

**NORME
INTERNATIONALE
INTERNATIONAL
STANDARD**

**CEI
IEC**

61131-3

Première édition
First edition
1993-03

Automates programmables –

**Partie 3:
Langages de programmation**

Programmable controllers –

**Part 3:
Programming languages**



Numéro de référence
Reference number
CEI/IEC 61131-3: 1993

Numéros des publications

Depuis le 1^{er} janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

Publications consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
Publié annuellement et mis à jour régulièrement
(Catalogue en ligne)*
- **Bulletin de la CEI**
Disponible à la fois au «site web» de la CEI* et comme périodique imprimé

Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique*, la CEI 60417: *Symboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles*, et la CEI 60617: *Symboles graphiques pour schémas*.

* Voir adresse «site web» sur la page de titre.

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
Published yearly with regular updates
(On-line catalogue)*
- **IEC Bulletin**
Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams*.

* See web site address on title page.

**NORME
INTERNATIONALE
INTERNATIONAL
STANDARD**

**CEI
IEC**

61131-3

Première édition
First edition
1993-03

Automates programmables –

**Partie 3:
Langages de programmation**

Programmable controllers –

**Part 3:
Programming languages**

© CEI 1993 Droits de reproduction réservés — Copyright - all rights reserved

Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'éditeur.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher

Bureau central de la Commission Electrotechnique Internationale 3, rue de Varembe Genève Suisse



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

SOMMAIRE

	Pages
AVANT-PROPOS	12
 Articles	
1 Généralités	14
1.1 Domaine d'application	14
1.2 Références normatives	14
1.3 Définitions	16
1.4 Résumé et prescriptions générales	26
1.4.1 Modèle logiciel	26
1.4.2 Modèle de communication	30
1.4.3 Modèle de programmation	34
1.5 Conformité	38
1.5.1 Systèmes d'automates programmables	40
1.5.2 Programmes	42
2 Eléments communs	44
2.1 Utilisation de caractères imprimés	44
2.1.1 Jeu de caractères	44
2.1.2 Identificateurs	46
2.1.3 Mots clés	48
2.1.4 Utilisation des espaces	48
2.1.5 Commentaires	48
2.2 Représentation externe des données	48
2.2.1 Libellés numériques	48
2.2.2 Libellés de cordons de caractères	50
2.2.3 Libellés de datation	52
2.3 Types de données	56
2.3.1 Types de données élémentaires	56
2.3.2 Types de données génériques	58
2.3.3 Types de données dérivés	60
2.4 Variables	68
2.4.1 Représentation	68
2.4.2 Initialisation	72
2.4.3 Déclaration	74
2.5 Unités d'organisation de programmes	84
2.5.1 Fonctions	84
2.5.2 Blocs fonctionnels	118
2.5.3 Programmes	148

CONTENTS

	Page
FOREWORD	13
 Clause	
1 General	15
1.1 Scope	15
1.2 Normative references	15
1.3 Definitions	17
1.4 Overview and general requirements	27
1.4.1 Software model	27
1.4.2 Communication model	31
1.4.3 Programming model	35
1.5 Compliance	39
1.5.1 Programmable controller systems	41
1.5.2 Programs	43
2 Common elements	45
2.1 Use of printed characters	45
2.1.1 Character set	45
2.1.2 Identifiers	47
2.1.3 Keywords	49
2.1.4 Use of spaces	49
2.1.5 Comments	49
2.2 External representation of data	49
2.2.1 Numeric literals	49
2.2.2 Character string literals	51
2.2.3 Time literals	53
2.3 Data types	57
2.3.1 Elementary data types	57
2.3.2 Generic data types	59
2.3.3 Derived data types	61
2.4 Variables	69
2.4.1 Representation	69
2.4.2 Initialization	73
2.4.3 Declaration	75
2.5 Program organization units	85
2.5.1 Functions	85
2.5.2 Function blocks	119
2.5.3 Programs	149

Articles	Pages
2.6 Eléments du diagramme fonctionnel en séquence (SFC)	150
2.6.1 Généralités	150
2.6.2 Etapes	150
2.6.3 Transitions	154
2.6.4 Actions	162
2.6.5 Règles d'évolution	180
2.6.6 Compatibilité des éléments de diagramme fonctionnel en séquence (SFC)	198
2.6.7 Prescriptions en matière de compatibilité	198
2.7 Eléments de configuration	200
2.7.1 Configurations, ressources et chemins d'accès	202
2.7.2 Tâches	206
3 Langages littéraux	226
3.1 Eléments Communs	226
3.2 Langage IL (liste d'instructions)	226
3.2.1 Instructions	226
3.2.2 Opérateurs, modificateurs et opérandes	228
3.2.3 Fonctions et blocs fonctionnels	230
3.3 Langage ST (littéral structuré)	232
3.3.1 Expressions	232
3.3.2 Enoncés	238
4 Langages graphiques	244
4.1 Eléments communs	244
4.1.1 Représentation des lignes et des blocs	244
4.1.2 Sens du flux dans les réseaux	246
4.1.3 Evaluation de réseaux	248
4.1.4 Eléments de commande d'exécution	252
4.2 Langage à contacts (LD)	256
4.2.1 Barres d'alimentation	256
4.2.2 Eléments de liaison et états	256
4.2.3 Contacts	258
4.2.4 Bobinages	258
4.2.5 Fonctions et blocs fonctionnels	258
4.2.6 Ordre d'évaluation des réseaux	258
4.3 Langage FBD (langage en blocs fonctionnels)	264
4.3.1 Généralités	264
4.3.2 Combinaison d'éléments	264
4.3.3 Ordre d'évaluation des réseaux	264
Annexes	
A Méthode de spécification pour les langages littéraux	266
A.1 Syntaxe	266
A.2 Sémantique	270

Clause	Page
2.6 Sequential Function Chart (SFC) elements	151
2.6.1 General	151
2.6.2 Steps	151
2.6.3 Transition	155
2.6.4 Actions	163
2.6.5 Rules of evolution	181
2.6.6 Compatibility of SFC elements	199
2.6.7 Compliance requirements	199
2.7 Configuration elements	201
2.7.1 Configurations, resources, and access paths	203
2.7.2 Tasks	207
3 Textual languages	227
3.1 Common elements	227
3.2 Language IL (Instruction List)	227
3.2.1 Instructions	227
3.2.2 Operators, modifiers and operands	229
3.2.3 Functions and function blocks	231
3.3 Language ST (Structured Text)	233
3.3.1 Expressions	233
3.3.2 Statements	239
4 Graphic languages	245
4.1 Common elements	245
4.1.1 Representation of lines and blocks	245
4.1.2 Direction of flow in networks	247
4.1.3 Evaluation of networks	249
4.1.4 Execution control elements	253
4.2 Language LD (Ladder Diagram)	257
4.2.1 Power rails	257
4.2.2 Link elements and states	257
4.2.3 Contacts	259
4.2.4 Coils	259
4.2.5 Functions and function blocks	259
4.2.6 Order of network evaluation	259
4.3 Language FBD (Function Block Diagram)	265
4.3.1 General	265
4.3.2 Combination of elements	265
4.3.3 Order of network evaluation	265
Annexes	
A Specification method for textual languages	267
A.1 Syntax	267
A.2 Semantics	271

Articles	Pages
B Spécifications formelles des éléments de langage	272
B.0 Modèle de programmation	272
B.1 Eléments communs	272
B.2 Langage IL (liste d'instructions)	292
B.3 Langage ST (langage littéral structuré)	294
C Délimiteurs et mots clés	298
D Paramètres dépendant de l'implémentation	304
E Situations d'erreur	308
F Exemples	310
F.1 Fonction WEIGH	310
F.2 Bloc fonctionnel CMD_MONITOR	312
F.3 Bloc fonctionnel FWD_REV_MON	318
F.4 Bloc fonctionnel STACK_INT	328
F.5 Bloc fonctionnel MIX_2_BRIX	338
F.6 Traitement de signaux analogiques	344
F.7 Programme GRAVEL	362
F.8 Programme AGV	378
G Index	386
H Essai de conformité logicielle	410
Tableaux	
1 Caractéristiques	46
2 Caractéristiques des identificateurs	46
3 La caractéristique commentaire	48
4 Libellés numériques	50
5 Caractéristiques des libellés de cordons de caractères	52
6 Combinaisons à deux chiffres dans les cordons de caractères	52
7 Caractéristiques de libellés de temps	54
8 Libellés de date et heure du jour	54
9 Exemples de libellés de date et heure du jour	56
10 Types de données élémentaires	58
11 Hiérarchie des types de données génériques	60
12 Caractéristiques des déclarations de types de données	64
13 Valeurs initiales par défaut	64
14 Caractéristiques de déclaration d'une valeur initiale de type de donnée	66
15 Caractéristiques des préfixes d'emplacement et de taille pour des variables représentées directement	70
16 Mots clés de déclarations de variables	76
17 Caractéristiques d'affectation de types de variables	78
18 Caractéristiques d'affectation de valeurs initiales de variables	82
19 Inversion graphique de signes booléens	86
20 Utilisation d'une entrée "EN" et d'une sortie "ENO"	90
21 Fonctions saisies et surchargées	94
22 Caractéristiques des fonctions de conversion de type	100
23 Fonctions standards à une seule variable numérique	102
24 Fonctions arithmétiques standards	104
25 Fonctions standards de décalage binaire	106

Clause	Page
B Formal specifications of language elements	273
B.0 Programming model	273
B.1 Common elements	273
B.2 Language IL (Instruction List)	293
B.3 Language ST (Structured Text)	295
C Delimiters and keywords	299
D Implementation-dependent parameters	305
E Error conditions	309
F Examples	311
F.1 Function WEIGH	311
F.2 Function block CMD_MONITOR	313
F.3 Function block FWD_REV_MON	319
F.4 Function block STACK_INT	329
F.5 Function block MIX_2_BRIX	339
F.6 Analog signal processing	345
F.7 Program GRAVEL	363
F.8 Program AGV	379
G Index	387
H Software compliance testing	411
Tables	
1 Character set features	47
2 Identifier features	47
3 Comment feature	49
4 Numeric literals	51
5 Character string literal feature	53
6 Two-character combinations in character strings	53
7 Duration literal features	55
8 Date and time of day literals	55
9 Examples of date and time of day literals	57
10 Elementary data types	59
11 Hierarchy of generic data types	61
12 Data type declaration features	65
13 Default initial values	65
14 Data type initial value declaration features	67
15 Location and size prefix features for directly represented variables	71
16 Variable declaration keywords	77
17 Variable type assignment features	79
18 Variable initial value assignment features	83
19 Graphical negation of Boolean signals	87
20 Use of EN input and ENO output	91
21 Typed and overloaded functions	95
22 Type conversion function features	101
23 Standard functions of one numeric variable	103
24 Standard arithmetic functions	105
25 Standard bit shift functions	107

Articles	Pages
26 Fonctions booléennes standard au niveau du bit	108
27 Fonctions standards de sélection	110
28 Fonctions standards de comparaison	112
29 Fonctions standards de cordons de caractères	114
30 Fonctions des types de données relatifs au temps	116
31 Fonctions de types de données énumérés	116
32 Exemples d'utilisation d'un paramètre d'E/S de bloc fonctionnel	120
33 Caractéristiques des déclarations de blocs fonctionnels	126
34 Blocs fonctionnels bistables standards	138
35 Blocs fonctionnels standards de détection de fronts	140
36 Blocs fonctionnels standards de compteurs	142
37 Blocs fonctionnels standards de temporisateurs	144
38 Bloc fonctionnels standards de temporisateurs – Schémas de temporisation	146
39 Caractéristiques de déclarations de programmes	148
40 Caractéristiques d'étage	154
41 Transitions et conditions de transition	158
42 Déclaration d'actions	164
43 Association action/étape	168
44 Caractéristiques de bloc d'action	170
45 Qualificatifs d'action	172
46 Evolution de séquence	184
47 Caractéristiques SFC compatibles	198
48 Prescriptions minimales relatives à la conformité des éléments de diagramme fonctionnel de séquence	198
49 Caractéristiques de déclaration de ressource et de configuration	204
50 Caractéristiques de tâche	212
51 Exemples de champs d'instruction	226
52 Opérateurs de liste d'instruction	230
53 Caractéristiques du lancement de bloc fonctionnel en langage IL	232
54 Opérateurs d'entrée standards des blocs fonctionnels en langage IL	232
55 Opérateurs du langage ST	236
56 Enoncés du langage ST	238
57 Représentation des lignes et des blocs	248
58 Eléments de commande d'exécution graphiques	254
59 Barres d'alimentation	256
60 Eléments de liaison	258
61 Contacts	260
62 Bobinages	262

Figures

1 Modèle logiciel	30
2 Modèle de communication	32
3 Combinaison d'éléments de langages pour automates programmables	38
4 Exemples d'utilisation de fonctions	84
5 Utilisation de noms de paramètres formels	88
6 Exemples de déclarations de fonctions	92
7 Exemples de conversion de type explicite, avec des fonctions surchargées	96
8 Exemples de conversion de type explicite, avec des fonctions saisies	98
9 Exemple d'instanciation de bloc fonctionnel	120
10 Exemples de déclarations de blocs fonctionnels	124
11 Utilisation graphique d'un nom de bloc fonctionnel en tant que variable	128

