

**INTERNATIONAL  
STANDARD**

**ISO/IEC  
13961  
IEEE  
Std 1596**

First edition  
2000-07

---

---

**Information technology –  
Scalable Coherent Interface (SCI)**



Reference number  
ISO/IEC 13961:2000(E)  
IEEE Std 1596, 1998 Edition

**Abstract:** The scalable coherent interface (SCI) provides computer-bus-like services but, instead of a bus, uses a collection of fast point-to-point unidirectional links to provide the far higher throughput needed for high-performance multiprocessor systems. SCI supports distributed, shared memory with optional cache coherence for tightly coupled systems, and message-passing for loosely coupled systems. Initial SCI links are defined at 1 Gbyte/s (16-bit parallel) and 1 Gb/s (serial). For applications requiring modular packaging, an interchangeable module is specified along with connector and power. The packets and protocols that implement transactions are defined and their formal specification is provided in the form of computer programs. In addition to the usual read-and-write transactions, SCI supports efficient multiprocessor lock transactions. The distributed cache-coherence protocols are efficient and can recover from an arbitrary number of transmission failures. SCI protocols ensure forward progress despite multiprocessor conflicts (no deadlocks or starvation).

**Keywords:** bus architecture, bus standard, cache coherence, distributed memory, fiber optic, interconnect, I/O system, link, mesh, multiprocessor, network, packet protocol, ring, seamless distributed computer, shared memory, switch, transaction set.

---

The Institute of Electrical and Electronics Engineers, Inc.  
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1998 by the Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. First published in 1998.

ISBN 2-8318-5375-3

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

## CONTENTS

	Page
FOREWORD .....	12
Clause	
1 Introduction.....	19
1.1 Document structure.....	19
1.2 SCI overview .....	20
1.2.1 Scope and directions.....	20
1.2.2 The SCI approach.....	21
1.2.3 System configurations.....	22
1.2.4 Initial physical models .....	23
1.2.5 SCI node model .....	24
1.2.6 Architectural parameters .....	25
1.2.7 A common CSR architecture .....	25
1.2.8 Structure of the specification.....	26
1.3 Interconnect topologies.....	26
1.3.1 Bridged systems .....	26
1.3.2 Scalable systems .....	27
1.3.3 Interconnected systems .....	27
1.3.4 Backplane rings .....	27
1.3.5 Interconnected rings .....	28
1.3.6 Rectangular grid interconnects.....	29
1.3.7 Butterfly switches.....	30
1.3.8 Vendor-dependent switches .....	31
1.4 Transactions.....	31
1.4.1 Packet formats.....	32
1.4.2 Input and output queues .....	33
1.4.3 Request and response queues.....	34
1.4.4 Switch queues.....	36
1.4.5 Subactions.....	36
1.4.6 Remote transactions (through agents).....	39
1.4.7 Move transactions.....	41
1.4.8 Broadcast moves .....	42
1.4.9 Broadcast passing by agents .....	43
1.4.10 Transaction types.....	44
1.4.11 Message passing .....	45
1.4.12 Global clocks .....	45
1.4.13 Allocation protocols.....	46
1.4.14 Queue allocation.....	47
1.5 Cache coherence.....	49
1.5.1 Interconnect constraints .....	49
1.5.2 Distributed directories .....	49
1.5.3 Standard optimizations.....	50
1.5.4 Future extensions .....	50
1.5.5 TLB purges .....	53

Clause	Page
1.6 Reliability, availability, and support (RAS).....	54
1.6.1 RAS overview .....	54
1.6.2 Autoconfiguration.....	54
1.6.3 Control and status registers .....	54
1.6.4 Transmission-error detection and isolation.....	55
1.6.5 Error containment .....	55
1.6.6 Hardware fault retry (ringlet-local, physical layer option) .....	56
1.6.7 Software fault recovery (end-to-end) .....	56
1.6.8 System debugging .....	57
1.6.9 Alternate routing .....	57
1.6.10 Online replacement.....	57
2 References, glossary, and notation .....	58
2.1 Normative references.....	58
2.2 Conformance levels .....	58
2.3 Terms and definitions .....	59
2.4 Bit and byte ordering.....	66
2.5 Numerical values .....	68
2.6 C code.....	68
3 Logical protocols and formats .....	68
3.1 Packet formats.....	68
3.1.1 Packet types .....	68
3.2 Send and echo packet formats.....	69
3.2.1 Request-send packet format .....	69
3.2.2 Request-echo packet format .....	72
3.2.3 Response-send packet.....	74
3.2.4 Standard status codes .....	76
3.2.5 Response-echo packet format.....	78
3.2.6 Interconnect-affected fields .....	79
3.2.7 Init packets .....	80
3.2.8 Cyclic redundancy code (CRC) .....	81
3.2.9 Parallel 16-bit CRC calculations.....	82
3.2.10 CRC stomping.....	84
3.2.11 Idle symbols.....	85
3.3 Logical packet encodings.....	86
3.3.1 Flag coding .....	86
3.4 Transaction types .....	89
3.4.1 Transaction commands .....	89
3.4.2 Lock subcommands .....	92
3.4.3 Unaligned DMA transfers .....	94
3.4.4 Aligned block-transfer hints.....	95
3.4.5 Move transactions.....	97
3.4.6 Global time synchronization .....	98
3.5 Elastic buffers.....	99
3.5.1 Elasticity models.....	99
3.5.2 Idle-symbol insertions .....	100
3.5.3 Idle-symbol deletions .....	101

