



Programmable controllers

Part 3: Programming languages



This Australian Standard® was prepared by Committee IT-006, Industrial Process Measurement, Control and Automation. It was approved on behalf of the Council of Standards Australia on 14 November 2014.
This Standard was published on 19 December 2014.

The following are represented on Committee IT-006:

- Australian Computer Society
 - Australian Industry Group
 - Australian Petroleum Production and Exploration Association
 - Australian Safety Critical Systems Association
 - Consult Australia
 - Engineers Australia
 - Institute of Chemical Engineers Australia
 - Institute of Instrumentation, Control and Automation Australia
 - ISACA
 - Process Control Society
 - University of Queensland
 - Workplace Health and Safety Queensland
-

This Standard was issued in draft form for comment as DR AS IEC 61131.3:2014.

Standards Australia wishes to acknowledge the participation of the expert individuals that contributed to the development of this Standard through their representation on the Committee and through the public comment period.

Keeping Standards up-to-date

Australian Standards® are living documents that reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued.

Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments that may have been published since the Standard was published.

Detailed information about Australian Standards, drafts, amendments and new projects can be found by visiting www.standards.org.au

Standards Australia welcomes suggestions for improvements, and encourages readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to Standards Australia, GPO Box 476, Sydney, NSW 2001.

Australian Standard[®]

Programmable controllers

Part 3: Programming languages

Originated as AS 4168.3—1994.
Previous edition AS IEC 61131.3—2004.
Second edition 2014.

COPYRIGHT

© Standards Australia Limited

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968.

Published by SAI Global Limited under licence from Standards Australia Limited, GPO Box 476, Sydney, NSW 2001, Australia

ISBN 978 1 74342 931 0

PREFACE

This Standard was prepared by the Standards Australia Committee IT-006, Industrial Process Measurement, Control and Automation, to supersede AS IEC 61131.3—2004.

The objective of this Standard is to specify syntax and semantics of programming languages for programmable controllers as defined in AS IEC 61131.1.

This Standard should be read in conjunction with the other parts of the AS IEC 61131 series.

This Standard is identical with, and has been reproduced from, IEC 61131-3, Ed. 3.0 (2013), *Programmable controllers—Part 3: Programming languages*.

This edition is a fully compatible extension of the previous (2004) edition. The main extensions are new data types and conversion functions, references, name spaces and the object oriented features of classes and function blocks. For further details, see Annex B.

As this Standard is reproduced from an International Standard, the following applies:

- (a) In the source text ‘this part of IEC 61131’ should read ‘this Australian Standard’.
- (b) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>	<i>Australian Standard</i>
IEC	AS IEC
61131 Programmable controllers	61131 Programmable controllers
61131-1 Part 1: General information	61131.1 Part 1: General information
61131-5 Part 5: Communications	61131.5 Part 5: Communications

Only normative references that have been adopted as Australian or Australian/New Zealand Standards have been listed.

The terms ‘normative’ and ‘informative’ are used to define the application of the annex to which they apply. A normative annex is an integral part of a standard, whereas an informative annex is only for information and guidance.

CONTENTS

1	Scope	9
2	Normative references	9
3	Terms and definitions	9
4	Architectural models	18
4.1	Software model	18
4.2	Communication model	19
4.3	Programming model	20
5	Compliance	22
5.1	General	22
5.2	Feature tables	22
5.3	Implementer's compliance statement	22
6	Common elements	24
6.1	Use of printed characters	24
6.1.1	Character set	24
6.1.2	Identifiers	24
6.1.3	Keywords	24
6.1.4	Use of white space	25
6.1.5	Comments	25
6.2	Pragma	26
6.3	Literals – External representation of data	26
6.3.1	General	26
6.3.2	Numeric literals and string literals	26
6.3.3	Character string literals	28
6.3.4	Duration literal	29
6.3.5	Date and time of day literal	30
6.4	Data types	30
6.4.1	General	30
6.4.2	Elementary data types (BOOL, INT, REAL, STRING, etc.)	30
6.4.3	Generic data types	33
6.4.4	User-defined data types	34
6.5	Variables	47
6.5.1	Declaration and initialization of variables	47
6.5.2	Variable sections	49
6.5.3	Variable length ARRAY variables	51
6.5.4	Constant variables	53
6.5.5	Directly represented variables (%)	54
6.5.6	Retentive variables (RETAIN, NON_RETAIN)	56
6.6	Program organization units (POUs)	58
6.6.1	Common features for POUs	58
6.6.2	Functions	70
6.6.3	Function blocks	99
6.6.4	Programs	117
6.6.5	Classes	118