Clause	Page
26 Standard bitwise Boolean functions	109
27 Standard selection functions	111
28 Standard comparison functions	113
29 Standard character string functions	115
30 Functions of time data types	117
31 Functions of enumerated data type	117
32 Examples of function block I/O parameter usage	121
33 Function block declaration features	127
34 Standard bistable function blocks	139
35 Standard edge detection function blocks	141
36 Standard counter function blocks	143
37 Standard timer function blocks	145
38 Standard timer function blocks – timing diagrams	147
39 Program declaration features	149
40 Step features	155
41 Transitions and transition conditions	159
42 Declaration of actions	165
43 Step/action association	169
44 Action block features	171
45 Action qualifiers	173
46 Sequence evolution	185
47 Compatible SFC features	199
48 SFC minimal compliance requirements	199
49 Configuration and resource declaration features	205
50 Task features	213
51 Examples of instruction fields	227
52 Instruction List (IL) operators	231
53 Function block invocation features for IL language	233
54 Standard function block input operators for IL language	233
55 Operators of the ST language	237
56 ST language statements	239
57 Representation of lines and blocks	249
58 Graphic execution control elements	255
59 Power rails	257
60 Link elements	259
61 Contacts	261
62 Coils	263

Figures

1 Software model	31
2 Communication model	33
3 Combination of programmable controller language elements	39
4 Examples of function usage	85
5 Use of formal parameter names	89
6 Examples of function declarations	93
7 Examples of explicit type conversion with overloaded functions	97
8 Examples of explicit type conversion with typed functions	99
9 Function block instantiation example	121
10 Examples of function block declarations	125
11 Graphical use of function block names as variables	129

Articles	Pages
12 Exemples d'utilisation de variables entrée/sortie	134
13 Exemple d'utilisation de sémaphore	136
14 Bloc fonctionnel ACTION_CONTROL – Interface externe	174
15 Corps de bloc fonctionnel ACTION_CONTROL	176
16 Exemple de commande d'action	178
17 Règles d'évolution de diagramme fonctionnel de séquence (SFC)	192
18 Erreurs SFC	194
19 Exemple de configuration	200
20 Exemples de configurations de déclarations CONFIGURATION et RESSOURCE	206
21 Synchronisation de blocs fonctionnels	220
22 Exemple d'énoncé EXIT	242
23 Exemple de chemin d'asservissement	252
24 Exemples de OR booléen	264

Withdrawn

Clause	Page
12 Examples of use of input/output variables	135
13 Semaphore usage example	137
14 ACTION_CONTROL function block – External interface	175
15 ACTION_CONTROL function block body	177
16 Action control example	179
17 SFC evolution rules	193
18 SFC errors	195
19 Configuration example	201
20 Examples of CONFIGURATION and RESSOURCE declaration features	207
21 Synchronization of function blocks	221
22 EXIT statement example	243
23 Feedback path example	253
24 Boolean OR examples	265

Withdrawn

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

AUTOMATES PROGRAMMABLES –

Partie 3: Langages de programmation

AVANT-PROPOS

- 1) Les décisions ou accords officiels de la CEI en ce qui concerne les questions techniques, préparés par des Comités d'Etudes où sont représentés tous les Comités nationaux s'intéressant à ces questions, expriment dans la plus grande mesure possible un accord international sur les sujets examinés.
- 2) Ces décisions constituent des recommandations internationales et sont agréées comme telles par les Comités nationaux.
- 3) Dans le but d'encourager l'unification internationale, la CEI exprime le vœu que tous les Comités nationaux adoptent dans leurs règles nationales le texte de la recommandation de la CEI, dans la mesure où les conditions nationales le permettent. Toute divergence entre la recommandation de la CEI et la règle nationale correspondante doit, dans la mesure du possible, être indiquée en termes clairs dans cette dernière.

La présente partie de la Norme internationale CEI 1131 a été établie par le sous-comité 65B: Dispositifs, du comité d'études n° 65 de la CEI: Mesure et commande dans les processus industriels.

Le texte de cette partie est issu des documents suivants:

DIS	Rapport de vote
65B(BC)85	65B(BC)87

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette partie.

La CEI 1131 comprendra les parties suivantes, dont celle-ci est la troisième, présentées sous le titre général: Automates programmables.

Partie 1: 1992, Informations générales.

Partie 2: 1992, Spécifications et essais des équipements.

Partie 3: 1993, Langages de programmation.

Partie 4, Recommandations à l'utilisateur (à l'étude).

Partie 5, Spécification service de messagerie (à l'étude).

Les annexes A, B, C, D et E font partie intégrante de cette partie.

Les annexes F, G et H sont données uniquement à titre d'information.

Un rapporte technique (TR) de type 2 est en préparation. Ce TR constituera un guide de «pré-normalisation» pour la mise en oeuvre et l'application des langages de programmation définis dans la présente partie de la CEI 1131, y compris des développements tels que l'interaction système/programme et les prescriptions pour les conditions d'environnement du support de programmation.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PROGRAMMABLE CONTROLLERS –

Part 3: Programming languages

FOREWORD

- 1) The formal decisions or agreements of the IEC on technical matters, prepared by technical committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.
- 2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.
- 3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

This part of International Standard IEC 1131 has been prepared by sub-committee 65B: Devices, of IEC technical committee 65: Industrial process measurement and control.

The text of this standard is based on the following documents:

DIS	Report on Voting
65B(CO)85	65B(CO)87

Full information on the voting for the approval of this standard can be found in the Voting Report indicated in the above table.

IEC 1131 will consist of the following parts, of which this is the third under the general title: Programmable controllers.

Part 1: 1992, General information.

Part 2: 1992, Equipment requirements and tests.

Part 3: 1993, Programming languages.

Part 4, User guidelines (*under consideration*).

Part 5, Messaging service specification (*under consideration*).

Annexes A, B, C, D and E form an integral part of this part of IEC 1131.
Annexes F, G and H are for information only.

A type 2 technical report (TR) will provide "pre-standardization" guidance for the implementation and application of the programming language defined in this part of IEC 1131, including such issues as operating system/program interaction and requirements for programming support environments.

AUTOMATES PROGRAMMABLES –

Partie 3: Langages de programmation

1 Généralités

1.1 *Domaine d'application*

Cette partie de la CEI 1131 s'applique à la représentation imprimée et affichée, à l'aide des caractères du jeu de caractères ISO/IEC 646 des langages de programmation devant être utilisés pour des automates programmables tels que définis dans la partie 1 de la CEI 1131. La représentation graphique et semi-graphique des éléments de langage définis dans cette partie est admise, mais n'est pas définie dans cette partie.

Les fonctions de programme entrée, essai, contrôle, système d'exploitation, etc., sont spécifiées dans la partie 1 de la CEI 1131.

1.2 *Références normatives*

Les documents normatifs suivants contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente partie de la CEI 1131. Au moment de la publication, les éditions indiquées étaient en vigueur. Tout document normatif est sujet à révision et les parties prenantes aux accords fondés sur la présente partie de la CEI 1131 sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur.

CEI 50: *Vocabulaire Electrotechnique International (VEI)*

CEI 559: 1989, *Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs*

CEI 617-12: 1991, *Symboles graphiques pour schémas – Partie 12: Opérateurs logiques binaires*

CEI 617-13: 1978, *Symboles graphiques pour schémas – Partie 13: Opérateurs analogiques*

CEI 848: 1988, *Etablissement des diagrammes fonctionnels pour systèmes de commande*

ISO/AFNOR: 1989, *Dictionnaire de l'informatique, ISBN 2-12-486911-6*

ISO/CEI 646: 1991, *Technologie de l'information – Jeu ISO de caractères codés à 7 éléments pour l'échange d'informations (publié actuellement en anglais seulement)*

ISO 8601: 1988, *Eléments de données et formats d'échange – Echange d'information – Représentation de la date et de l'heure*

PROGRAMMABLE CONTROLLERS –

Part 3: Programming languages

1 General

1.1 Scope

This part of IEC 1131 applies to the printed and displayed representation, using characters of the ISO/IEC 646 character set, of the programming languages to be used for Programmable Controllers as defined in Part 1 of IEC 1131. Graphic and semigraphic representation of the language elements which are defined in this part is allowed, but is not defined in this part.

The functions of program entry, testing, monitoring, operating system, etc., are specified in Part 1 of IEC 1131.

1.2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 1131. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this part of IEC 1131 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 50: *International Electrotechnical Vocabulary (IEV)*

IEC 559: 1989, *Binary floating-point arithmetic for microprocessor systems*

IEC 617-12: 1991, *Graphical symbols for diagrams – Part 12: Binary logic elements*

IEC 617-13: 1978, *Graphical symbols for diagrams – Part 13: Analogue elements*

IEC 848: 1988, *Preparation of function charts for control systems*

ISO/AFNOR: 1989, *Dictionary of computer science, ISBN 2-12-486911-6*

ISO/IEC 646: 1991, *Information technology – ISO 7-bit coded character set for information processing interchange*

ISO 8601: 1988, *Data elements and interchange formats – Information interchange – Representations of dates and times*

ISO 7185: 1990, *Technologies de l'information – Langages de programmation – Pascal* (publié actuellement en anglais seulement)

ISO 7498: 1984, *Systèmes de traitement de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base*

1.3 Définitions

Pour les besoins de la présente partie de la CEI 1131, les définitions suivantes s'appliquent. Les définitions applicables à toutes les parties de la CEI 1131 sont données dans la partie 1.

NOTES

- 1 Les termes définis dans ce paragraphe sont en italiques lorsqu'ils figurent dans les corps des définitions.
- 2 Les définitions suivies de l'indication «(ISO)» sont tirées du *Dictionnaire de l'informatique ISO/AFNOR*.
- 3 En ce qui concerne les termes qui n'ont pas été définis dans la présente norme, il convient de consulter le *Dictionnaire de l'informatique ISO/AFNOR* et le *Vocabulaire Électrotechnique International*.

1.3.1 **temps absolu**: La combinaison des informations fournies par l'heure du jour et la date.

1.3.2 **chemin d'accès**: L'association d'un nom symbolique avec une variable pour les besoins d'une communication ouverte.

1.3.3 **action**: Une variable booléenne, ou ensemble d'opérations à effectuer, en même temps qu'une structure de commande associée, telle que spécifiée en 2.6.4.

1.3.4 **bloc d'action**: Un élément de langage graphique qui utilise une variable booléenne d'entrée pour déterminer la valeur d'une variable booléenne de sortie ou la condition de validation relative à une action, conformément à une structure de commande prédéterminée, telle que définie en 4.1.3.

1.3.5 **agrégat**: Un ensemble structuré d'objets relatifs aux données défini comme un seul *type de données*. (ISO)

1.3.6 **argument**: Synonyme de *paramètre d'entrée* ou de *paramètre de sortie*.

1.3.7 **tableau**: Un *agrégat* dont chaque constituant possède des attributs identiques, et peut être désigné sans ambiguïté par indiciage.

1.3.8 **affectation**: Un procédé permettant d'attribuer une valeur à une variable ou à un *agrégat*. (ISO)

1.3.9 **nombre basé**: Un nombre représenté dans une base spécifiée autre que la base dix.

1.3.10 **bloc fonctionnel bistable**: Un *bloc fonctionnel* disposant de deux états stables, commandé par une ou plusieurs entrées.

1.3.11 **cordon de bits**: Un élément de données composé d'un ou de plusieurs bits.

ISO 7185: 1990, *Information technology – Programming languages – Pascal*

ISO 7498: 1984, *Information processing systems – Open systems interconnection – Basic reference model*

1.3 Definitions

For the purposes of this part of IEC 1131, the following definitions apply. Definitions applying to all parts of IEC 1131 are given in part 1.

NOTES

- 1 Terms defined in this subclause are italicized where they appear in the bodies of definitions.
- 2 The notation "(ISO)" following a definition indicates that the definition is taken from the ISO/AFNOR *Dictionary of computer science*.
- 3 The ISO/AFNOR *Dictionary of computer science* and the *International Electrotechnical Vocabulary* should be consulted for terms not defined in this standard.

1.3.1 **absolute time**: The combination of time of day and date information.

1.3.2 **access path**: The association of a symbolic name with a variable for the purpose of open communication.

1.3.3 **action**: A Boolean variable, or a collection of operations to be performed, together with an associated control structure, as specified in 2.6.4.

1.3.4 **action block**: A graphical language element which utilizes a Boolean input variable to determine the value of a Boolean output variable or the enabling condition for an *action*, according to a predetermined control structure as defined in 2.6.4.5.

1.3.5 **aggregate**: A structured collection of data objects, forming a *data type*. (ISO)

1.3.6 **argument**: Synonymous with *input parameter* or *output parameter*.

1.3.7 **array**: An *aggregate* that consists of data objects, with identical attributes, each of which may be uniquely referenced by *subscripting*. (ISO)

1.3.8 **assignment**: A mechanism to give a value to a variable or to an *aggregate*. (ISO)

1.3.9 **based number**: A number represented in a specified base other than ten.

1.3.10 **bistable function block**: A *function block* with two stable states controlled by one or more inputs.

1.3.11 **bit string**: A data element consisting of one or more bits.

1.3.12 corps: Cette partie d'une *unité d'organisation de programme* qui spécifie les opérations à effectuer sur les *opérandes* déclarés de l'unité d'organisation de programme, lorsque son exécution est *lancée*.

1.3.13 appel: Un élément de langage permettant de déclencher l'exécution d'une *fonction* ou d'un *bloc fonctionnel*.

1.3.14 cordon de caractères: Un *agrégat* composé d'une suite ordonnée de caractères.

1.3.15 commentaire: Un élément de langage, permettant d'insérer dans un programme des textes quelconques sans incidence sur l'exécution de ce programme. (ISO)

1.3.16 compilation: Processus consistant à traduire la spécification d'une *unité d'organisation de programme* ou d'un *type de donnée* en son équivalent en langage machine ou en une autre forme intermédiaire.

1.3.17 configuration: Un élément de langage correspondant à un *système d'automate programmable* tel que défini dans la CEI 1131-1.

