IEC 61691-7

Edition 2.0 2025-06

INTERNATIONAL STANDARD

IEEE Std 1666™

Behavioural languages –
Part 7: SystemC® Language Reference Manual

ICS 25.040.01; 35.060

ISBN 978-2-8327-0467-7

IEC 61691-7:2025-06(en) IEEE Std 1666-2023



THIS PUBLICATION IS COPYRIGHT PROTECTED
Copyright © 2023 IEEE

All rights reserved. IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Inc. Unless otherwise specified, 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 IEC Central Office. Any questions about IEEE copyright should be addressed to the IEEE. Enquiries about obtaining additional rights to this publication and other information requests should be addressed to the IEC or your local IEC member National Committee.

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue
New York, NY 10016-5997
United States of America
stds.ipr@ieee.org
www.ieee.org

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC/IEEE publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews, graphical symbols and the glossary. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 500 terminological entries in English and French, with equivalent terms in 25 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

1. Overview ... 23

1.1 Scope ... 23
1.2 Purpose ... 23
1.3 Word usage ... 23
1.4 Subsets ... 24
1.5 Relationship with C++ standard ... 24
1.6 Guidance for readers ... 24

2. Normative references ... 26

3. Terminology and conventions used in this standard ... 27

3.1 Terminology ... 27
3.1.1 Shall, should, may, can ... 27
3.1.2 Implementation, application ... 27
3.1.3 Call, called from, derived from ... 27
3.1.4 Specific technical terms ... 27
3.2 Syntactical conventions ... 29
3.2.1 Implementation-defined ... 29
3.2.2 Disabled ... 29
3.2.3 Ellipsis (...) ... 29
3.2.4 Class names ... 29
3.2.5 Embolded text ... 30
3.3 Semantic conventions ... 30
3.3.1 Class definitions and the inheritance hierarchy ... 30
3.3.2 Function definitions and side-effects ... 30
3.3.3 Functions that use const char* as parameter ... 30
3.3.4 Functions whose return type is a reference or a pointer ... 31
3.3.5 Namespaces and internal naming ... 33
3.3.6 Non-compliant applications and errors ... 33
3.4 Notes and examples ... 33

4. Elaboration and simulation semantics ... 34

4.1 Overview ... 34
4.2 Elaboration ... 34
4.2.1 Overview ... 34
4.2.2 Instantiation ... 35
4.2.3 Process macros ... 36
4.2.4 Port binding and export binding ... 36
4.2.5 Setting the time resolution ... 37
4.3 Simulation ... 38
4.3.1 Overview ... 38
4.3.2 The scheduling algorithm ... 38
4.3.3 Initialization, cycles, pauses,and suspension in the scheduling algorithm ... 41
4.4 Running elaboration and simulation ... 42
4.4.1 Overview ... 42
4.4.2 Function declarations ... 42
4.4.3 sc_elab_and_sim ... 43
4.4.4 sc_argc and sc_argv ... 43

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

4.4.5 Running under application control using functions sc_main and sc_start ... 44
4.4.6 Running under control of the kernel ... 46
4.5 Elaboration and simulation callbacks ... 46
4.5.1 Overview ... 46
4.5.2 before_end_of_elaboration ... 47
4.5.3 end_of_elaboration ... 48
4.5.4 start_of_simulation ... 49
4.5.5 end_of_simulation ... 50
4.6 Other functions related to the scheduler ... 50
4.6.1 Function declarations ... 50
4.6.2 sc_pause ... 52
4.6.3 sc_suspend_all, sc_unsuspend_all, sc_unsuspendable, and sc_suspendable ... 53
4.6.4 sc_stop, sc_set_stop_mode, and sc_get_stop_mode ... 53
4.6.5 sc_time_stamp ... 54
4.6.6 sc_delta_count and sc_delta_count_at_current_time ... 55
4.6.7 sc_is_running ... 55
4.6.8 Functions to detect pending activity ... 56
4.6.9 sc_get_status ... 57
4.6.10 sc_stage_callback_if ... 58
4.6.11 sc_register_stage_callback ... 60
4.6.12 sc_unregister_stage_callback ... 60
4.6.13 Callbacks execution ... 60