6.6.6	Interface	137
6.6.7	Object oriented features for function blocks	146
6.6.8	Polymorphism.....	152
6.7	Sequential Function Chart (SFC) elements	155
6.7.1	General	155
6.7.2	Steps.....	155
6.7.3	Transitions	157
6.7.4	Actions	160
6.7.5	Rules of evolution.....	168
6.8	Configuration elements.....	176
6.8.1	General	176
6.8.2	Tasks	180
6.9	Namespaces	186
6.9.1	General	186
6.9.2	Declaration	186
6.9.3	Usage.....	192
6.9.4	Namespace directive USING.....	192
7	Textual languages	195
7.1	Common elements.....	195
7.2	Instruction list (IL)	195
7.2.1	General	195
7.2.2	Instructions.....	195
7.2.3	Operators, modifiers and operands.....	196
7.2.4	Functions and function blocks.....	198
7.3	Structured Text (ST).....	201
7.3.1	General	201
7.3.2	Expressions.....	201
7.3.3	Statements	203
8	Graphic languages	208
8.1	Common elements.....	208
8.1.1	General	208
8.1.2	Representation of variables and instances.....	209
8.1.3	Representation of lines and blocks	211
8.1.4	Direction of flow in networks	212
8.1.5	Evaluation of networks	213
8.1.6	Execution control elements.....	214
8.2	Ladder diagram (LD)	215
8.2.1	General	215
8.2.2	Power rails	216
8.2.3	Link elements and states	216
8.2.4	Contacts	216
8.2.5	Coils.....	218
8.2.6	Functions and function blocks.....	219
8.2.7	Order of network evaluation.....	219
8.3	Function Block Diagram (FBD)	219
8.3.1	General	219
8.3.2	Combination of elements	219
8.3.3	Order of network evaluation.....	220
Annex A (normative)	Formal specification of the languages elements	221

Annex B (informative) List of major changes and extensions of the third edition.....	228
Bibliography.....	229
Figure 1 – Software model	18
Figure 2 – Communication model.....	20
Figure 3 – Combination of programmable controller language elements.....	21
Figure 4 – Implementer’s compliance statement (Example).....	23
Figure 5 – Hierarchy of the generic data types	34
Figure 6 – Initialization by literals and constant expressions (Rules).....	35
Figure 7 – Variable declaration keywords (Summary).....	50
Figure 8 – Usage of VAR_GLOBAL, VAR_EXTERNAL and CONSTANT (Rules).....	51
Figure 9 – Conditions for the initial value of a variable (Rules).....	57
Figure 10 – Formal and non-formal representation of call (Examples)	63
Figure 11 – Data type conversion rules – implicit and/or explicit (Summary)	67
Figure 12 – Supported implicit type conversions	68
Figure 13 – Usage of function block input and output parameters (Rules)	108
Figure 14 – Usage of function block input and output parameters (Illustration of rules)	109
Figure 15 – Standard timer function blocks – timing diagrams (Rules)	116
Figure 16 – Overview of inheritance and interface implementation	119
Figure 17 – Inheritance of classes (Illustration).....	128
Figure 18 – Interface with derived classes (Illustration).....	138
Figure 19 – Inheritance of interface and class (Illustration)	143
Figure 20 – Function block with optional body and methods (Illustration)	149
Figure 21 – Inheritance of function block body with SUPER() (Example).....	151
Figure 22 – ACTION_CONTROL function block – External interface (Summary).....	165
Figure 23 – ACTION_CONTROL function block body (Summary).....	166
Figure 24 – Action control (Example)	168
Figure 25 – SFC evolution (Rules)	174
Figure 26 – SFC errors (Example)	175
Figure 27 – Configuration (Example).....	177
Figure 28 – CONFIGURATION and RESOURCE declaration (Example)	180
Figure 29 – Accessibility using namespaces (Rules).....	189
Figure 30 – Common textual elements (Summary)	195
Table 1 – Character set	24
Table 2 – Identifiers	24
Table 3 – Comments.....	25
Table 4 – Pragma	26
Table 5 – Numeric literals	27
Table 6 – Character string literals	28
Table 7 – Two-character combinations in character strings	29
Table 8 – Duration literals.....	29
Table 9 – Date and time of day literals.....	30