1.3.18 bloc fonctionnel compteur: Un *bloc fonctionnel* qui accumule une valeur pour le nombre de changements détectés sur une ou plusieurs *entrées* spécifiées.

1.3.19 type de données: Un ensemble de valeurs associées à l'ensemble des opérations permises sur ces valeurs. (ISO)

1.3.20 date et heure: La date de l'année en cours et l'heure du jour, dont la représentation est effectuée conformément aux règles fixées par ISO 8601.

1.3.21 déclaration: Le mécanisme que permet d'établir la définition d'un *élément de langage*. Normalement, une déclaration nécessite le rattachement d'un identificateur à l'élément de langage et l'affectation d'attributs, tels que des *types de données* et des algorithmes, à l'objet de langage concerné.

1.3.22 délimiteur: Un caractère ou une combinaison de caractères servant à séparer les *éléments de langage* du programme.

1.3.23 représentation directe: Un moyen de représenter une variable dans un système d'automate programmable à partir de laquelle une correspondance définie par le fabricant peut être déterminée directement avec un emplacement logique ou physique.

1.3.24 double mot: Un élément de donnée composé de 32 bits.

1.3.25 évaluation: Le processus qui consiste à déterminer une valeur pour une expression ou pour une *fonction*, ou pour les *sorties* d'un réseau ou d'un *bloc fonctionnel*, au cours de l'exécution d'un programme.

1.3.26 élément de commande d'exécution: Un *élément de langage* qui commande le flux de l'exécution d'un programme.

- 1.3.12 body:** That portion of a *program organization unit* which specifies the operations to be performed on the declared *operands* of the program organization unit when its execution is *invoked*.
- 1.3.13 call:** A language construct for *invoking* the execution of a *function* or *function block*.
- 1.3.14 character string:** An *aggregate* that consists of an ordered sequence of characters.
- 1.3.15 comment:** A language construct for the inclusion of text in a program and having no impact on the execution of the program. (ISO)
- 1.3.16 compile:** To translate a *program organization unit* or *data type* specification into its machine language equivalent or an intermediate form.
- 1.3.17 configuration:** A language element corresponding to a *programmable controller system* as defined in IEC 1131-1.
- 1.3.18 counter function block:** A *function block* which accumulates a value for the number of changes sensed at one or more specified *inputs*.
- 1.3.19 data type:** A set of values together with a set of permitted operations. (ISO)
- 1.3.20 date and time:** The date within the year and the time of day, represented according to ISO 8601.
- 1.3.21 declaration:** The mechanism for establishing the definition of a *language element*. A declaration normally involves attaching an identifier to the language element, and allocating attributes such as *data types* and algorithms to it.
- 1.3.22 delimiter:** A character or combination of characters used to separate program *language elements*.
- 1.3.23 direct representation:** A means of representing a variable in a programmable controller program from which a manufacturer-specified correspondence to a physical or *logical location* may be determined directly.
- 1.3.24 double word:** A data element containing 32 bits.
- 1.3.25 evaluation:** The process of establishing a value for an expression or a *function*, or for the *outputs* of a network or *function block*, during program execution.
- 1.3.26 execution control element:** A *language element* which controls the flow of program execution.

1.3.27 front descendant: Le passage d'une variable booléenne de la valeur 1 à la valeur 0.

1.3.28 fonction: Une *unité d'organisation de programme* qui, lorsqu'elle est exécutée, fournit exactement un élément de donnée (qui peut avoir plusieurs valeurs, par exemple un *tableau* ou une *structure*), et dont l'*appel* peut servir, dans les langages littéraux, comme *opérande* dans une expression.

1.3.29 instance de bloc fonctionnel (bloc fonctionnel): Une *instance* d'un *type de bloc fonctionnel*.

1.3.30 type de bloc fonctionnel: Un *élément de langage* de programmation d'automate programmable qui consiste en: (i) la définition d'une structure de données répartie entre une entrée, une sortie et des variables internes; et (ii) un ensemble d'opérations à effectuer sur les éléments de la structure de données lorsqu'une *instance* du type de bloc fonctionnel est *appelée*.

1.3.31 schéma en blocs fonctionnels: Un ou plusieurs réseaux de *fonctions*, de *blocs fonctionnels*, d'éléments de données, d'*étiquettes* et d'éléments de branchement, représentés sous forme graphique.

1.3.32 type de donnée générique: Un *type de donnée* représentant plusieurs types de données, comme cela est spécifié en 2.3.2.

1.3.33 champ global: Champ d'application d'une déclaration qui a une incidence sur toutes les unités d'organisation de programmes, dans une *ressource* ou dans une *configuration*.

1.3.34 variable globale: Une variable dont le *champ* est *global*.

1.3.35 adressage hiérarchique: La *représentation directe* d'un élément de donnée comme faisant partie d'une hiérarchie physique ou logique, par exemple un point à l'intérieur d'un module situé dans un panier, lui-même logé dans une armoire, etc.

1.3.36 identificateur: Une combinaison de lettres, de chiffres et de caractères de soulignement tels que spécifiés en 2.1.2, qui doit nécessairement commencer par une lettre ou par un caractère de soulignement, et qui désigne un *élément de langage*.

1.3.37 valeur initiale: La valeur affectée à une variable, au moment du démarrage du système.

1.3.38 paramètre d'entrée (entrée): Un paramètre qui sert à fournir un argument à une *unité d'organisation de programme*.

1.3.39 instance: Une copie individuelle désignée de la structure de donnée, associée à un *type de bloc fonctionnel* ou à un *type de programme*, et qui persiste entre un *déclenchement* des opérations associées et le déclenchement suivant.

1.3.40 nom d'instance: Un *identificateur* associé à une instance spécifique.

1.3.41 instanciation: La création d'une *instance*.

1.3.27 falling edge: The change from 1 to 0 of a Boolean variable.

1.3.28 function: A *program organization unit* which, when executed, yields exactly one data element (which may be multi-valued, e.g., an *array* or *structure*), and whose *invocation* can be used in textual languages as an *operand* in an expression.

1.3.29 function block instance (function block): An *instance* of a *function block type*.

1.3.30 function block type: A programmable controller programming *language element* consisting of: (i) the definition of a data structure partitioned into input, output, and internal variables; and (ii) a set of operations to be performed upon the elements of the data structure when an *instance* of the function block type is *invoked*.

1.3.31 function block diagram: One or more networks of graphically represented *functions*, *function blocks*, data elements, *labels*, and connective elements.

1.3.32 generic data type: A *data type* which represents more than one type of data, as specified in 2.3.2.

1.3.33 global scope: Scope of a declaration applying to all program organization units within a *resource* or *configuration*.

1.3.34 global variable: A variable whose *scope* is *global*.

1.3.35 hierarchical addressing: The *direct representation* of a data element as a member of a physical or logical hierarchy, e.g., a point within a module which is contained in a rack, which in turn is contained in a cubicle, etc.

1.3.36 identifier: A combination of letters, numbers, and underline characters, as specified in 2.1.2, which begins with a letter or underline and which names a *language element*.

1.3.37 initial value: The value assigned to a variable at system start-up.

1.3.38 input parameter (input): A parameter which is used to supply an argument to a *program organization unit*.

1.3.39 instance: An individual, named copy of the data structure associated with a *function block type* or *program type*, which persists from one *invocation* of the associated operations to the next.

1.3.40 instance name: An *identifier* associated with a specific *instance*.

1.3.41 instantiation: The creation of an *instance*.

- 1.3.42 **libellé entier**: Un *libellé* qui représente directement une valeur de type SINT, INT, DINT, LINT, BOOL, BYTE, WORD, DWORD, ou LWORD, telle que définie en 2.3.1.
- 1.3.43 **déclenchement**: Le processus de lancement de l'exécution des opérations spécifiées dans une *unité d'organisation de programme*.
- 1.3.44 **mot clé**: Une unité lexicale qui caractérise un *élément de langage*, par exemple "IF".
- 1.3.45 **étiquette**: Une construction de langage désignant une instruction, un réseau, ou un groupe de réseaux, et comprenant un identificateur.
- 1.3.46 **élément de langage**: Tout élément identifié par un symbole placé à gauche d'une règle de production dans la spécification formelle donnée à l'annexe B de la présente partie de la CEI 1131.
- 1.3.47 **libellé**: Une unité lexicale qui représente explicitement une valeur. (ISO)
- 1.3.48 **champ local**: Le *champ* d'application d'une *déclaration* ou d'une *étiquette* qui ne s'applique qu'à l'*unité d'organisation de programme* dans laquelle la déclaration ou l'étiquette apparaît.
- 1.3.49 **emplacement logique**: L'emplacement d'une variable *adressée de manière hiérarchisée* dans un schéma qui peut ou non intégrer une relation à la structure physique des entrées, des sorties et de la mémoire de automates programmables.
- 1.3.50 **réel long**: Un nombre réel représenté dans un *mot long*.
- 1.3.51 **mot long**: Un élément de donnée composé de 64 bits.
- 1.3.52 **mémoire (stockage des données utilisateur)**: Une unité fonctionnelle dans laquelle le programme utilisateur peut mémoriser des données et d'où il peut récupérer les données mémorisées.
- 1.3.53 **élément nommé**: Un élément d'une *structure*, qui est désigné par l'*identificateur* qui lui est associé.
- 1.3.54 **bloc fonctionnel temporisateur de déclenchement/enclechement**: Un *bloc fonctionnel* qui retarde, d'une durée spécifiée, le *front descendant (montant)* d'une *entrée* booléenne.
- 1.3.55 **opérande**: Un *élément de langage* sur lequel une opération est effectuée.
- 1.3.56 **opérateur**: Un symbole représentant l'action à exécuter dans une opération.
- 1.3.57 **paramètre de sortie (sortie)**: Un *paramètre* servant à renvoyer le(s) résultat(s) de l'évaluation d'une *unité d'organisation de programme*.
- 1.3.58 **surchargée**: Se dit d'une opération ou d'une *fonction* capable d'intervenir sur des données de types différents, comme spécifié en 2.5.1.4.

1.3.42 Integer literal: A *literal* which directly represents a value of type SINT, INT, DINT, LINT, BOOL, BYTE, WORD, DWORD, or LWORD, as defined in 2.3.1.

1.3.43 invocation: The process of initiating the execution of the operations specified in a *program organization unit*.

1.3.44 keyword: A lexical unit that characterizes a *language element*, e.g., "IF".

1.3.45 label: A language construction naming an instruction, network, or group of networks, and including an *identifier*.

1.3.46 language element: Any item identified by a symbol on the left-hand side of a production rule in the formal specification given in annex B of this part of IEC 1131.

1.3.47 literal: A lexical unit that directly represents a value. (ISO)

1.3.48 local scope: The *scope* of a *declaration* or *label* applying only to the *program organization unit* in which the declaration or label appears.

1.3.49 logical location: The location of a *hierarchically addressed* variable in a schema which may or may not bear any relation to the physical structure of the programmable controller's inputs, outputs, and memory.

1.3.50 long real: A real number represented in a *long word*.

1.3.51 long word: A 64-bit data element.

1.3.52 memory (user data storage): A functional unit to which the user program can store data and from which it can retrieve the stored data.

1.3.53 named element: An element of a *structure* which is named by its associated *identifier*.

1.3.54 off-delay (on-delay) timer function block: A *function block* which delays the *falling (rising) edge* of a Boolean *input* by a specified duration.

1.3.55 operand: A *language element* on which an operation is performed.

1.3.56 operator: A symbol that represents the action to be performed in an operation.

1.3.57 output parameter (output): A *parameter* which is used to return the result(s) of the *evaluation* of a *program organization unit*.

1.3.58 overloaded: With respect to an operation or *function*, capable of operating on data of different types, as specified in 2.5.1.4.

1.3.59 sens du courant: Le flux symbolique de l'énergie électrique dans un schéma à contacts (LD), utilisé pour indiquer la progression d'un algorithme logique de résolution.

1.3.60 programmer: Pour concevoir, écrire et tester des programmes utilisateur.

1.3.61 unité d'organisation de programme: Une *fonction*, *bloc fonctionnel*, ou programme.

NOTE - Ce terme peut se rapporter soit un *type*, soit à une *instance*.

1.3.62 libellé réel: Un *libellé* représentant des données de type REAL ou LREAL.

1.3.63 ressource: Un *élément de langage* correspondant à une "fonction de traitement de signal", ainsi qu'à son "interface homme-machine" et à ses éventuelles "fonctions d'interface de capteurs et d'actionneurs", telles que définies dans la CEI 1131-1.

1.3.64 données non volatiles: Des données mémorisées de telle manière que leur valeur reste invariable après une mise hors tension puis une remise sous tension.

1.3.65 retour à la ligne: Un élément de langage dans une *unité d'organisation de programme* qui désigne la fin d'une séquence d'exécution dans cette unité.

1.3.66 front montant: Le passage d'une variable booléenne de la valeur 0 à la valeur 1.

1.3.67 champ d'application: La partie d'un *élément de langage* à l'intérieur de laquelle s'applique une *déclaration* ou une *étiquette*.

1.3.68 sémantique: Les relations existant entre les éléments symboliques d'un langage de programmation et leur signification, interprétation et utilisation.

1.3.69 représentation semigraphique: Représentation d'informations graphiques par l'intermédiaire d'un ensemble limité de caractères.

1.3.70 élément de donnée unitaire: Un élément de donnée qui ne comporte qu'une seule valeur.

1.3.71 étape: Une situation dans laquelle le comportement d'une *unité d'organisation de programme* vis-à-vis de ses *entrées* et de ses *sorties* suit un ensemble de règles définies par les *actions* associées de l'étape.