5. Core language class definitions ... 61

5.1 Class header files ... 61
5.1.1 Overview ... 61
5.1.2 #include "systemc" ... 61
5.1.3 #include "systemc.h" ... 61
5.2 sc_module ... 62
5.2.1 Description ... 62
5.2.2 Class definition ... 62
5.2.3 Constraints on usage ... 65
5.2.4 kind ... 65
5.2.5 SC_MODULE ... 65
5.2.6 Constructors ... 66
5.2.7 SC_CTOR ... 66
5.2.8 SC_METHOD, SC_THREAD, SC_CTHREAD ... 67
5.2.9 SC_NAMED ... 68
5.2.10 Method process ... 68
5.2.11 Thread and clocked thread processes ... 69
5.2.12 Clocked thread processes ... 70
5.2.13 reset_signal_is and async_reset_signal_is ... 72
5.2.14 sensitive ... 73
5.2.15 dont_initialize ... 74
5.2.16 set_stack_size ... 75
5.2.17 next_trigger ... 75
5.2.18 wait ... 77
5.2.19 Positional port binding ... 78
5.2.20 before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation ... 80
5.2.21 get_child_objects and get_child_events ... 80
5.2.22 sc_gen_unique_name ... 81
5.2.23 sc_behavior and sc_channel ... 81

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

5.3 sc_module_name ... 82
5.3.1 Description ... 82
5.3.2 Class definition ... 82
5.3.3 Constraints on usage ... 83
5.3.4 Module hierarchy ... 83
5.3.5 Member functions ... 84
5.4 *sc_sensitive†* ... *85*
5.4.1 Description ... 85
5.4.2 Class definition ... 85
5.4.3 Constraints on usage ... 85
5.4.4 operator<< ... 85
5.5 sc_spawn_options and sc_spawn ... 86
5.5.1 Description ... 86
5.5.2 Class definition ... 86
5.5.3 Constraints on usage ... 88
5.5.4 Constructors ... 88
5.5.5 Member functions ... 88
5.5.6 sc_spawn ... 89
5.5.7 SC_FORK and SC_JOIN ... 92
5.6 sc_process_handle ... 93
5.6.1 Description ... 93
5.6.2 Class definition ... 93
5.6.3 Constraints on usage ... 95
5.6.4 Constructors ... 95
5.6.5 Member functions ... 95
5.6.6 Member functions for process control ... 98
5.6.7 sc_get_current_process_handle ... 114
5.6.8 sc_is_unwinding ... 115
5.7 sc_event_finder and sc_event_finder_t ... 116
5.7.1 Description ... 116
5.7.2 Class definition ... 116
5.7.3 Constraints on usage ... 116
5.8 sc_event_and_list and sc_event_or_list ... 118
5.8.1 Description ... 118
5.8.2 Class definition ... 118
5.8.3 Constraints and usage ... 119
5.8.4 Constructors, destructor, assignment ... 119
5.8.5 Member functions and operators ... 119
5.9 *sc_event_and_expr†* and *sc_event_or_expr†* ... 121
5.9.1 Description ... 121
5.9.2 Class definition ... 121
5.9.3 Constraints on usage ... 122
5.9.4 Operators ... 122
5.10 sc_event ... 123
5.10.1 Description ... 123
5.10.2 Class definition ... 123
5.10.3 Constraints on usage ... 124
5.10.4 Constructors, destructor, and event naming ... 124
5.10.5 Functions for naming and hierarchy traversal ... 125
5.10.6 notify and cancel ... 126
5.10.7 Event lists ... 127
5.10.8 None event ... 127
5.10.9 Multiple event notifications ... 127
5.11 sc_time ... 128

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

5.11.1 Description ... 128
5.11.2 Class definition ... 128
5.11.3 Constructors ... 129
5.11.4 Functions and operators ... 130
5.11.5 Time resolution ... 130
5.11.6 sc_max_time ... 131
5.11.7 SC_ZERO_TIME ... 131
5.12 sc_port ... 131
5.12.1 Description ... 131
5.12.2 Class definition ... 131
5.12.3 Template parameters ... 133
5.12.4 Constraints on usage ... 134
5.12.5 Constructors ... 135
5.12.6 kind ... 135
5.12.7 Named port binding ... 135
5.12.8 Member functions for bound ports and port-to-port binding ... 136
5.12.9 before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation ... 140
5.13 sc_export ... 140
5.13.1 Description ... 140
5.13.2 Class definition ... 140
5.13.3 Template parameters ... 141
5.13.4 Constraints on usage ... 141
5.13.5 Constructors ... 142
5.13.6 kind ... 142
5.13.7 Export binding ... 142
5.13.8 Member functions for bound exports and export-to-export binding ... 143
5.13.9 before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation ... 144
5.14 sc_interface ... 144
5.14.1 Description ... 144
5.14.2 Class definition ... 145
5.14.3 Constraints on usage ... 145
5.14.4 register_port ... 145
5.14.5 default_event ... 146
5.15 sc_prim_channel ... 147
5.15.1 Description ... 147
5.15.2 Class definition ... 147
5.15.3 Constraints on usage ... 148
5.15.4 Constructors, destructor, and hierarchical names ... 148
5.15.5 kind ... 149
5.15.6 request_update and update ... 149
5.15.7 async_attach_suspending and async_detach_suspending ... 150
5.15.8 next_trigger and wait ... 150
5.15.9 before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation ... 151
5.16 sc_object ... 152
5.16.1 Description ... 152
5.16.2 Class definition ... 152
5.16.3 Constraints on usage ... 153
5.16.4 Constructors and destructor ... 153
5.16.5 name, basename, and kind ... 154
5.16.6 print and dump ... 155
5.16.7 Functions for object hierarchy traversal ... 155