Table 10 – Elementary data types	31
Table 11 – Declaration of user-defined data types and initialization	35
Table 12 – Reference operations	46
Table 13 – Declaration of variables	48
Table 14 – Initialization of variables	49
Table 15 – Variable-length <code>ARRAY</code> variables	52
Table 16 – Directly represented variables	54
Table 17 – Partial access of <code>ANY_BIT</code> variables	60
Table 18 – Execution control graphically using <code>EN</code> and <code>ENO</code>	65
Table 19 – Function declaration	72
Table 20 – Function call	74
Table 21 – Typed and overloaded functions	76
Table 22 – Data type conversion function	78
Table 23 – Data type conversion of numeric data types	80
Table 24 – Data type conversion of bit data types	82
Table 25 – Data type conversion of bit and numeric types	83
Table 26 – Data type conversion of date and time types	85
Table 27 – Data type conversion of character types	86
Table 28 – Numerical and arithmetic functions	87
Table 29 – Arithmetic functions	88
Table 30 – Bit shift functions	89
Table 31 – Bitwise Boolean functions	89
Table 32 – Selection functions ^d	90
Table 33 – Comparison functions	91
Table 34 – Character string functions	92
Table 35 – Numerical functions of time and duration data types	93
Table 36 – Additional functions of time data types <code>CONCAT</code> and <code>SPLIT</code>	94
Table 37 – Function for endianness conversion	98
Table 38 – Functions of enumerated data types	98
Table 39 – Validate functions	99
Table 40 – Function block type declaration	100
Table 41 – Function block instance declaration	104
Table 42 – Function block call	105
Table 43 – Standard bistable function blocks ^a	112
Table 44 – Standard edge detection function blocks	113
Table 45 – Standard counter function blocks	113
Table 46 – Standard timer function blocks	115
Table 47 – Program declaration	117
Table 48 – Class	120
Table 49 – Class instance declaration	122
Table 50 – Textual call of methods – Formal and non-formal parameter list	125
Table 51 – Interface	137
Table 52 – Assignment attempt	146

Table 53 – Object oriented function block	147
Table 54 – SFC step	156
Table 55 – SFC transition and transition condition	158
Table 56 – SFC declaration of actions	160
Table 57 – Step/action association	162
Table 58 – Action block.....	163
Table 59 – Action qualifiers.....	163
Table 60 – Action control features.....	168
Table 61 – Sequence evolution – graphical.....	169
Table 62 – Configuration and resource declaration	178
Table 63 – Task.....	182
Table 64 – Namespace	191
Table 65 – Nested namespace declaration options	192
Table 66 – Namespace directive <code>USING</code>	194
Table 67 – Parenthesized expression for IL language	197
Table 68 – Instruction list operators	197
Table 69 – Calls for IL language	199
Table 70 – Standard function block operators for IL language	201
Table 71 – Operators of the ST language.....	202
Table 72 – ST language statements	203
Table 73 – Graphic execution control elements.....	215
Table 74 – Power rails and link elements	216
Table 75 – Contacts.....	217
Table 76 – Coils.....	218

AUSTRALIAN STANDARD

Programmable controllers**Part 3:
Programming languages****1 Scope**

This part of IEC 61131 specifies syntax and semantics of programming languages for programmable controllers as defined in Part 1 of IEC 61131.

The functions of program entry, testing, monitoring, operating system, etc., are specified in Part 1 of IEC 61131.

This part of IEC 61131 specifies the syntax and semantics of a unified suite of programming languages for programmable controllers (PCs). This suite consists of two textual languages, Instruction List (IL) and Structured Text (ST), and two graphical languages, Ladder Diagram (LD) and Function Block Diagram (FBD).

An additional set of graphical and equivalent textual elements named Sequential Function Chart (SFC) is 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.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

ISO/IEC 10646:2012, *Information technology – Universal Coded Character Set (UCS)*

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

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61131-1 and the following apply.

3.1**absolute time**

combination of time of day and date information