1.3.72 type de donnée structurée: Un type de donnée *agrégat* qui a été déclaré à l'aide d'une déclaration de STRUCT ou de FUNCTION_BLOCK.

1.3.73 indiciation: Un mécanisme permettant de référencer un élément de *tableau* à l'aide d'une référence de tableau et d'une ou de plusieurs expressions qui, lorsqu'elles sont évaluées, indiquent la position de cet élément.

1.3.74 représentation symbolique: L'utilisation d'*identificateurs* pour nommer des variables.

1.3.75 tâche: Un *élément de commande d'exécution* qui permet l'exécution périodique, ou déclenchée, d'un group d'*unités d'organisations de programmes* associées.

1.3.59 power flow: The symbolic flow of electrical power in a ladder diagram, used to denote the progression of a logic solving algorithm.

1.3.60 program (verb): To design, write, and test user programs.

1.3.61 program organization unit: A *function*, *function block*, or *program*.

NOTE - This term may refer to either a *type* or an *instance*.

1.3.62 real literal: A *literal* representing data of type REAL or LREAL.

1.3.63 resource: A *language element* corresponding to a "signal processing function" and its "man-machine interface" and "sensor and actuator interface functions", if any, as defined in IEC 1131-1.

1.3.64 retentive data: Data stored in such a way that its value remains unchanged after a power down / power up sequence.

1.3.65 return: A language construction within a *program organization unit* designating an end to the execution sequences in the unit.

1.3.66 rising edge: The change from 0 to 1 of a Boolean variable.

1.3.67 scope: That portion of a *language element* within which a *declaration* or *label* applies.

1.3.68 semantics: The relationships between the symbolic elements of a programming language and their meanings, interpretation and use.

1.3.69 semigraphic representation: Representation of graphic information by the use of a limited set of characters.

1.3.70 single data element: A data element consisting of a single value.

1.3.71 step: A situation in which the behavior of a *program organization unit* with respect to its *inputs* and *outputs* follows a set of rules defined by the associated *actions* of the step.

1.3.72 structured data type: An *aggregate* data type which has been declared using a STRUCT or FUNCTION_BLOCK declaration.

1.3.73 subscripting: A mechanism for referencing an *array* element by means of an array reference and one or more expressions that, when evaluated, denote the position of the element.

1.3.74 symbolic representation: The use of *identifiers* to name variables.

1.3.75 task: An *execution control element* providing for periodic or triggered execution of a group of associated *program organization units*.

1.3.76 libellé de datation: Un *libellé* représentant des données de type TIME, DATE, TIME_OF_DAY, ou DATE_AND_TIME.

1.3.77 transition: La condition par laquelle une commande passe d'une ou de plusieurs *étapes* antérieures à une ou plusieurs *étapes* ultérieures le long d'une liaison dirigée.

1.3.78 entier non signé: Un *libellé entier* ne comportant, ni signe plus (+), ni signe moins (-).

1.3.79 OR câblé: Une construction matérielle permettant de réaliser la fonction booléenne OR dans le langage à contacts (LD), qui s'effectue en reliant les extrémités droites de liaison horizontales à des liaisons verticales.

1.4 Résumé et prescriptions générales

La présente partie de la CEI 1131 spécifie la syntaxe et la sémantique d'une série unifiée de langages de programmation pour automates programmables (AP). Cette série se compose de deux langages littéraux: le langage IL (liste d'instructions) et le langage ST (littéral structuré), et de deux langages graphiques: le langage LD (langage à contacts) et le langage FBD (diagramme fonctionnel).

Cette partie définit des éléments de schémas "diagramme fonctionnel en séquence" (SFC) destinés à structurer l'organisation interne des *programmes* d'automates programmables et de *blocs fonctionnels*. En outre, elle définit des éléments de configuration destinés à aider à l'installation de *programmes* d'automates programmables dans des systèmes d'automates programmables.

En outre, cette partie définit des caractéristiques destinées à faciliter les échanges d'informations entre des automates programmables et d'autres composants de systèmes automatisés.

Il est possible d'utiliser les éléments de langages de programmation définis dans cette partie dans un environnement de programmation interactive. La spécification de tels environnements n'est pas couverte par le domaine d'application de cette partie; cependant, un tel environnement doit être capable de produire une documentation relative à des programmes littéraux ou graphiques, dans les formats spécifiés dans cette partie.

Dans cette partie, les renseignements sont présentés "de bas en haut", c'est-à-dire que les éléments de langage les plus simples sont présentés en premier, afin de minimiser les renvois dans le texte. La suite du présent paragraphe donne un aperçu général des renseignements présentés dans cette partie et comporte quelques prescriptions générales.

1.4.1 Modèle logiciel

La figure 1 illustre les éléments fondamentaux d'un langage évolué, ainsi que leurs relations réciproques. Ceux-ci se composent d'éléments qui ont été *programmés* à l'aide des langages définis dans la présente partie, c'est-à-dire des *programmes* et des *blocs fonctionnels*; et d'*éléments de configuration*, à savoir: des *configurations*, des *ressources*, des *tâches*, des *variables globales* et des *chemins d'accès*, qui sont destinés à aider à l'installation de *programmes* d'automates programmables dans des systèmes d'automates programmables.

1.3.76 time literal: A *literal* representing data of type TIME, DATE, TIME_OF_DAY, or DATE_AND_TIME.

1.3.77 transition: The condition whereby control passes from one or more predecessor *steps* to one or more successor steps along a directed link.

1.3.78 unsigned integer: An *integer literal* not containing a leading plus (+) or minus (-) sign.

1.3.79 wired OR: A construction for achieving the Boolean OR function in the LD language by connecting together the right ends of horizontal connectives with vertical connectives.

1.4 Overview and general requirements

This part of IEC 1131 specifies the syntax and semantics of a unified suite of programming languages for programmable controllers (PCs). These consist of two textual languages, IL (Instruction List) and ST (Structured Text), and two graphical languages, LD (Ladder Diagram) and FBD (Function Block Diagram).

Sequential Function Chart (SFC) elements are defined for structuring the internal organization of programmable controller *programs* and *function blocks*. Also, *configuration elements* are defined which support the installation of programmable controller *programs* into programmable controller systems.

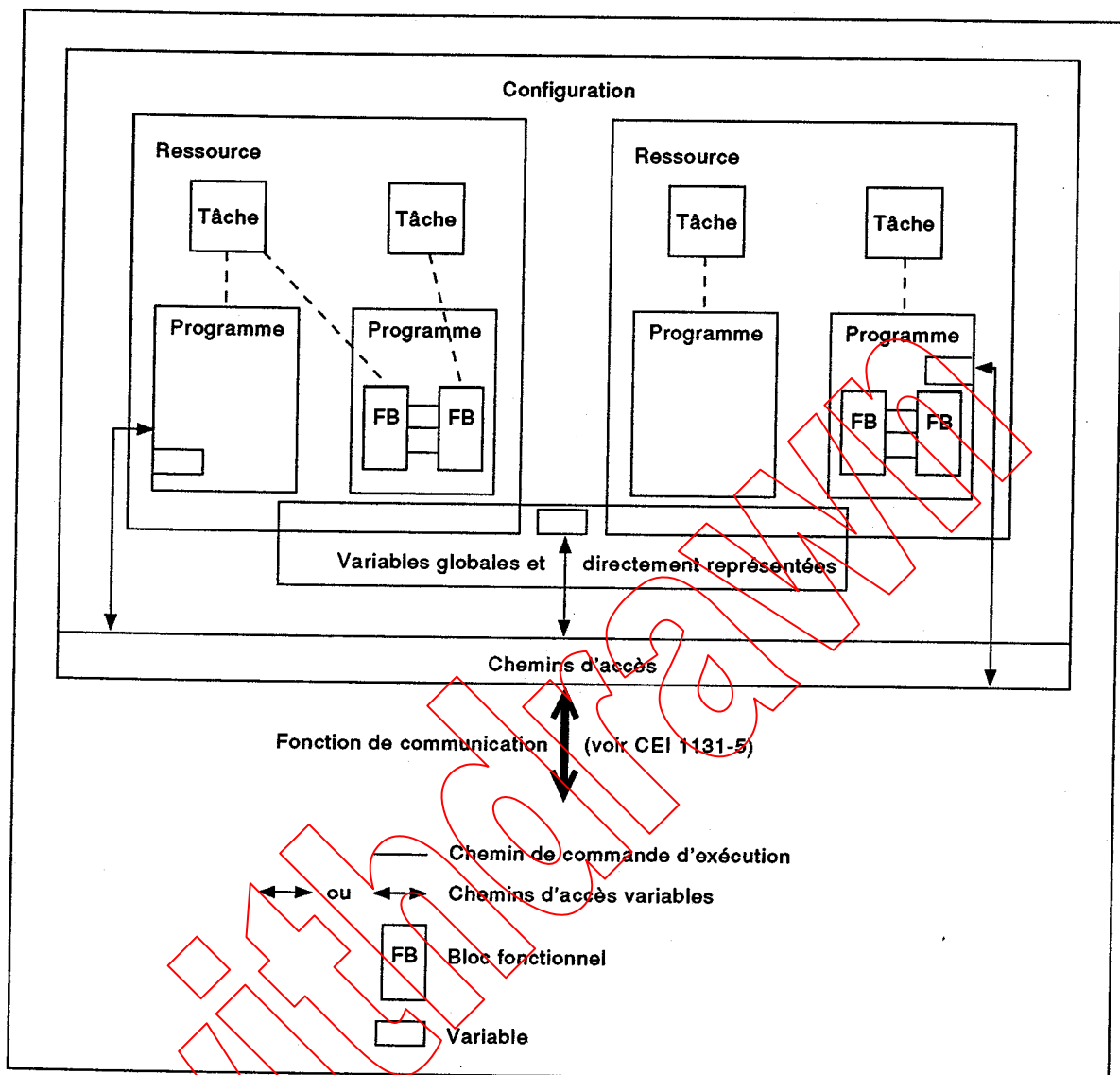
In addition, features are defined which facilitate communication among programmable controllers and other components of automated systems.

The programming language elements defined in this part may be used in an interactive programming environment. The specification of such environments is beyond the scope of this part; however, such an environment shall be capable of producing textual or graphic program documentation in the formats specified in this part.

The material in this part is arranged in "bottom-up" fashion, that is, simpler language elements are presented first, in order to minimize forward references in the text. The remainder of this subclause provides an overview of the material presented in this part and incorporates some general requirements.

1.4.1 Software model

The basic high-level language elements and their interrelationships are illustrated in figure 1. These consist of elements which are *programmed* using the languages defined in this part, that is, *programs* and *function blocks*; and *configuration elements*, namely, *configurations*, *resources*, *tasks*, *global variables*, and *access paths*, which support the installation of programmable controller *programs* into programmable controller systems.

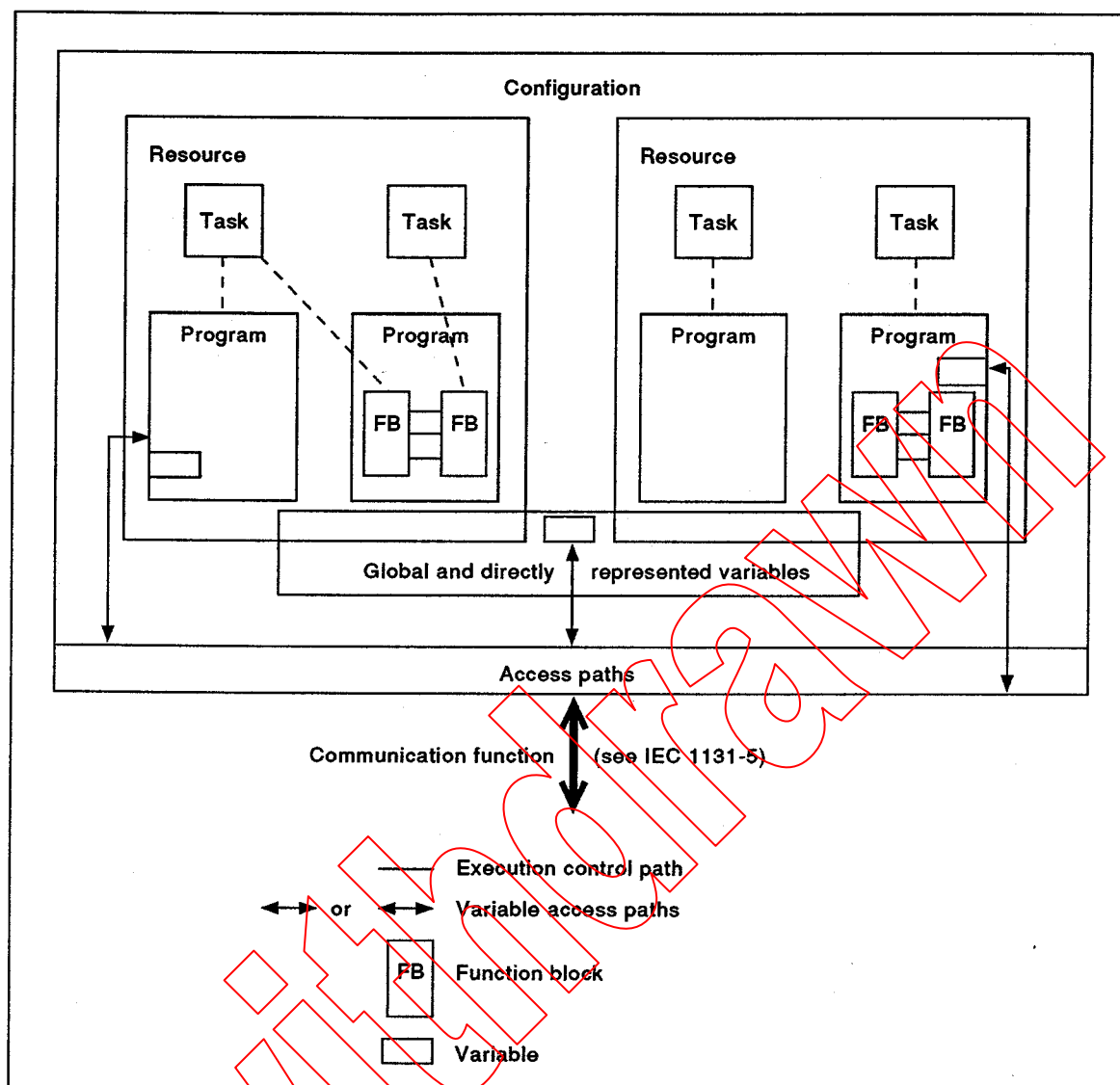


NOTES

- 1 Cette figure n'est donnée qu'à titre indicatif. La représentation graphique n'est pas normative.
- 2 Dans une configuration à ressource unique, il n'est pas nécessaire de représenter explicitement la ressource.