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

5.16.8 Member functions for attributes 157
5.16.9 get_hierarchy_scope 158
5.17 Hierarchical naming of objects and events 158
5.17.1 Overview 158
5.17.2 sc_hierarchical_name_exists 159
5.17.3 sc_register_hierarchical_name, sc_unregister_hierarchical_name 160
5.18 sc_attr_base 160
5.18.1 Description 160
5.18.2 Class definition 160
5.18.3 Member functions 161
5.19 sc_attribute 161
5.19.1 Description 161
5.19.2 Class definition 161
5.19.3 Template parameters 161
5.19.4 Member functions and data members 161
5.20 sc_attr_cltn 162
5.20.1 Description 162
5.20.2 Class definition 162
5.20.3 Constraints on usage 162
5.20.4 Iterators 162
5.21 sc_hierarchy_scope 163
5.21.1 Description 163
5.21.2 Class definition 163
5.21.3 Constraints on usage 163
5.21.4 Constructor 164
5.21.5 get_root 164

6. Predefined channel class definitions 165

6.1 sc_signal_in_if 165
6.1.1 Description 165
6.1.2 Class definition 165
6.1.3 Member functions 165
6.2 sc_signal_in_if<bool> and sc_signal_in_if<sc_dt::sc_logic> 166
6.2.1 Description 166
6.2.2 Class definition 166
6.2.3 Member functions 167
6.3 sc_signal_inout_if 167
6.3.1 Description 167
6.3.2 Class definition 167
6.3.3 Member functions 168
6.4 sc_signal 168
6.4.1 Description 168
6.4.2 Class definition 169
6.4.3 Template parameter T 169
6.4.4 Reading and writing signals 170
6.4.5 Constructors 171
6.4.6 register_port 171
6.4.7 Member functions for reading 172
6.4.8 Member functions for writing 172
6.4.9 Member functions for events 172
6.4.10 Diagnostic member functions 173
6.4.11 operator<< 173
6.5 sc_signal<bool,WRITER_POLICY> and sc_signal<sc_dt::sc_logic,WRITER_POLICY> .. 174

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

6.5.1 Description ... 174
6.5.2 Class definition ... 174
6.5.3 Member functions ... 176
6.6 sc_buffer ... 177
6.6.1 Description ... 177
6.6.2 Class definition ... 177
6.6.3 Constructors ... 177
6.6.4 Member functions ... 178
6.7 sc_clock ... 179
6.7.1 Description ... 179
6.7.2 Class definition ... 179
6.7.3 Characteristic properties ... 180
6.7.4 Constructors ... 180
6.7.5 write ... 180
6.7.6 Diagnostic member functions ... 181
6.7.7 before_end_of_elaboration ... 181
6.7.8 sc_in_clk ... 181
6.8 sc_in ... 181
6.8.1 Description ... 181
6.8.2 Class definition ... 181
6.8.3 Member functions ... 182
6.8.4 sc_trace ... 183
6.8.5 end_of_elaboration ... 183
6.9 sc_in<bool> and sc_in<sc_dt::sc_logic> ... 183
6.9.1 Description ... 183
6.9.2 Class definition ... 183
6.9.3 Member functions ... 185
6.10 sc_inout ... 186
6.10.1 Description ... 186
6.10.2 Class definition ... 186
6.10.3 Member functions ... 187
6.10.4 initialize ... 187
6.10.5 sc_trace ... 187
6.10.6 end_of_elaboration ... 188
6.10.7 Binding ... 188
6.11 sc_inout<bool> and sc_inout<sc_dt::sc_logic> ... 188
6.11.1 Description ... 188
6.11.2 Class definition ... 188
6.11.3 Member functions ... 190
6.12 sc_out ... 190
6.12.1 Description ... 190
6.12.2 Class definition ... 190
6.12.3 Member functions ... 191
6.13 sc_signal_resolved ... 191
6.13.1 Description ... 191
6.13.2 Class definition ... 191
6.13.3 Constructors ... 192
6.13.4 Resolution semantics ... 192
6.13.5 Member functions ... 193
6.14 sc_in_resolved ... 194
6.14.1 Description ... 194
6.14.2 Class definition ... 194
6.14.3 Member functions ... 195
6.15 sc_inout_resolved ... 195