Clause	Page
3.6 Bandwidth allocation.....	101
3.6.1 Fair bandwidth allocation .....	102
3.6.2 Setting ringlet priority.....	104
3.6.3 Bandwidth partitioning.....	106
3.6.4 Types of transmission protocols.....	108
3.6.5 Pass-transmission protocol .....	108
3.6.6 Low-transmission protocol.....	111
3.6.7 Idle insertions .....	114
3.6.8 High-transmission protocol.....	114
3.7 Queue allocation.....	116
3.7.1 Queue reservations.....	116
3.7.2 Multiple active sends.....	118
3.7.3 Unfair reservations.....	119
3.7.4 Queue-selection protocols.....	119
3.7.5 Re-send priorities.....	119
3.8 Transaction errors.....	120
3.8.1 Requester timeouts (response-expected packets) .....	120
3.8.2 Time-of-death timeout (optional, all nodes) .....	120
3.8.3 Responder-processing errors .....	122
3.9 Transmission errors .....	123
3.9.1 Error isolation .....	123
3.9.2 Scrubber maintenance .....	125
3.9.3 Producer-detected errors .....	126
3.9.4 Consumer-detected errors.....	128
3.10 Address initialization.....	129
3.10.1 Transaction addressing.....	129
3.10.2 Reset types.....	131
3.10.3 Unique node identifiers .....	132
3.10.4 Ringlet initialization.....	133
3.10.5 Simple-subset ringlet resets.....	135
3.10.6 Ringlet resets.....	135
3.10.7 Ringlet clears (optional) .....	137
3.10.8 Inserting initialization packets .....	138
3.10.9 Address initialization .....	139
3.11 Packet encoding .....	140
3.11.1 Common encoding features (L18) .....	140
3.11.2 Parallel encoding with 18 signals (P18).....	141
3.11.3 Serial encoding with 20-bit symbols (S20).....	141
3.12 SCI-specific control and status registers .....	144
3.12.1 SCI transaction sets.....	144
3.12.2 SCI resets.....	145
3.12.3 SCI-dependent fields within standard CSRs .....	145
3.12.4 SCI-dependent CSRs.....	148
3.12.5 SCI-dependent ROM.....	151
3.12.6 Interrupt register formats.....	155
3.12.7 Interleaved logical addressing.....	157

Clause	Page
4 Cache-coherence protocols.....	158
4.1 Introduction.....	158
4.1.1 Objectives.....	158
4.1.2 SCI transaction components .....	158
4.1.3 Physical addressing .....	159
4.1.4 Coherence directory overview .....	159
4.1.5 Memory and cache tags .....	160
4.1.6 Instruction-execution model .....	161
4.1.7 Coherence document structure .....	162
4.2 Coherence update sequences.....	163
4.2.1 List prepend.....	163
4.2.2 List-entry deletion .....	165
4.2.3 Update actions.....	167
4.2.4 Cache-line locks .....	167
4.2.5 Stable sharing lists.....	168
4.3 Minimal-set coherence protocols.....	171
4.3.1 Sharing-list updates .....	171
4.3.2 Cache fetching.....	171
4.3.3 Cache rollouts.....	173
4.3.4 Instruction-execution model .....	174
4.4 Typical-set coherence protocols.....	175
4.4.1 Sharing-list updates .....	175
4.4.2 Read-only fetch.....	175
4.4.3 Read-write fetch.....	177
4.4.4 Data modifications .....	178
4.4.5 Mid and head deletions .....	179
4.4.6 DMA reads and writes .....	181
4.4.7 Instruction-execution model .....	183
4.5 Full-set coherence protocols .....	184
4.5.1 Full-set option summary.....	184
4.5.2 CLEAN-list creation.....	184
4.5.3 Sharing-list additions .....	185
4.5.4 Cache washing .....	187
4.5.5 Cache flushing .....	189
4.5.6 Cache cleansing .....	191
4.5.7 Pairwise sharing .....	192
4.5.8 Pairwise-sharing faults .....	196
4.5.9 QOLB sharing .....	197
4.5.10 Cache-access properties.....	200
4.5.11 Instruction-execution model .....	201
4.6 C-code naming conventions.....	202
4.7 Coherent read and write transactions.....	203
4.7.1 Extended mread transactions.....	204
4.7.2 Cache cread and cwrite64 transactions.....	205
4.7.3 Smaller tag sizes .....	206