Figure 1 – Modèle logiciel

Une *configuration* se définit comme l'élément de langage qui correspond à un *système d'automate programmable* tel que défini dans la CEI 1131-1. Une *ressource* correspond à une "fonction de traitement du signal" ainsi qu'à ses éventuelles fonctions "interface homme-machine" et "interface capteurs et actionneurs", telles que définies dans la CEI 1131-1. Une configuration contient une ou plusieurs *ressources*, chacune de ces ressources contenant un ou plusieurs *programmes* exécutés sous le contrôle de zéro ou plusieurs *tâches*. Un programme peut ne contenir aucun *bloc fonctionnel*, ou il peut contenir un ou plusieurs blocs fonctionnels ou autres éléments de programme tels que définis dans la présente partie.



NOTES

- 1 This figure is illustrative only. The graphical representation is not normative.
- 2 In a configuration with a single resource, the resource need not be explicitly represented.

Figure 1 – Software model

A *configuration* is the language element which corresponds to a *programmable controller system* as defined in IEC 1131-1. A *resource* corresponds to a "signal processing function" and its "man-machine interface" and "sensor and actuator interface" functions (if any) as defined in IEC 1131-1. A *configuration* contains one or more *resources*, each of which contains one or more *programs* executed under the control of zero or more *tasks*. A *program* may contain zero or more *function blocks* or other language elements as defined in this part.

Les *configurations* et les *ressources* peuvent être lancées et arrêtées par l'intermédiaire des fonctions "interface opérateur", "programmation, essai et contrôle", ou "système d'exploitation", définies dans la CEI 1131-1. Le lancement d'une *configuration* doit entraîner l'initialisation de ses *variables globales*, conformément aux règles spécifiées en 2.4.2, suivie du lancement de toutes les autres *ressources* présentes dans la configuration. Le lancement d'une *ressource* doit entraîner l'initialisation de toutes les *variables* dans la *ressource*, suivie de la validation de toutes les *tâches* dans la *ressource*. L'arrêt d'une *ressource* doit entraîner l'invalidation de toutes ses *tâches*, alors que l'arrêt d'une *configuration* doit entraîner l'arrêt de toutes ses *ressources*. Les mécanismes de commande de *tâches* sont définis en 2.7.2, alors que les mécanismes relatifs au démarrage et à l'arrêt des *configurations* et des *ressources*, par l'intermédiaire de fonctions de communication, sont définis dans la CEI 1131-5.

Les *programmes*, les *ressources*, les *variables globales*, les *chemins d'accès* (ainsi que leurs autorisations d'accès correspondantes) et les *configurations* peuvent être chargés et supprimés par la "fonction communication" définie dans la CEI 1131-1. Le chargement ou la suppression d'une *configuration* ou d'une *ressource* doivent être équivalents au chargement ou à la suppression de tous les éléments qu'elle contient.

Les *chemins d'accès* et leurs autorisations d'accès correspondantes sont définis en 2.7.1.

La projection des éléments de langage, définis dans le présent paragraphe, sur les objets de communication, est définie dans la CEI 1131-5.

1.4.2 Modèle de communication

La figure 2 illustre les différentes manières dont les valeurs de variables peuvent être échangées entre les éléments logiciels.

Comme l'illustre la figure 2a, des valeurs de variables, dans un programme, peuvent être directement échangées en reliant la sortie d'un élément de programme à l'entrée d'un autre élément de programme. Cette liaison est indiquée de façon explicite dans les langages graphiques et de façon implicite dans les langages littéraux.

Des valeurs de variables peuvent être échangées entre programmes dans la même configuration par l'intermédiaire de *variables globales*, comme, par exemple, la variable *x* illustrée à la figure 2b. Ces variables doivent être déclarées comme GLOBAL dans la configuration, et comme EXTERNAL dans les programmes, comme spécifié en 2.4.3.

Comme l'illustre la figure 2c, les valeurs de variables peuvent être échangées entre les différentes parties d'un programme dans la même configuration, ou dans des configurations différentes, ou entre un programme d'automate programmable et un autre programme, à l'aide des blocs de fonction de communication définis dans la CEI 1131-5 et décrits en 2.5.2.3.5. En outre, les systèmes d'automates programmables ou autres peuvent transférer des données qui sont rendues disponibles par des *chemins d'accès*, telles qu'illustrées à la figure 2d, à l'aide des mécanismes définis dans la CEI 1131-5.

Configurations and *resources* can be started and stopped via the "operator interface", "programming, testing, and monitoring", or "operating system" functions defined in IEC 1131-1. The starting of a *configuration* shall cause the initialization of its *global variables* according to the rules given in 2.4.2, followed by the starting of all the resources in the configuration. The starting of a resource shall cause the initialization of all the *variables* in the resource, followed by the enabling of all the *tasks* in the resource. The stopping of a *resource* shall cause the disabling of all its *tasks*, while the stopping of a configuration shall cause the stopping of all its resources. Mechanisms for the control of tasks are defined in 2.7.2, while mechanisms for the starting and stopping of *configurations* and *resources* via communication functions are defined in IEC 1131-5.

Programs, *resources*, *global variables*, *access paths* (and their corresponding access privileges), and *configurations* can be loaded or deleted by the "communication function" defined in IEC 1131-1. The loading or deletion of a *configuration* or *resource* shall be equivalent to the loading or deletion of all the elements it contains.

Access paths and their corresponding access privileges are defined in 2.7.1.

The mapping of the language elements defined in this subclause on to communication objects is defined in IEC 1131-5.

1.4.2 Communication model

Figure 2 illustrates the ways that values of variables can be communicated among software elements.

As shown in figure 2a, variable values within a program can be communicated directly by connection of the output of one program element to the input of another. This connection is shown explicitly in graphical languages and implicitly in textual languages.

Variable values can be communicated between programs in the same configuration via *global variables* such as the variable *x* illustrated in figure 2b. These variables shall be declared as GLOBAL in the configuration, and as EXTERNAL in the programs, as specified in 2.4.3.

As illustrated in figure 2c, the values of variables can be communicated between different parts of a program, between programs in the same or different configurations, or between a programmable controller program and a non-programmable controller system, using the communication function blocks defined in IEC 1131-5 and described in 2.5.2.3.5. In addition, programmable controllers or non-programmable controller systems can transfer data which is made available by *access paths*, as illustrated in figure 2d, using the mechanisms defined in IEC 1131-5.