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

6.15.1 Description ... 195
6.15.2 Class definition ... 195
6.15.3 Member functions ... 196
6.16 sc_out_resolved ... 196
6.16.1 Description ... 196
6.16.2 Class definition ... 196
6.16.3 Member functions ... 197
6.17 sc_signal_rv ... 197
6.17.1 Description ... 197
6.17.2 Class definition ... 197
6.17.3 Semantics and member functions ... 198
6.18 sc_in_rv ... 198
6.18.1 Description ... 198
6.18.2 Class definition ... 198
6.18.3 Member functions ... 199
6.19 sc_inout_rv ... 199
6.19.1 Description ... 199
6.19.2 Class definition ... 199
6.19.3 Member functions ... 200
6.20 sc_out_rv ... 200
6.20.1 Description ... 200
6.20.2 Class definition ... 200
6.20.3 Member functions ... 201
6.21 sc_fifo_in_if ... 201
6.21.1 Description ... 201
6.21.2 Class definition ... 201
6.21.3 Member functions ... 202
6.22 sc_fifo_out_if ... 202
6.22.1 Description ... 202
6.22.2 Class definition ... 202
6.22.3 Member functions ... 203
6.23 sc_fifo ... 204
6.23.1 Description ... 204
6.23.2 Class definition ... 204
6.23.3 Template parameter T ... 205
6.23.4 Constructors ... 205
6.23.5 register_port ... 205
6.23.6 Member functions for reading ... 206
6.23.7 Member functions for writing ... 206
6.23.8 The update phase ... 207
6.23.9 Member functions for events ... 207
6.23.10 Member functions for available values and free slots ... 207
6.23.11 Diagnostic member functions ... 207
6.23.12 operator<< ... 208
6.24 sc_fifo_in ... 208
6.24.1 Description ... 208
6.24.2 Class definition ... 209
6.24.3 Member functions ... 209
6.25 sc_fifo_out ... 209
6.25.1 Description ... 209
6.25.2 Class definition ... 210
6.25.3 Member functions ... 210
6.26 sc_mutex_if ... 212
6.26.1 Description ... 212

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

6.26.2 Class definition 212
6.26.3 Member functions 212
6.27 sc_mutex 212
6.27.1 Description 212
6.27.2 Class definition 213
6.27.3 Constructors 213
6.27.4 Member functions 213
6.28 sc_semaphore_if 214
6.28.1 Description 214
6.28.2 Class definition 214
6.28.3 Member functions 215
6.29 sc_semaphore 215
6.29.1 Description 215
6.29.2 Class definition 215
6.29.3 Constructors 215
6.29.4 Member functions 216
6.30 sc_event_queue 216
6.30.1 Description 216
6.30.2 Class definition 217
6.30.3 Constraints on usage 217
6.30.4 Constructors 217
6.30.5 kind 217
6.30.6 Member functions 218
6.31 sc_stub, sc_unbound, sc_tie 219

7. SystemC data types 220

7.1 Introduction 220
7.2 Common characteristics 222
7.2.1 Overview 222
7.2.2 Initialization and assignment operators 223
7.2.3 Precision of arithmetic expressions 223
7.2.4 Base class default word length 224
7.2.5 Word length 225
7.2.6 Bit-select 225
7.2.7 Part-select 226
7.2.8 Concatenation 227
7.2.9 Reduction operators 228
7.2.10 Integer conversion 228
7.2.11 String input and output 229
7.2.12 Conversion of application-defined types in integer expressions 229
7.3 String literals 230
7.4 *sc_value_base†* *231*
7.4.1 Description 231
7.4.2 Class definition 232
7.4.3 Constraints on usage 232
7.4.4 Member functions 232
7.5 Limited-precision integer types 233
7.5.1 Type definitions 233
7.5.2 sc_int_base 233
7.5.3 sc_uint_base 238
7.5.4 sc_int 242
7.5.5 sc_uint 245
7.5.6 Bit-selects 247

