_data: STRUCT OVERLAP
   head     AT %B0:      INT;          // Am Speicherort 0
   length   AT %B2:      USINT;        // Am Speicherort 2
   flag2    AT %X3.3:    BOOL;         // Am Speicherort 3,3
   data1    AT %B5:      BYTE;         // Am Speicherort 5, überlappend
   data2    AT %B5:      REAL;         // An Speicherorten 5 bis 8
   end      AT %B19:     BYTE;         // Bei 19, mit Lücke
  END_STRUCT;

  Com_data: STRUCT OVERLAP   // C1 und C2 überlappen
   C1 at %B0: Com1_data;
   C2 at %B0: Com2_data;
 END_STRUCT;
END_TYPE
```


Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.4.4.7.2 Initialisierung

Überlappende Strukturen können nicht explizit initialisiert werden.

6.4.4.8 Direkt dargestellte Komponenten einer Struktur – partiell festgelegt mit „*“

Die Notation mit Sternchen (*) in Tabelle 11 kann verwendet werden, um noch nicht vollständig festgelegte Speicherorte für direkt dargestellte Komponenten einer Struktur zu kennzeichnen.

BEISPIEL Zuweisung der Komponenten einer Struktur an Eingänge und Ausgänge, für die noch keine Speicherorte festgelegt sind.

```
TYPE
  HW_COMP: STRUCT;
   IN         AT %I*: BOOL;
   VAL        AT %I*: DWORD;
   OUT        AT %Q*: BOOL;
   OUT_VAR    AT %Q*: WORD;
   ITNL_VAR: REAL; // Ohne Speicherort
   END_STRUCT;
END_TYPE
```

Falls eine direkt dargestellte Komponente einer Struktur in einer Speicherortzuweisung im Deklarationsteil eines Programms, eines Funktionsbaustein-Typs oder einer Klasse verwendet wird, muss ein Sternchen (*) anstelle des Präfixes für die Größe und der vorzeichenlose(n) ganze(n) Zahl(en) in der Verkettung verwendet werden, um anzuzeigen, dass die direkte Darstellung noch nicht vollständig festgelegt ist.

Die Verwendung dieser Eigenschaft erfordert, dass der Speicherort der auf diese Weise deklarierten strukturierten Variable vollständig im `VAR_CONFIG...END_VAR`-Konstrukt der Konfiguration für jede Instanz des einschließenden Typs festgelegt sein muss.

Variablen dieses Typs dürfen nicht in `VAR_INPUT`-, `VAR_IN_OUT`- oder `VAR_TEMP`-Sektionen verwendet werden.

Es ist ein Fehler, wenn im `VAR_CONFIG...END_VAR`-Konstrukt irgendeine vollständige Festlegung für eine unvollständige Adressfestlegung fehlt, die durch die Angabe des Sternchens (*) in einer Instanz von Programmen oder Funktionsbaustein-Typen mit solchen unvollständigen Festlegungen ausgedrückt wird.

6.4.4.9 Direkt abgeleiteter Datentyp

6.4.4.9.1 Allgemeines

Ein anwenderdefinierter Datentyp darf direkt von einem elementaren Datentyp oder einem vorher vom Anwender definierten Datentyp abgeleitet werden.

Der effektive Datentyp eines direkt abgeleiteten Datentyps T ist der Typ U, von dem er abgeleitet ist, außer wenn U selbst ein direkt abgeleiteter Datentyp ist. In diesem Fall ist der effektive Datentyp von T der effektive Datentyp von U.

Dies darf zur Festlegung von neuen typspezifischen Anfangswerten verwendet werden.

BEISPIEL Direkt abgeleiteter Datentyp.

```
TYPE
 myInt1123:     INT := 123;
 myNewArrayType: ANALOG_16_INPUT_DATA := [8(-1023), 8(1023)];
 Com3_data:   Com2_data:= (head := 3, length :=40);
 R1: REAL := 1.0;
 R2: R1; // Der effektive Datentyp von R2 ist REAL
END_TYPE
```


Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09

6.4.4.9.2 Initialisierung

Der voreingestellte Anfangswert eines direkt abgeleiteten Datentyps ist der Anfangswert des Datentyps, von dem er abgeleitet ist.

Es darf ein anwenderdefinierter Anfangswert als Anfangswert definiert werden, der dann Vorrang vor dem voreingestellten Anfangswert hat, entsprechend den Bestimmungen des effektiven Datentyps.

BEISPIEL 1 Anwenderdefinierte Datentypen – Verwendung.

Wenn die Deklaration von `ANALOG_16_INPUT_DATA` in Tabelle 11

und die Deklaration `VAR INS: ANALOG_16_INPUT_DATA; END_VAR` gegeben sind,

dann können die Variablen `INS[1]` bis `INS[16]` überall dort verwendet werden, wo eine Variable vom Typ `INT` verwendet werden kann.

BEISPIEL 2

Ebenso, wenn die Definition von `Com_data` in Tabelle 11

und zusätzlich die Deklaration `VAR telegram: Com_data; END_VAR` gegeben sind,

dann kann die Variable `telegram.length` überall dort verwendet werden, wo eine Variable vom Typ `USINT` verwendet werden kann.

BEISPIEL 3

Wenn die Deklarationen von `ANALOG_16_ INPUT_CONFIGURATION`, `ANALOG_CHANNEL_CONFIGURATION` und `ANALOG_DATA` in Tabelle 11 gegeben sind,

und die Deklaration `VAR CONF: ANALOG_16_INPUT_CONFIGURATION; END_VAR` gegeben sind,

dann kann die Variable `CONF.CHANNEL[2].MIN_SCALE` überall dort verwendet werden, wo eine Variable vom Typ `INT` verwendet werden kann.

6.4.4.10 Referenzen

6.4.4.10.1 Deklaration der Referenz

Ein Referenz-Datentyp wird mit dem Schlüsselwort `REF_TO` gefolgt von einer Typbeschreibung eines Basistyps definiert.

ANMERKUNG Der Basistyp darf ein elementarer oder ein anwenderdefinierter Datentyp sein.

Wenn der Basistyp ein direkt abgeleiteter Datentyp ist oder wenn seine Definition direkt abgeleitete Datentypen enthält, werden stattdessen ihre effektiven Typen (siehe 6.4.4.9) betrachtet.

BEISPIEL 1

```
TYPE
   A : INT;
   B : ARRAY [1..2] OF INT;
   C : ARRAY [1..2] OF A;
   D : C;
   E : REF_TO B;
   F : REF_TO D;  (* Typ F ist gleich Typ E *)
END_TYPE
```

Ein Wert eines Referenz-Datentyps ist entweder

– der Wert `NULL` oder
– eine Referenz auf eine Instanz des Basisdatentyps oder
– eine Referenz auf eine Instanz eines Typs, der den Basistyp erweitert, wenn der Basistyp eine Klasse oder ein Funktionsbaustein ist.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 61131-3:2023-09