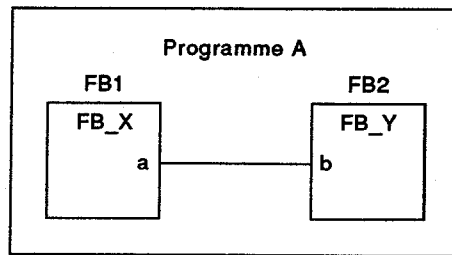


Figure 2a – Liaison de flux de données dans un programme

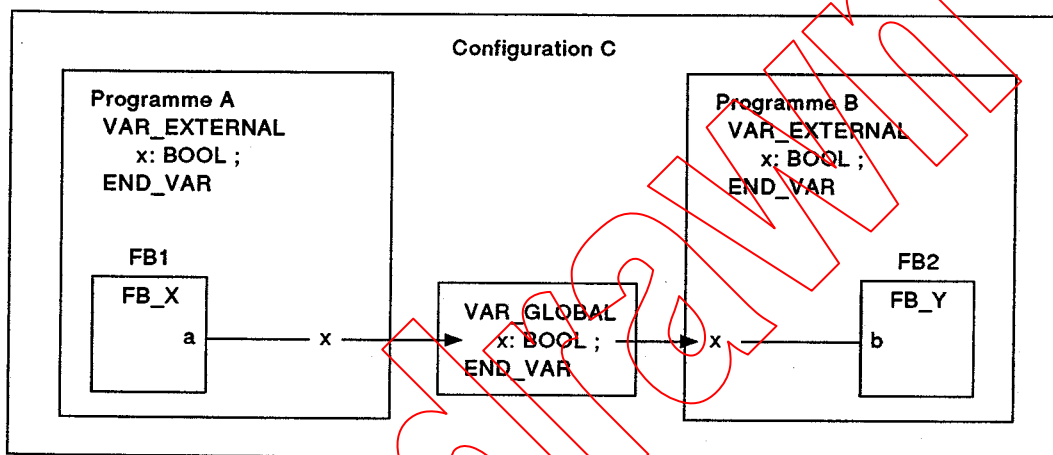


Figure 2b – Communication par l'intermédiaire de variables globales

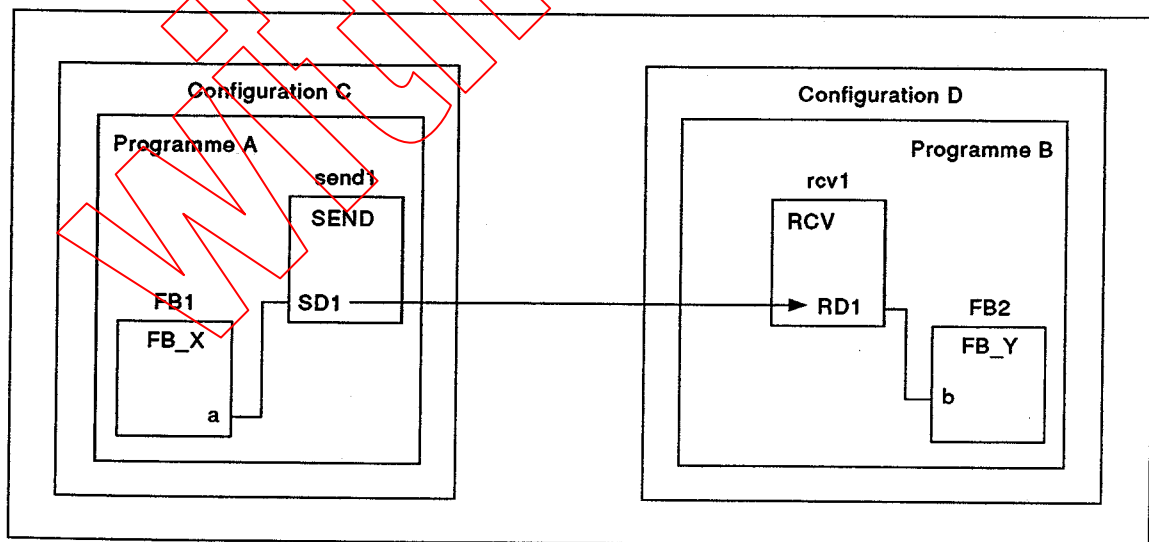


Figure 2c – Blocs fonctionnels de communication

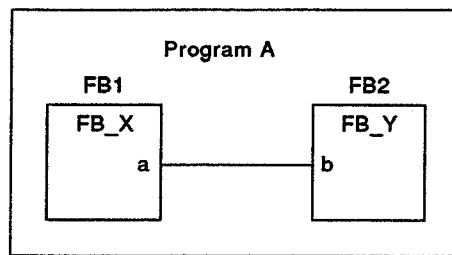


Figure 2a – Data flow connection within a program

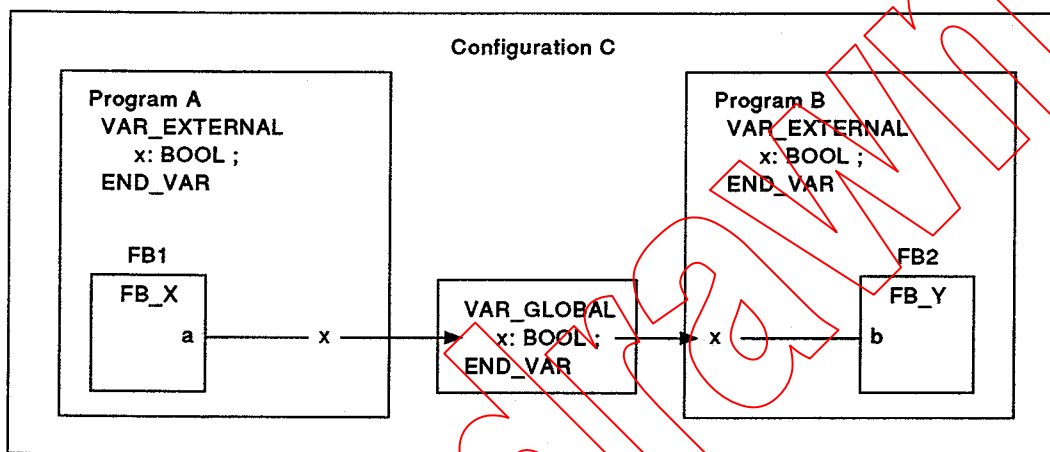


Figure 2b – Communication via GLOBAL variables

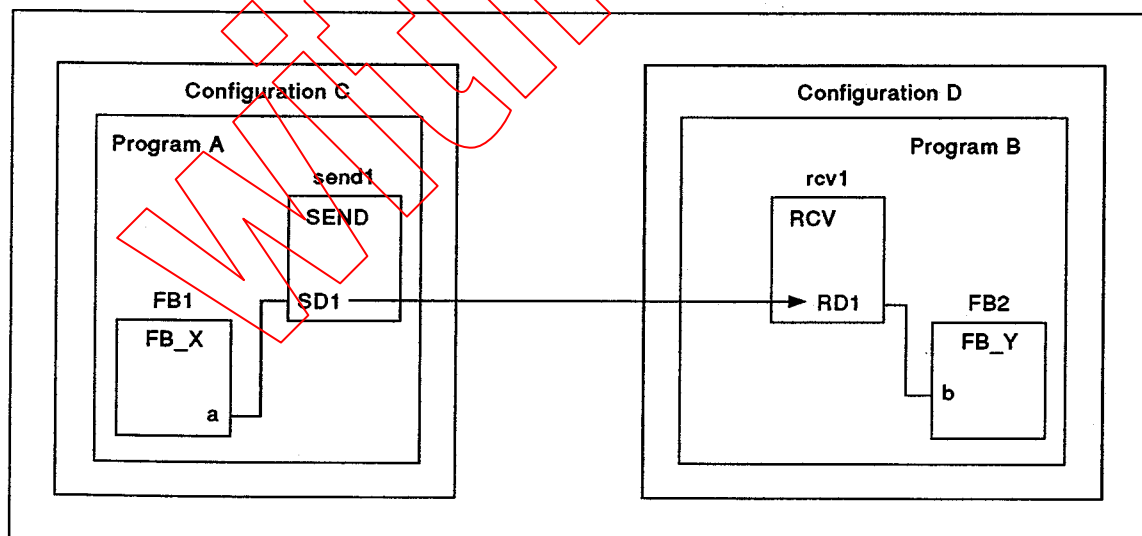


Figure 2c – Communication function blocks

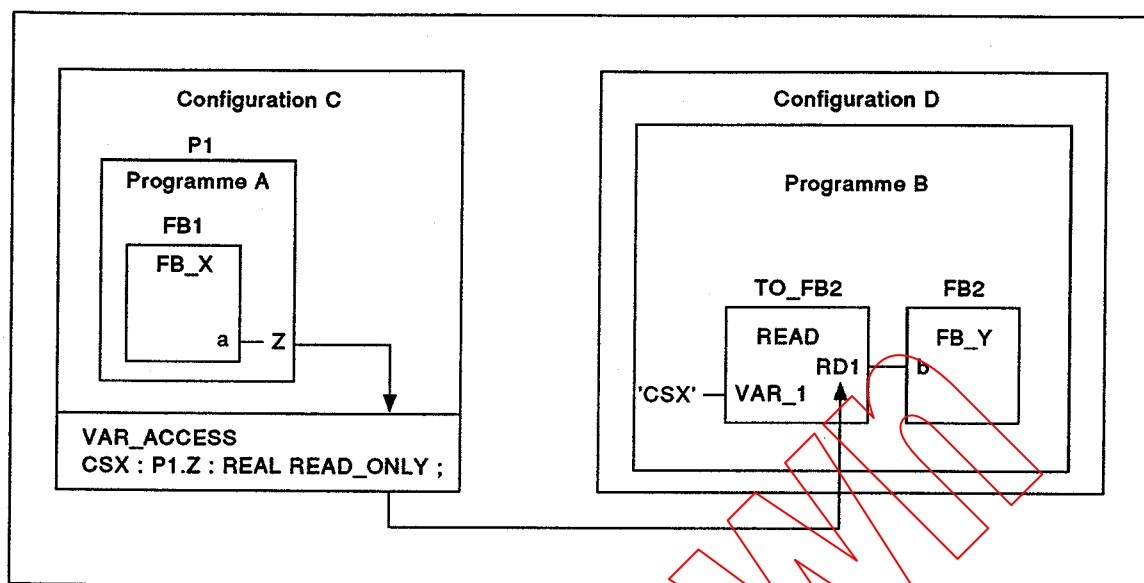


Figure 2d – Communication par l'intermédiaire de chemins d'accès

NOTES

- 1 Cette figure n'est fournie qu'à titre indicatif. La représentation graphique n'est pas normative.
- 2 Dans ces exemples, les configurations C et D sont respectivement considérées comme ayant une seule ressource.
- 3 Les détails relatifs aux blocs fonctionnels de communication ne sont pas représentés dans cette figure. Se reporter à 2.5.2.3.5 et à la CEI 1131-5.
- 4 Comme spécifié en 2.7, les chemins d'accès peuvent être déclarés sur des variables directement représentées, sur des variables globales, ou sur des variables d'entrée, de sortie, ou internes programme.
- 5 La CEI 1131-5 spécifie les moyens dont peuvent disposer les systèmes d'automates programmables et d'autres systèmes pour l'écriture et la lecture de variables.

Figure 2 – Modèle de communication

1.4.3 Modèle de communication

Les éléments des langages de programmation d'automates programmables et les paragraphes au sein desquels ils figurent dans la présente partie sont classés comme suit:

- Types de données (2.3)
- Unités d'organisation de programmes (2.5)
 - Fonctions (2.5.1)
 - Blocs fonctionnels (2.5.2)
 - Programmes (2.5.3)
- Éléments de schémas "diagramme fonctionnel en séquence" (SFC) (2.6)
- Éléments de configuration (2.7)
 - Variables globales 2.7.1)
 - Ressources (2.7.1)
 - Tâches (2.7.2)
 - Chemins d'accès (2.7.1)

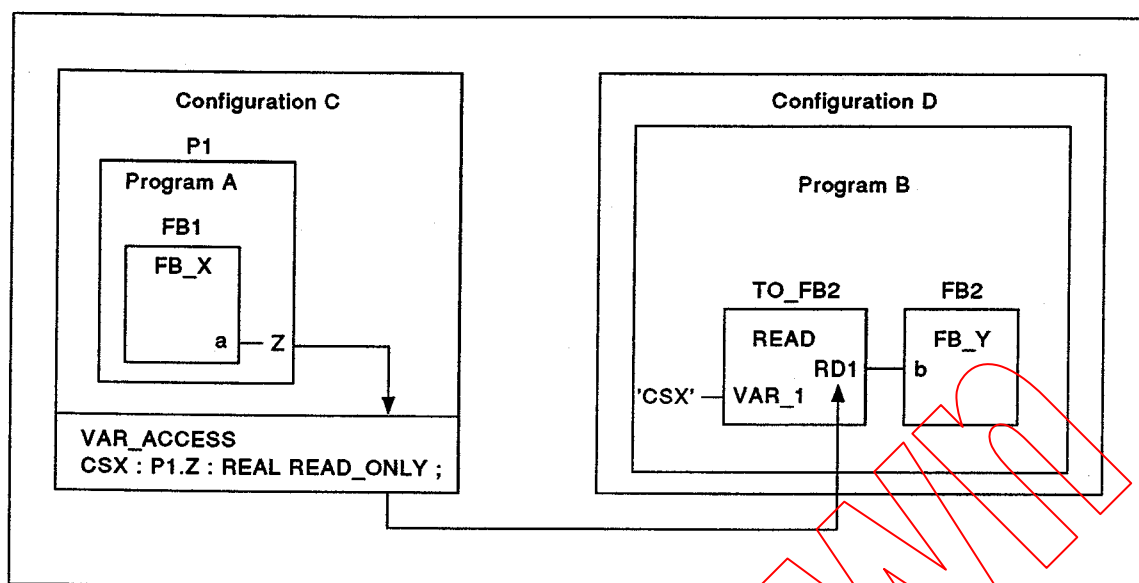


Figure 2d – Communication via access paths

NOTES

- 1 This figure is illustrative only. The graphical representation is not normative.
- 2 In these examples, configurations c and d are each considered to have a single resource.
- 3 The details of the communication function blocks are not shown in this figure. See 2.5.2.3.5 and IEC 1131-5.
- 4 As specified in 2.7, access paths can be declared on directly represented variables, global variables, or program input, output, or internal variables.
- 5 IEC 1131-5 specifies the means by which both PC and non-PC systems can use access paths for reading and writing of variables.

Figure 2 – Communication model

1.4.3 Programming model

The elements of programmable controller programming languages, and the subclauses in which they appear in this part, are classified as follows:

- Data types (2.3)
- Program organization units (2.5)
 - Functions (2.5.1)
 - Function blocks (2.5.2)
 - Programs (2.5.3)
- Sequential Function Chart (SFC) elements (2.6)
- Configuration elements (2.7)
 - Global variables (2.7.1)
 - Resources (2.7.1)
 - Tasks (2.7.2)
 - Access paths (2.7.1)

Comme l'illustre la figure 3, la combinaison de ces éléments doit obéir aux règles suivantes:

- 1) Les *types de données* dérivés doivent être déclarés conformément aux indications de 2.3.3, à l'aide des types de données standards, spécifiés en 2.3.1 et 2.3.2, ainsi que de tout type de donnée précédemment dérivé.
- 2) Les *fonctions* dérivées peuvent être déclarées conformément aux indications de 2.5.1.3, à l'aide de types de données standards ou dérivés, des fonctions standards définies au paragraphe 2.5.1.5, ainsi que de tout type de fonction précédemment dérivé. Cette déclaration doit utiliser les mécanismes définis pour le langage IL, ST, LD ou FBD.
- 3) Les *blocs fonctionnels* dérivés peuvent être déclarés conformément aux indications de 2.5.2.2, à l'aide de types de données et de fonctions standards ou dérivés, des blocs fonctionnels définis en 2.5.2.3, ainsi que de tout bloc fonctionnel précédemment dérivé. Cette déclaration doit utiliser les mécanismes définis pour le langage IL, ST ou FBD, et peut inclure des éléments de schéma "diagramme fonctionnel en séquence" (SFC), tels que définis en 2.6.
- 4) Un *programme* doit être déclaré conformément aux indications de 2.5.3, à l'aide de types de données, de fonctions et de blocs fonctionnels standards ou dérivés. Cette déclaration doit utiliser le mécanisme défini pour le langage IL, ST, LD ou FBD, et peut inclure des éléments de schéma "diagramme fonctionnel en séquence" (SFC), tels que définis en 2.6.
- 5) Les *programmes* peuvent être combinés en *configurations*, à l'aide des éléments définis en 2.7, c'est-à-dire des *variables globales*, des *ressources*, des *tâches* et des *chemins d'accès*.

La référence aux types de données, aux fonctions et aux blocs fonctionnels "précédemment dérivés" dans les règles précédentes, est destinée à garantir que, lorsqu'un tel élément a été déclaré, sa définition est disponible, par exemple, dans une "bibliothèque" d'éléments dérivés, pour une utilisation dans des dérivations ultérieures. Par conséquent, la déclaration d'un type d'élément dérivé ne doit pas être contenue dans la déclaration d'un autre type d'élément dérivé.

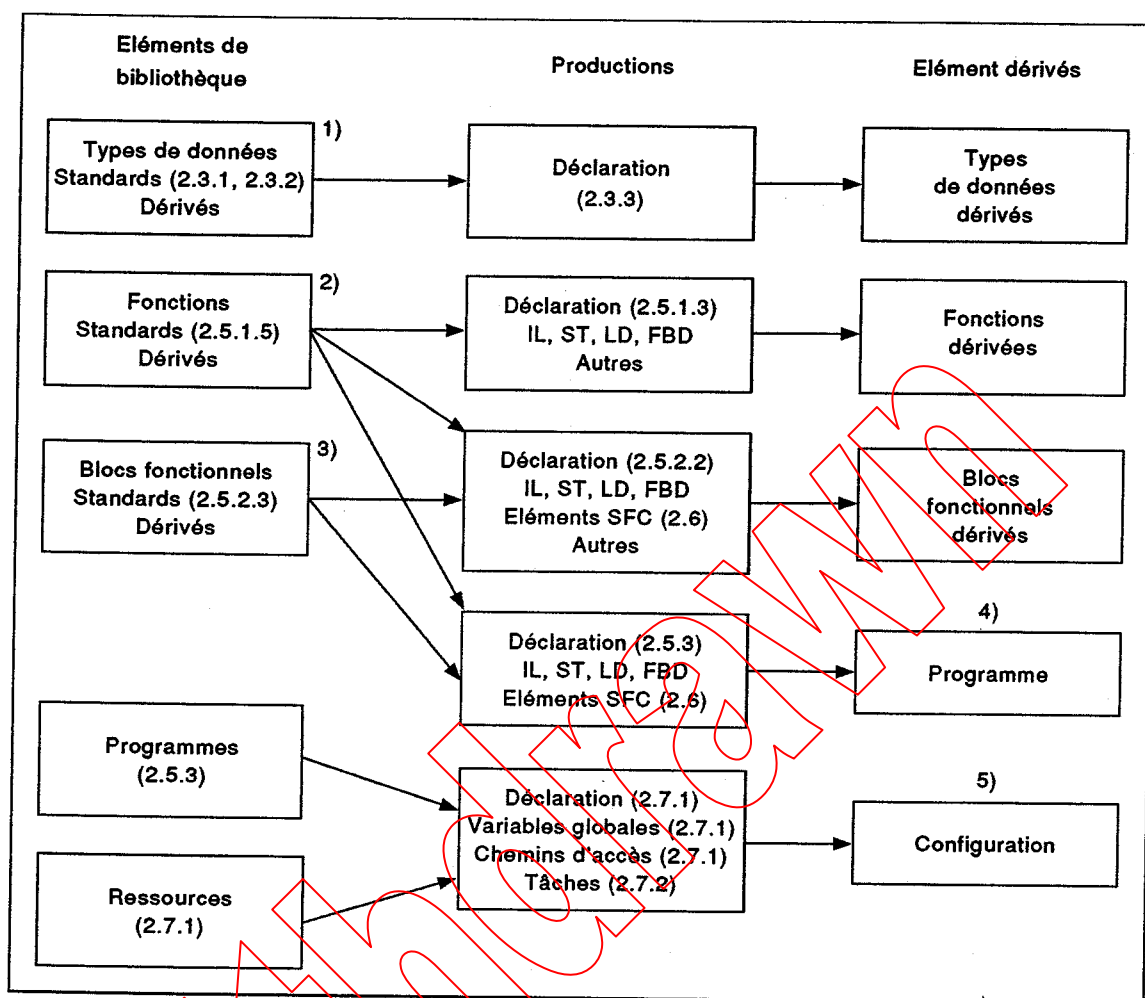
Il est possible d'utiliser un langage de programmation autre que ceux définis dans la présente norme dans la déclaration d'une *fonction* ou d'un *bloc fonctionnel*. Il est nécessaire de définir dans la présente norme le moyen par lequel un programme utilisateur, écrit dans l'un des langages définis dans la présente norme, déclenche l'exécution d'une telle fonction dérivée ou d'un tel bloc fonctionnel dérivé, et accède aux données associées à une telle fonction ou à un tel bloc fonctionnel.