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

7.5.7 Part-selects ... 251
7.6 Finite-precision integer types ... 256
7.6.1 Type definitions ... 256
7.6.2 Constraints on usage ... 256
7.6.3 sc_signed ... 257
7.6.4 sc_unsigned ... 263
7.6.5 sc_bigint ... 269
7.6.6 sc_biguint ... 271
7.6.7 Bit-selects ... 273
7.6.8 Part-selects ... 276
7.7 Integer concatenations ... 281
7.7.1 Description ... 281
7.7.2 Class definition ... 281
7.7.3 Constraints on usage ... 283
7.7.4 Assignment operators ... 283
7.7.5 Implicit type conversion ... 283
7.7.6 Explicit type conversion ... 284
7.7.7 Other member functions ... 284
7.8 Generic base proxy class ... 284
7.8.1 Description ... 284
7.8.2 Class definition ... 284
7.8.3 Constraints on usage ... 284
7.9 Logic and vector types ... 285
7.9.1 Type definitions ... 285
7.9.2 sc_logic ... 285
7.9.3 sc_bv_base ... 289
7.9.4 sc_lv_base ... 295
7.9.5 sc_bv ... 300
7.9.6 sc_lv ... 302
7.9.7 Bit-selects ... 304
7.9.8 Part-selects ... 308
7.9.9 Concatenations ... 313
7.10 Fixed-point types ... 320
7.10.1 Overview ... 320
7.10.2 Fixed-point representation ... 320
7.10.3 Fixed-point type conversion ... 322
7.10.4 Fixed-point data types ... 322
7.10.5 Fixed-point expressions and operations ... 323
7.10.6 Bit and part selection ... 327
7.10.7 Variable-precision fixed-point value limits ... 327
7.10.8 Fixed-point word length and mode ... 327
7.10.9 Conversions to character string ... 330
7.10.10 Finite word-length effects ... 331
7.10.11 sc_fxnum ... 354
7.10.12 sc_fxnum_fast ... 358
7.10.13 sc_fxval ... 363
7.10.14 sc_fxval_fast ... 368
7.10.15 sc_fix ... 372
7.10.16 sc_ufix ... 375
7.10.17 sc_fix_fast ... 378
7.10.18 sc_ufix_fast ... 381
7.10.19 sc_fixed ... 384
7.10.20 sc_ufixed ... 386
7.10.21 sc_fixed_fast ... 388

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

7.10.22sc_ufixed_fast 390
7.10.23Bit-selects 393
7.10.24Part-selects 396
7.11 Contexts 402
7.11.1 Overview 402
7.11.2 sc_length_param 402
7.11.3 sc_length_context 403
7.11.4 sc_fxtype_params 404
7.11.5 sc_fxtype_context 407
7.11.6 sc_fxcast_switch 408
7.11.7 sc_fxcast_context 409
7.12 Control of string representation 410
7.12.1 Description 410
7.12.2 Class definition 410
7.12.3 Functions 410

8. SystemC utilities 411

8.1 Trace files 411
8.1.1 Overview 411
8.1.2 Class definition and function declarations 411
8.1.3 sc_trace_file 411
8.1.4 sc_create_vcd_trace_file 412
8.1.5 sc_close_vcd_trace_file 412
8.1.6 sc_write_comment 412
8.1.7 sc_trace 412
8.2 sc_report 414
8.2.1 Description 414
8.2.2 Class definition 414
8.2.3 Constraints on usage 415
8.2.4 sc_verbosity 415
8.2.5 sc_severity 416
8.2.6 Copy constructor and assignment 416
8.2.7 Member functions 416
8.3 sc_report_handler 417
8.3.1 Description 417
8.3.2 Class definition 417
8.3.3 Constraints on usage 419
8.3.4 sc_actions 419
8.3.5 report 420
8.3.6 set_actions 420
8.3.7 stop_after 421
8.3.8 get_count 422
8.3.9 Verbosity level 422
8.3.10 suppress and force 423
8.3.11 set_handler 423
8.3.12 get_new_action_id 424
8.3.13 sc_interrupt_here and sc_stop_here 424
8.3.14 get_cached_report and clear_cached_report 424
8.3.15 set_log_file_name and get_log_file_name 425
8.4 sc_exception 425
8.4.1 Description 425
8.4.2 Class definition 425
8.5 sc_vector 426

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

8.5.1 Description ... 426
8.5.2 Class definition ... 426
8.5.3 Constraints on usage ... 429
8.5.4 Constructors and destructors ... 429
8.5.5 init and create_element ... 430
8.5.6 Incremental additions to sc_vector during elaboration phase ... 432
8.5.7 kind, size, get_elements ... 432
8.5.8 operator[] and at ... 432
8.5.9 Iterators ... 433
8.5.10 bind ... 433
8.5.11 sc_assemble_vector ... 435
8.6 Utility functions ... 437
8.6.1 Function declarations ... 437
8.6.2 sc_abs ... 437
8.6.3 sc_max ... 438
8.6.4 sc_min ... 438
8.6.5 Version and copyright ... 438

9. Overview of TLM-2.0 and compliance with TLM-2.0 standard ... 440

9.1 Overview ... 440
9.2 Compliance with the TLM-2.0 standard ... 441

10. Introduction to TLM-2.0 ... 442

10.1 Background ... 442
10.2 Transaction-level modeling, use cases, and abstraction ... 442
10.3 Coding styles ... 443
10.3.1 Overview ... 443
10.3.2 Untimed coding style ... 443
10.3.3 Loosely-timed coding style and temporal decoupling ... 444
10.3.4 Synchronization in loosely-timed models ... 445
10.3.5 Approximately-timed coding style ... 445
10.3.6 Characterization of loosely-timed and approximately-timed coding styles ... 446
10.3.7 Switching between loosely-timed and approximately-timed modeling ... 446
10.3.8 Cycle-accurate modeling ... 446
10.3.9 Blocking versus non-blocking transport interfaces ... 446
10.3.10 Use cases and coding styles ... 447
10.4 Initiators, targets, sockets, and transaction bridges ... 447
10.5 DMI and debug transport interfaces ... 449
10.6 Combined interfaces and sockets ... 449
10.7 Namespaces ... 450
10.8 Header files and version numbers ... 450
10.8.1 Overview ... 450
10.8.2 Software version information ... 450
10.8.3 Definitions ... 450
10.8.4 Rules ... 451

11. TLM-2.0 core interfaces ... 452

11.1 Overview ... 452
11.2 Transport interfaces ... 452
11.2.1 Overview ... 452
11.2.2 Blocking transport interface ... 452

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

11.2.3 Non-blocking transport interface 457
11.2.4 Timing annotation with the transport interfaces 465
11.2.5 Migration path from TLM-1 468
11.3 Direct memory interface 468
11.3.1 Introduction 468
11.3.2 Class definition 469
11.3.3 get_direct_mem_ptr 470
11.3.4 template argument and tlm_generic_payload class 471
11.3.5 tlm_dmi class 472
11.3.6 invalidate_direct_mem_ptr 475
11.3.7 DMI versus transport 475
11.3.8 DMI and temporal decoupling 476
11.3.9 Optimization using a DMI hint 476
11.4 Debug transport interface 476
11.4.1 Introduction 476
11.4.2 Class definition 477
11.4.3 TRANS template argument and tlm_generic_payload class 477
11.4.4 Rules 477

12. TLM-2.0 global quantum 480

12.1 Introduction 480
12.2 Header file 480
12.3 Class definition 480
12.4 tlm_global_quantum 481

13. Combined TLM-2.0 interfaces and sockets 482

13.1 Combined interfaces 482
13.1.1 Introduction 482
13.1.2 Class definition 482
13.2 Initiator and target sockets 483
13.2.1 Introduction 483
13.2.2 Class definition 483
13.2.3 tlm_base_socket_if 487
13.2.4 tlm_base_initiator_socket_b and tlm_base_target_socket_b 487
13.2.5 tlm_base_initiator_socket and tlm_base_target_socket 488
13.2.6 tlm_initiator_socket and tlm_target_socket 490

14. TLM-2.0 generic payload 493

14.1 Introduction 493
14.2 Extensions and interoperability 493
14.2.1 Overview 493
14.2.2 Use the generic payload directly, with ignorable extensions 494
14.2.3 Define a new protocol traits class containing a typedef for tlm_generic_payload 495
14.2.4 Define a new protocol traits class and a new transaction type 495
14.3 Generic payload attributes and member functions 496
14.4 Class definition 496
14.5 Generic payload memory management 498
14.6 Constructors, assignment, and destructor 502
14.7 Default values and modifiability of attributes 502
14.8 Option attribute 504
14.9 Command attribute 506

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

14.10 Address attribute 506
14.11 Data pointer attribute 507
14.12 Data length attribute 508
14.13 Byte enable pointer attribute 508
14.14 Byte enable length attribute 509
14.15 Streaming width attribute 510
14.16 DMI allowed attribute 510
14.17 Response status attribute 511
14.17.1 Overview 511
14.17.2 The standard error response 512
14.18 Endianness 515
14.18.1 Introduction 515
14.18.2 Rules 516
14.19 Helper functions to determine host endianness 518
14.19.1 Introduction 518
14.19.2 Definition 518
14.19.3 Rules 519
14.20 Helper functions for endianness conversion 519
14.20.1 Introduction 519
14.20.2 Definition 520
14.20.3 Rules 520
14.21 Generic payload extensions 522
14.21.1 Introduction 522
14.21.2 Rationale 522
14.21.3 Extension pointers, objects and transaction bridges 523
14.21.4 Rules 523

15. TLM-2.0 base protocol and phases 529

15.1 Phases 529
15.1.1 Introduction 529
15.1.2 Class definition 529
15.1.3 Rules 530
15.2 Base protocol531
15.2.1 Introduction 531
15.2.2 Class definition 532
15.2.3 Base protocol phase sequences 532
15.2.4 Permitted phase transitions 534
15.2.5 Ignorable phases 537
15.2.6 Base protocol timing parameters and flow control 539
15.2.7 Base protocol rules concerning timing annotation 543
15.2.8 Base protocol rules concerning b_transport 544
15.2.9 Base protocol rules concerning request and response ordering 544
15.2.10 Base protocol rules for switching between b_transport and nb_transport 545
15.2.11 Other base protocol rules 546
15.2.12 Summary of base protocol transaction ordering rules 546
15.2.13 Guidelines for creating base-protocol-compliant components 547