As shown in figure 3, the combination of these elements shall obey the following rules:

- 1) Derived *data types* shall be declared as specified in 2.3.3, using the standard data types specified in 2.3.1 and 2.3.2 and any previously derived data types.
- 2) Derived *functions* can be declared as specified in 2.5.1.3, using standard or derived data types, the standard functions defined in 2.5.1.5, and any previously derived functions. This declaration shall use the mechanisms defined for the IL, ST, LD or FBD language.
- 3) Derived *function blocks* can be declared as specified in 2.5.2.2, using standard or derived data types and functions, the standard function blocks defined in 2.5.2.3, and any previously derived function blocks. This declaration shall use the mechanisms defined for the IL, ST, LD, or FBD language, and can include Sequential Function Chart (SFC) elements as defined in 2.6.
- 4) A *program* shall be declared as specified in 2.5.3, using standard or derived data types, functions, and function blocks. This declaration shall use the mechanisms defined for the IL, ST, LD, or FBD language, and can include Sequential Function Chart (SFC) elements as defined in 2.6.
- 5) *Programs* can be combined into *configurations* using the elements defined in 2.7, that is, *global variables*, *resources*, *tasks*, and *access paths*.

Reference to "previously derived" data types, functions, and function blocks in the above rules is intended to imply that once such a derived element has been declared, its definition is available, e.g., in a "library" of derived elements, for use in further derivations. Therefore, the declaration of a derived element type shall not be contained within the declaration of another derived element type.

A programming language other than one of those defined in this standard may be used in the declaration of a *function* or *function block*. The means by which a user program written in one of the languages defined in this standard invokes the execution of, and accesses the data associated with, such a derived function or function block shall be as defined in this standard.



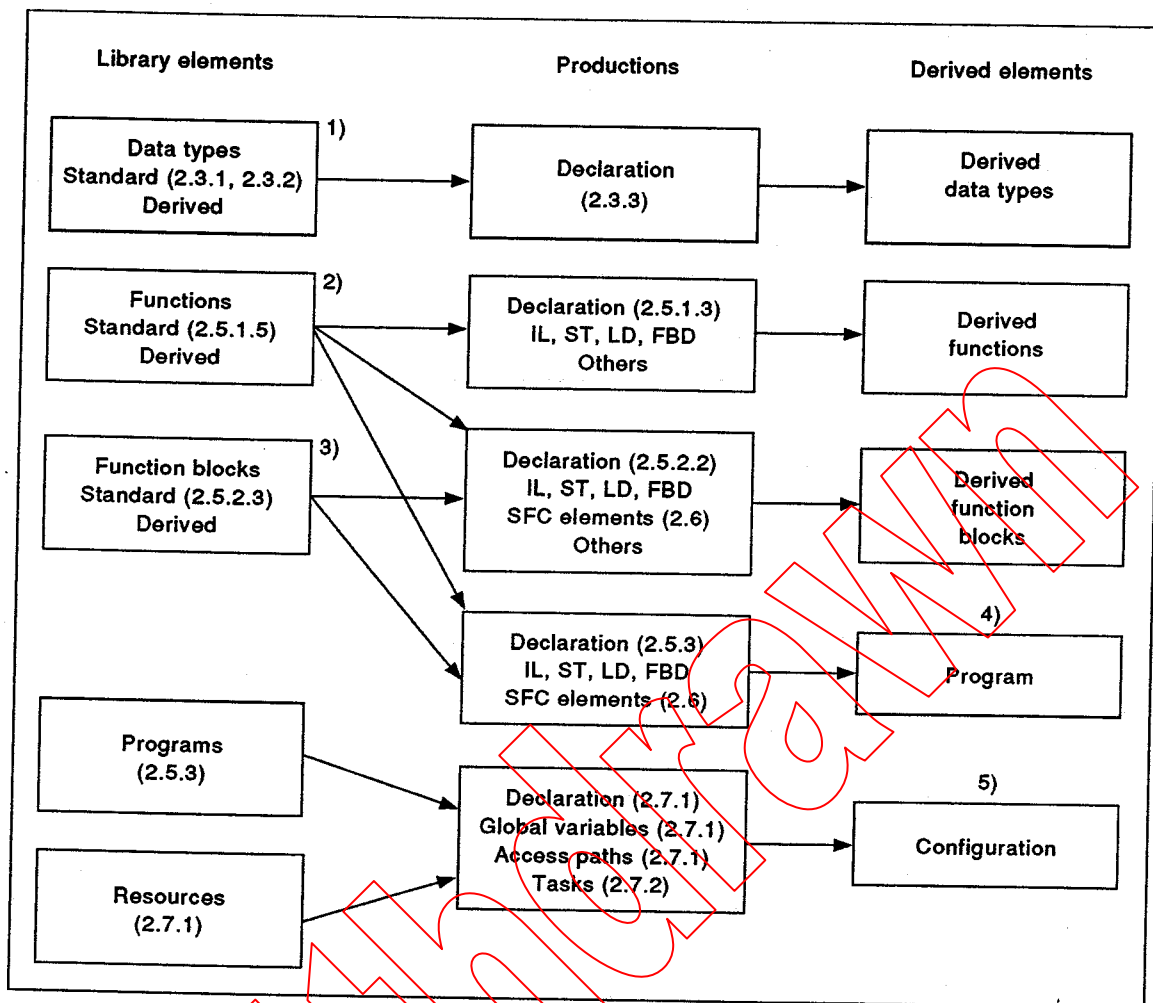
NOTES

- 1 Les numéros 1) à 5) se rapportent aux alinéas correspondants de 1.4.3.
- 2 Les types de données sont utilisés dans toutes les productions. Pour plus de clarté, les liaisons correspondantes ont été omises dans cette figure.

Figure 3 – Combinaison d'éléments de langages pour automates programmables:
 Langage à contacts-LD (4.2)
 Schéma en blocs fonctionnels-FBD (4.3)
 Liste d'instructions-IL (3.2)
 Littéral structuré-ST (3.3)
 Autres: Autres langages de programmation (1.2.3)

1.5 Conformité

Le présent paragraphe définit les prescriptions auxquelles doivent satisfaire les systèmes et les programmes d'automates programmables qui sont réputés conformes aux prescriptions de cette partie de la CEI 1131.



NOTES

- 1 The parenthesized numbers 1) to 5) refer to corresponding paragraphs in 1.4.3.
- 2 Data types are used in all productions. For clarity, the corresponding linkages are omitted in this figure.

Figure 3 – Combination of programmable controller language elements
 LD-Ladder Diagram (4.2)
 FBD-Function Block Diagram (4.3)
 IL-Instruction List (3.2)
 ST-Structured Text (3.3)
 Others: Other programming languages (1.4.3)

1.5 Compliance

This subclause defines the requirements which shall be met by programmable controller systems and programs which claim compliance with this part of IEC 1131.

1.5.1 Systèmes d'automates programmables

Un système d'automate programmable, tel qu'il a été défini dans le CEI 1131-1, qui prétend se conformer, totalement ou partiellement, aux exigences de cette partie de la CEI 1131, ne doit être considéré comme tel que lorsque les conditions ci-dessous sont remplies.

Un énoncé de conformité doit être inclus dans la documentation accompagnant le système, ou doit être produit par le système lui-même. La formulation de l'énoncé de conformité doit être comme suit:

"Ce système est conforme aux prescriptions de la CEI 1131-3, en ce qui concerne les caractéristiques de langage suivantes:"

suivie d'un ensemble de tableaux de conformité présentant le format suivant:

Titre du tableau

Tableau n°	Caractéristique n°	Description des caractéristiques
...	...	...

Les numéros et la description des caractéristiques doivent être pris dans les tableaux donnés dans le paragraphe correspondant dans la présente partie de la CEI 1131. Les titres des tableaux doivent être pris parmi ceux indiqués dans le tableau ci-dessous.

Titre du tableau	Pour les caractéristiques indiquées en:
Eléments communs	Article 2
Eléments littéraux communs	Paragraphe 3.1
Eléments de langage IL	Paragraphe 3.2.1 à 3.2.3
Eléments de langage ST	Paragraphe 3.3.1 à 3.3.2.4
Eléments graphiques communs	Paragraphe 4.1 à 4.1.4
Eléments de langage LD	Paragraphe 4.2 à 4.2.6
Eléments de langage FBD	Paragraphe 4.3 à 4.3.3

Un système d'automate programmable, conforme aux exigences de la présente partie, eu égard à un langage défini dans la présente partie:

- ne doit pas nécessiter l'inclusion d'éléments de langage de remplacement ou supplémentaires, pour accomplir l'une des fonctions spécifiées dans la présente partie;
- doit être accompagné d'un document qui spécifie les valeurs de tous les paramètres propres à une application, énumérées dans l'annexe D;
- doit être capable de déterminer si un élément de langage utilisateur est conforme ou non à toute prescription de la présente partie, lorsqu'une telle non-conformité n'est pas imputable à l'une des erreurs indiquées à l'annexe E, et de signaler cette détermination à l'utilisateur. Dans le cas où le système n'examine pas la totalité de l'unité

1.5.1 Programmable controller systems

A programmable controller system, as defined in IEC 1131-1, which claims to comply, wholly or partially, with the requirements of this part of IEC 1131 shall do so only as described below.

A compliance statement shall be included in the documentation accompanying the system, or shall be produced by the system itself. The form of the compliance statement shall be:

"This system complies with the requirements of IEC 1131-3, for the following language features:"

followed by a set of compliance tables in the following format:

Table title

Table No.	Feature No.	Features description
...	...	...

Table and feature numbers and descriptions are to be taken from the tables given in the relevant subclauses of this part of IEC 1131. Table titles are to be taken from the following table.

Table title	For features in:
Common elements	Clause 2
Common textual elements	Subclause 3.1
IL language elements	Subclauses 3.2.1 to 3.2.3
ST language elements	Subclauses 3.3.1 to 3.3.2.4
Common graphical elements	Subclauses 4.1 to 4.1.4
LD language elements	Subclauses 4.2 to 4.2.6
FBD language elements	Subclauses 4.3 to 4.3.3

A programmable controller system complying with the requirements of this part with respect to a language defined in this part:

- shall not require the inclusion of substitute or additional language elements in order to accomplish any of the features specified in this part;
- shall be accompanied by a document that specifies the values of all implementation-dependent parameters as listed in annex D;
- shall be able to determine whether or not a user's language element violates any requirement of this part, where such a violation is not designated an error in annex E, and report the result of this determination to the user. In the case where the system does not examine the whole program organization unit, the user shall be notified that

d'organisation de programme, l'utilisateur doit être avisé du fait que la détermination est incomplète, si aucune non-conformité n'a été décelée dans la partie examinée de l'unité d'organisation de programme;

d) doit traiter chaque non-conformité utilisateur qui correspond à une erreur indiquée à l'annexe E de la présente partie, en procédant selon au moins une des façons suivantes:

1) il doit y avoir une déclaration dans un document d'accompagnement indiquant que l'erreur n'a pas été signalée;

2) le système doit signaler, au cours de la préparation du programme pour l'exécution, que cette erreur risque de se produire;

3) le système doit signaler l'erreur, au cours de la préparation du programme pour l'exécution;

4) le système doit signaler l'erreur pendant l'exécution du programme et lancer les procédures appropriées de traitement des erreurs, définies par le système ou par l'utilisateur;

et si une éventuelle non-conformité, reconnue comme une erreur, est traitée de la manière décrite au point 1) de l'alinéa d) ci-dessus, une note se rapportant à chacun de ces traitements doit apparaître dans une section séparée du document d'accompagnement;

e) doit être accompagné d'un document qui décrit séparément toutes caractéristiques prises en charge par le système, mais qui sont interdites ou non spécifiées dans la présente partie. De telles caractéristiques doivent être décrites comme des "extensions au langage <langage> tel que défini dans la CEI 1131-3";

f) doit être capable de traiter, d'une manière similaire à celle spécifiée pour les erreurs, toute utilisation de toute extension de ce type;

g) doit être capable de traiter, d'une manière similaire à celle spécifiée pour les erreurs, toute utilisation de l'une des caractéristiques propres à une application, spécifiées à l'annexe D;

h) ne doit utiliser aucun nom de type de donnée, de fonction, ou de bloc fonctionnel standard défini dans la présente partie, pour des caractéristiques définies par le fabricant et dont la fonctionnalité est différente de celle décrite dans la présente partie;

i) doit être accompagné d'un document définissant, sous la forme spécifiée à l'annexe A, la syntaxe formelle de tous les éléments de langage littéral, prise en charge par le système.