16. TLM-2.0 utilities 550

16.1 Overview 550
16.2 Convenience sockets 550
16.2.1 Introduction 550
16.2.2 Simple sockets 551

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

16.2.3 Tagged simple sockets 557
16.2.4 Multi-sockets 560
16.3 Quantum keeper 566
16.3.1 Introduction 566
16.3.2 Header file 567
16.3.3 Class definition 567
16.3.4 General guidelines for processes using temporal decoupling 567
16.3.5 tlm_quantumkeeper 568
16.4 Payload event queue 570
16.4.1 Introduction 570
16.4.2 Header file 571
16.4.3 Class definition 571
16.4.4 Rules 571
16.5 Instance-specific extensions 572
16.5.1 Introduction 572
16.5.2 Header file 573
16.5.3 Class definition 573

17. TLM-1 message passing interface and analysis ports 575

17.1 Overview 575
17.2 Put, get, peek, and transport interfaces 575
17.2.1 Description 575
17.2.2 Class definition 575
17.2.3 Blocking versus non-blocking interfaces 577
17.2.4 Blocking interface methods 578
17.2.5 Non-blocking interface methods 578
17.2.6 Argument passing and transaction lifetime 579
17.2.7 Constraints on the transaction data type 580
17.3 TLM-1 fifo interfaces 580
17.3.1 Description 580
17.3.2 Class definition 580
17.3.3 Member functions 581
17.4 tlm_fifo 582
17.4.1 Description 582
17.4.2 Class definition 582
17.4.3 Template parameter T 583
17.4.4 Constructors and destructor 583
17.4.5 Member functions 584
17.4.6 Delta cycle semantics 585
17.5 Analysis interface and analysis ports 587
17.5.1 Overview 587
17.5.2 Class definition 587
17.5.3 Rules 588

Annex A (informative) Glossary 590

Annex B (informative) Introduction to SystemC 607

Annex C (informative) Deprecated features 611

Annex D (informative) Changes between IEEE Std 1666-2011 and IEEE Std 1666-2023 613
Annex E (informative) IEEE List of participants 617

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

BEHAVIOURAL LANGUAGES –

Part 7: SystemC ® Language Reference Manual

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

   IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards. Use of an IEEE standard is wholly voluntary.IEEE documents are made available for use subject to important notices and legal disclaimers (see https://standards.ieee.org/ipr/disclaimers.html for more information).

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

IEC 61691-7/ IEEE Std 1666 was processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
1666 (2023)	91/2024/FDIS	91/2035/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

IEEE Std 1666™-2023
(Revision of IEEE Std 1666-2011)

IEEE Standard for Standard SystemC® Language Reference Manual

Developed by

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 5 June 2023

IEEE SA Standards Board

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Abstract: SystemC® is defined in this standard. SystemC is an ISO standard C++ class library for system and hardware design for use by designers and architects who need to address complex systems that are a hybrid between hardware and software. This standard provides a precise and complete definition of the SystemC class library so that a SystemC implementation can be developed with reference to this standard alone. The primary audiences for this standard are the implementors of the SystemC class library, the implementors of tools supporting the class library, and the users of the class library.

Keywords: C++, computer languages, digital systems, discrete event simulation, electronic design automation, electronic system level, electronic systems, embedded software, fixed-point, hardware description language, hardware design, hardware verification, IEEE 1666™, SystemC, system modeling, system-on-chip, transaction level

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents**.

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the IEEE SA myProject system. An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; https://www.copyright.com/. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore or contact IEEE. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the IEEE SA Patent Policy.

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Introduction

This introduction is not part of IEEE Std 1666-2023, IEEE Standard for Standard SystemC® Language Reference Manual.

This document defines SystemC, which is a C++ class library.

As the electronics industry builds more complex systems involving large numbers of components including software, there is an increasing need for a modeling language that can manage the complexity and size of these systems. SystemC provides a mechanism for managing this complexity with its facility for modeling hardware and software together at multiple levels of abstraction. This capability is not available in traditional hardware description languages.

Stakeholders in SystemC include Electronic Design Automation (EDA) companies who implement SystemC class libraries and tools, integrated circuit (IC) suppliers who extend those class libraries and use SystemC to model their intellectual property, and end users who use SystemC to model their systems.

Before the publication of this standard, SystemC was defined by an open-source, proof-of-concept C++ library, also known as *the reference simulator*, available from the Accellera Systems Initiative. In the event of discrepancies between the behavior of the reference simulator and statements made in this standard, this standard shall be taken to be definitive.

This standard is not intended to serve as a user's guide or to provide an introduction to SystemC. Readers requiring a SystemC tutorial or information on the intended use of SystemC should consult the Accellera Systems Initiative Web site (www.accellera.org) to locate the many books and training classes available.