L'expression "être capable de" est utilisée dans le présent paragraphe pour permettre la mise en application d'un commutateur logiciel grâce auquel l'utilisateur peut commander la signalisation des erreurs.

Dans les cas où la compilation ou la saisie d'un programme est suspendue en raison de l'insuffisance des tableaux, etc., une détermination incomplète du type "aucune non-conformité n'a été décelée, mais l'examen est incomplet" satisfera aux prescriptions de ce paragraphe.

1.5.2 Programmes

Un programme d'automate programmable, conforme aux prescriptions de la CEI 1131-3:

a) ne doit utiliser que les caractéristiques spécifiées dans la présente partie pour le langage spécifique utilisé;

the determination is incomplete whenever no violations have been detected in the portion of the program organization unit examined;

d) shall treat each user violation that is designated an error in annex E in at least one of the following ways:

- 1) there shall be a statement in an accompanying document that the error is not reported;
- 2) the system shall report during preparation of the program for execution that an occurrence of that error is possible;
- 3) the system shall report the error during preparation of the program for execution;
- 4) the system shall report the error during execution of the program and initiate appropriate system- or user-defined error handling procedures;

and if any violations that are designated as errors are treated in the manner described in d)1) above, then a note referencing each such treatment shall appear in a separate section of the accompanying document;

e) shall be accompanied by a document that separately describes any features accepted by the system that are prohibited or not specified in this part. Such features shall be described as being "extensions to the <language> language as defined in IEC 1131-3";

f) shall be able to process in a manner similar to that specified for errors any use of any such extension;

g) shall be able to process in a manner similar to that specified for errors any use of one of the implementation-dependent features specified in annex D;

h) shall not use any of the standard data type, function or function block names defined in this part for manufacturer-defined features whose functionality differs from that described in this part;

i) shall be accompanied by a document defining, in the form specified in annex A, the formal syntax of all textual language elements supported by the system.

The phrase "be able to" is used in this subclause to permit the implementation of a software switch with which the user may control the reporting of errors.

In cases where compilation or program entry is aborted due to some limitation of tables, etc., an incomplete determination of the kind "no violations were detected, but the examination is incomplete" will satisfy the requirements of this subclause.

1.5.2 Programs

A programmable controller program complying with the requirements of IEC 1131-3:

- a) shall use only those features specified in this part for the particular language used;

- b) ne doit utiliser aucune caractéristique identifiée comme une extension du langage;
- c) ne doit appuyer sur aucune interprétation particulière des caractéristiques propres à une application.

Les résultats rendus par un programme conforme doivent être identiques lorsqu'ils sont traités par un système conforme qui prend en charge les caractéristiques utilisées par le programme, sauf si ces résultats sont influencés par le séquençement de l'exécution du programme, l'utilisation de caractéristiques propres à une application (telles qu'énumérées dans l'annexe D) dans le programme, et l'exécution de procédures de traitement d'erreurs.

2 Éléments communs

Le présent article définit les éléments littéraux et graphiques qui sont communs à tous les langages de programmation d'automates programmables spécifiés dans cette partie de la CEI 1131.

2.1 Utilisation de caractères imprimés

2.1.1 Jeu de caractères

Les langages littéraux et les éléments littéraux des langages graphiques doivent être représentés en fonction du "tableau des codes de base" du jeu de caractères ISO/IEC 646.

Le codage de caractères, à partir des jeux de caractères nationaux ou étendus (8 bits), doit être conforme à l'ISO/IEC 646.

Le *jeu de caractères requis*, illustré en tant que caractéristique 1 dans le tableau 1, se compose de tous les caractères des colonnes 3 à 7 du "tableau des codes de base", présenté en tant que tableau 1 dans l'ISO/IEC 646, à l'exception des caractères minuscules et des positions de caractères réservées ou disponibles en option, pour l'utilisation dans des jeux de caractères nationaux.

Le fabricant doit choisir une option (a ou b) pour chacune des caractéristiques (3a, b) à (6a, b) du tableau 1, conformément aux règles suivantes:

- Le "signe livre sterling" (£) doit être utilisé à la place du "signe numéro" (#), lorsque le premier signe occupe la position de caractère 2/3 de l'application nationale du jeu de caractères ISO/IEC 646.
- Le "signe monétaire" doit être utilisé à la place du signe "dollar" (\$), lorsque le premier occupe la position de caractère 2/4 d'une application nationale du jeu de caractères ISO 646.
- Lorsque la position de caractère 7/12 dans le jeu de caractères ISO 646 est utilisée par un autre caractère dans un jeu de caractères national, "le point d'exclamation" (!) doit être utilisé en position 2/1 pour représenter des lignes verticales.
- Pour la délimitation des indices, les parenthèses à droite et à gauche "()" doivent être utilisées à la place des crochets à droite et à gauche "[]" lorsque ces derniers occupent les positions de caractères d'une application nationale du jeu de caractères ISO 646.

NOTE - L'utilisation de caractères provenant de jeux de caractères nationaux constitue un exemple d'extension de la présente norme.

- b) shall not use any features identified as extensions to the language;
- c) shall not rely on any particular interpretation of implementation-dependent features.

The results produced by a complying program shall be the same when processed by any complying system which supports the features used by the program, except as these results are influenced by program execution timing, the use of implementation-dependent features (as listed in annex D) in the program, and the execution of error handling procedures.

2 Common elements

This clause defines textual and graphic elements which are common to all the programmable controller programming languages specified in this part of IEC 1131.

2.1 Use of printed characters

2.1.1 Character set

Textual languages and textual elements of graphic languages shall be represented in terms of the "Basic code table" of the ISO/IEC 646 character set.

The encoding of characters from national or extended (8-bit) character sets shall be consistent with ISO/IEC 646.

The *required character set* shown as feature 1 in table 1 consists of all the characters in columns 3 to 7 of the "Basic code table" given as table 1 in ISO/IEC 646, except for lower-case letters and those character positions which are reserved or optionally available for use in national character sets.

The manufacturer shall support one option (a or b) for each of features (3a,b) to (6a,b) of table 1, according to the following rules:

- The "pound sign" (£) shall be used in place of the "number sign" (#) when the former occupies character position 2/3 of a national implementation of the ISO/IEC 646 character set.
- The "currency sign" shall be used in place of the "dollar sign" (\$) when the former occupies character position 2/4 of a national implementation of the ISO/IEC 646 character set.
- When the 7/12 character position in the ISO/IEC 646 character set is used by another character in a national set, the "exclamation mark" (!) at position 2/1 shall be used to represent vertical lines.
- For delimitation of subscripts, the left and right parentheses "()" shall be used in place of the left and right brackets "[]" when the latter occupy character positions of a national implementation of the ISO/IEC 646 character set.

NOTE - The use of characters from national character sets is a typical extension of this standard.

Tableau 1 – Caractéristiques des jeux de caractères

N°	Description
1	Jeu de caractères requis (voir 2.1.1)
2	Caractères minuscules
3a 3b	Signe numéro (#) ou Signe livre sterling (£)
4a 4b	Signe dollar (\$) ou Signe monétaire
5a 5b	Barre verticale () ou Point d'exclamation (!)
6a 6b	Délimiteur d'indice: Crochet gauche et droit "[]" ou Parenthèses gauche et droite "()"
NOTE - Lorsque des lettres minuscules (caractéristique 2) sont prises en charge, les caractères majuscules représentant des mêmes lettres ne doivent pas être une signification particulière dans les éléments de langage (sauf dans les commentaires tels que définis en 2.1.5 et dans les libellés de cordons de caractères et des variables du type STRING, respectivement définis en 2.2.2 et 2.3.1); par exemple, les identificateurs "abcd", "ABCD" et "aBCd" doivent être interprétés de façon identique.	

2.1.2 Identificateurs

Un *identificateur* est un cordon de lettres, de chiffres et de caractères de soulignement qui doit commencer par une lettre ou par un caractère de soulignement.

Les caractères de soulignement doivent avoir une signification particulière dans les identificateurs; par exemple, "A_BCD" et "AB_CD" doivent être interprétés comme étant des identificateurs différents. Les caractères de soulignement multiples en tête ou intégrés ne sont pas autorisés.

Les identificateurs ne doivent pas comporter de caractères d'espaces intercalaires (SP).

Au moins six caractères uniques doivent être pris en charge dans tous les systèmes qui acceptent l'utilisation d'identificateurs; par exemple, "ABCDE1" doit être considéré comme différent de "ABCDE2" dans tous ces types de systèmes.

Les caractéristiques des identificateurs et des exemples se trouvent à la figure 2.

Tableau 2 – Caractéristiques des identificateurs

N°	Description des caractéristiques	Exemples
1	Majuscules et nombres	IW215 IW215Z QX75 IDENT
2	Majuscule et minuscule, nombres, caractères de soulignement intégrés	Tous les exemples ci-dessus, plus: LIM_SW_5 LimSw5 abcd ab_Cd
3	Majuscule et minuscule, nombres, caractères de soulignement en tête ou intégrés	Tous les exemples ci-dessus, plus: _MAIN _12V7

Table 1 – Character set features

No	Description
1	Required character set (see 2.1.1)
2	Lower case characters
3a 3b	Number sign (#) or Pound sign (£)
4a 4b	Dollar sign (\$) or Currency sign
5a 5b	Vertical bar () or Exclamation mark (!)
6a 6b	Subscript delimiters: Left and right brackets "[]" or Left and right parentheses "()"
NOTE - When lower-case letters (feature 2) are supported, the case of letters shall not be significant in language elements (except within terminal symbols as defined in annexes A and B, comments as defined in 2.1.5, string literals as defined in 2.2.2, and variables of type STRING as defined in 2.3.1), e.g., the identifiers "abcd", "ABCD" and "aBCd" shall be interpreted identically.	

2.1.2 Identifiers

An *identifier* is a string of letters, digits, and underline characters which shall begin with a letter or underline character.

Underlines shall be significant in identifiers, e.g., "A_BCD" and "AB_CD" shall be interpreted as different identifiers. Multiple leading or multiple embedded underlines are not allowed.

Identifiers shall not contain imbedded space (SP) characters.

At least six characters of uniqueness shall be supported in all systems which support the use of identifiers, e.g., "ABCDE1" shall be interpreted as different from "ABCDE2" in all such systems.

Identifier features and examples are shown in table 2.

Table 2 – Identifier features

No.	Feature description	Examples
1	Upper case and numbers	IW215 IW215Z QX75 IDENT
2	Upper and lower case, numbers, embedded underlines	All the above plus: LIM_SW_5 LimSw5 abcd ab_Cd
3	Upper and lower case, numbers, leading or embedded underlines	All the above plus: _MAIN _12V7

2.1.3 Mots clés

Les *mots clés* sont des combinaisons uniques de caractères utilisés comme des éléments syntaxiques individuels, tels que définis à l'annexe B. Tous les mots clés utilisés dans la présente partie sont énumérés à l'annexe C. Les mots clés ne doivent pas comporter de caractères d'espaces intercalaires. Les mots clés indiqués à l'annexe C ne doivent servir à aucune autre fin, comme par exemple les noms de variables ou extensions, tels que définis en 1.5.1.

NOTE - Les organismes nationaux de normalisation peuvent publier des tableaux de traductions des mots clés donnés à l'annexe C.

2.1.4 Utilisation des espaces

L'utilisateur doit être autorisé à insérer un ou plusieurs espaces (position de code 2/0 dans le jeu de caractères ISO/IEC 646) n'importe où dans le texte de programmes d'automates programmables, sauf dans des mots clés, des libellés, des identificateurs, ou des combinaisons de délimiteurs (pour des commentaires tels que ceux définis ci-après).

2.1.5 Commentaires

Les commentaires de l'utilisateur doivent être respectivement délimités, à leur début et à leur fin, par les combinaisons de caractères spéciaux "(" et ")" comme indiqué au tableau 3. Sauf dans le langage IL tel que défini au paragraphe 3.2, des commentaires doivent être autorisés n'importe où dans le programme, lorsque des espaces sont autorisés, à l'exception des libellés de cordons de caractères définis au paragraphe 2.2.2. Les commentaires ne doivent avoir aucune signification syntaxique ni sémantique, dans aucun des langages définis dans la présente partie.

Les commentaires imbriqués, comme par exemple (* (* IMBRIQUE *) *), ne sont pas autorisés.

Tableau 3 – La caractéristique commentaire

N°	Description des caractéristiques	Exemples
1	Commentaires	(*****) (* Un commentaire encadré *) (*****)

2.2 Représentation externe des données

Les représentations externes des données, dans les divers langages de programmation d'automates programmables, doivent se composer de libellés numériques, de cordons de caractères et de libellés de datation.

2.2.1 Libellés numériques

Il existe deux catégories de libellés numériques: les libellés entiers et les libellés réels. Un libellé numérique se définit comme un nombre décimal ou comme un nombre basé. Le nombre maximal de chiffres, pour chaque type de libellé numérique, doit être suffisant pour exprimer l'étendue entière et la précision des valeurs de tous les types de données qui sont représentés par le libellé dans une mise en oeuvre donnée.