Acknowledgments

Contributed updates to IEEE Std 1666-2011 from the Accellera Systems Initiative SystemC Language Working Group are acknowledged.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

IEEE Standard for Standard SystemC® Language Reference Manual

1. Overview

1.1 Scope

This standard defines SystemC® with Transaction Level Modeling (TLM) as an ISO standard C++ class library for system and hardware design.[1]

1.2 Purpose

The general purpose of this standard is to provide a C++-based standard for designers and architects who need to address complex systems that are a hybrid between hardware and software.

The specific purpose of this standard is to provide a precise and complete definition of the SystemC class library including a TLM library so that a SystemC implementation can be developed with reference to this standard alone. This standard is not intended to serve as a user's guide or to provide an introduction to SystemC, but it does contain useful information for end users.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).[2, 3]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

[1] SystemC® is a registered trademark of the Accellera Systems Initiative.

[2] The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

[3] The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals is *able to*).

See also Clause 3 for information about other terminology, syntactical conventions, semantic conventions, and notes and examples used in this standard.

1.4 Subsets

It is anticipated that tool vendors will create implementations that support only a subset of this standard or that impose further constraints on the use of this standard. Such implementations are not fully compliant with this standard but may nevertheless claim partial compliance with this standard and may use the name SystemC. See also 9.2 for a description of TLM-2.0-compliance.

1.5 Relationship with C++ standard

This standard is closely related to the C++ programming language and adheres to the terminology used in ISO/IEC 14882:2017 (i.e., C++17).[4] As a result, C++17 is the baseline for SystemC. Implementations and tools may support other C++ versions (i.e., greater than 2017); however, they shall support at least C++17 to claim compliance with IEEE Std 1666-2023. This standard does not seek to restrict the usage of C++17; a SystemC application may use any of the facilities provided by C++17, which in turn may use any of the facilities provided by C. However, where the facilities provided by this standard are used, they shall be used in accordance with the rules and constraints set out in this standard.

This standard defines the public interface to the SystemC class library and the constraints on how those classes may be used. The SystemC class library may be implemented in any manner whatsoever, provided only that the obligations imposed by this standard are honored.

A C++ class library may be extended using the mechanisms provided by the C++ language. Implementors and users are free to extend SystemC in this way, provided that they do not violate this standard.

NOTE—It is possible to create a well-formed C++ program that is legal according to the C++ programming language standard but that violates this standard. An implementation is not obliged to detect every violation of this standard.[5]

1.6 Guidance for readers

Readers who are not entirely familiar with SystemC should start with the introduction to SystemC in Annex B, which provides a brief informal summary of the subject intended to aid in the understanding of the normative definitions given throughout this standard. Such readers may also find it helpful to scan the examples embedded in the normative definitions and to review the glossary in Annex A.

In addition to the overview and normative references in Clause 1 and Clause 2, respectively, all readers should pay close attention to Clause 3, where certain terminology and conventions used in this standard are explained. An understanding of the terminology defined in Clause 3 is necessary for a precise interpretation of this standard.

Clause 4 defines the behavior of the SystemC kernel and is central to an understanding of SystemC. The semantic definitions given in the subsequent clauses detailing the individual classes are built on the foundations laid in Clause 4.

[4] Information on references can be found in Clause 2.

[5] Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

Clause 5 through Clause 8 define the public interface to the SystemC class library. The following information is listed for each class:

a) A C++ source code listing of the class definition
b) A statement of any constraints on the use of the class and its members
c) A statement of the semantics of the class and its members
d) For certain classes, a description of functions, typedefs, and macros associated with the class
e) Informative examples illustrating both typical and atypical uses of the class

Readers should bear in mind that the primary obligation of a tool vendor is to implement the abstract semantics defined in Clause 4, using the framework and constraints provided by the class definitions starting in Clause 5.

Clause 9 through Clause 16 define the public interface to the TLM-2.0 class library, including the classes of the interoperability layer and the utilities.

Clause 17 defines the TLM-1 message passing interface, including **tlm_fifo** and analysis ports.

Annex A is a glossary giving informal descriptions of the terms used in this standard.

Annex B is intended to aid the reader in the understanding of the structure and intent of the SystemC class library.

Annex C lists the deprecated features, that is, features that were present in version 2.0.1 of the Open SystemC Initiative (OSCI) open source proof-of-concept SystemC implementation but are not part of this standard.

Annex D lists the changes between IEEE Std 1666-2011 and IEEE Std 1666-2023.

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used; therefore, each referenced document is cited in text, and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

This standard shall be used in conjunction with the following publication:

ISO/IEC 14882:2017, Programming Languages — C++.[6]

[6] ISO/IEC publications are available from the International Organization for Standardization (https://www.iso.org), the International Electrotechnical Commission (https://www.iec.ch), and the American National Standards Institute (https://www.ansi.org).

Published by IEC under licence from IEEE. © 2023 IEEE. All rights reserved.