Clause	Page
5 C-code structure .....	207
5.1 Node structure .....	207
5.1.1 Signals within a node .....	207
5.1.2 Packet transfers among node components .....	208
5.1.3 Transfer-cloud components .....	208
5.2 A node's linc component .....	210
5.2.1 A linc's subcomponents .....	210
5.2.2 A linc's elastic buffer .....	212
5.2.3 Other linc components .....	213
5.3 Other node components .....	213
5.3.1 A node's core component .....	213
5.3.2 A node's memory component .....	213
5.3.3 A node's exec component .....	214
5.3.4 A node's proc component .....	215
6 Physical layers .....	216
6.1 Type 1 module .....	217
6.1.1 Module characteristics .....	217
6.1.2 Module compatibility considerations .....	217
6.1.3 Module size .....	218
6.1.4 Warpage, bowing, and deflection .....	224
6.1.5 Cooling .....	225
6.1.6 Connector .....	226
6.1.7 Power and ground connection .....	227
6.1.8 Pin allocation for backplane parallel 18-signal encoding .....	229
6.1.9 Slot-identification signals .....	231
6.2 Type 18-DE-500 signals and power control .....	232
6.2.1 SCI differential signals .....	233
6.2.2 Status lines .....	233
6.2.3 Serial Bus signals .....	233
6.2.4 Signal levels and skew .....	233
6.2.5 Power-conversion control .....	236
6.3 Type 18-DE-500 module extender cable .....	238
6.4 Type 18-DE-500 cable-link .....	240
6.5 Serial interconnection .....	242
6.5.1 Serial interface Type 1-SE-1250, single-ended electrical .....	243
6.5.2 Optical interface, fiber-optic signal type 1-FO-1250 .....	249
6.5.3 Test methods .....	252
Annex A (informative) Ringlet initialization .....	254
Annex B (informative) SCI design models .....	257
B.1 Fast counters .....	257
B.2 Translation-lookaside-buffer coherence .....	257
B.3 Coherent lock models .....	261
B.4 Coherence-performance models .....	263
Bibliography .....	265

	Page
Figure 1 – Physical-layer alternatives .....	23
Figure 2 – SCI node model .....	24
Figure 3 – 64-bit-fixed addressing .....	25
Figure 4 – Bridged systems .....	26
Figure 5 – Backplane rings .....	28
Figure 6 – Interconnected rings .....	29
Figure 7 – 2-D processor grids .....	29
Figure 8 – Butterfly ringlets .....	30
Figure 9 – Switch interface .....	31
Figure 10 – Subactions .....	32
Figure 11 – Send-packet format, simplified .....	32
Figure 12 – Responder queues .....	34
Figure 13 – Logical requester/responder queues .....	35
Figure 14 – Paired request and response queues .....	35
Figure 15 – Basic SCI bridge, paired request and response queues .....	36
Figure 16 – Local transaction components .....	37
Figure 17 – Local transaction components (busied by responder) .....	38
Figure 18 – Remote transaction components .....	40
Figure 19 – Remote move-transaction components .....	41
Figure 20 – Broadcast starts .....	43
Figure 21 – Broadcast resumes .....	43
Figure 22 – Transaction formats .....	44
Figure 23 – Bandwidth partitioning .....	46
Figure 24 – Resource bottlenecks .....	47
Figure 25 – Queue allocation avoids starvation .....	48
Figure 26 – Distributed cache tags .....	49
Figure 27 – Request combining .....	52
Figure 28 – Binary tree .....	52
Figure 29 – TLB purging .....	53
Figure 30 – Hardware fault-retry sequence .....	56
Figure 31 – Software fault-retry on coherent data .....	57
Figure 32 – Big-endian packet notation .....	67
Figure 33 – Big-endian register notation .....	67
Figure 34 – Send- and echo-packet formats .....	69
Figure 35 – Request-packet format .....	70
Figure 36 – Request-packet symbols .....	70
Figure 37 – Request-echo packet format .....	72
Figure 38 – Response-packet format .....	74
Figure 39 – Response-packet symbols .....	75
Figure 40 – Response-echo packet format .....	78
Figure 41 – Initialization-packet format .....	80
Figure 42 – Initialization-packet format example ( <i>companyId</i> -based <i>uniqueId</i> value) .....	81
Figure 43 – Serialized implementation of 16-bit CRC .....	82
Figure 44 – Parallel CRC check .....	84
Figure 45 – Remote transaction components (local request-send damaged) .....	85
Figure 46 – Logical idle-symbol encoding .....	85
Figure 47 – Flag framing convention .....	86
Figure 48 – Logical send- and init-packet framing convention .....	87
Figure 49 – Logical echo-packet framing convention .....	87
Figure 50 – Logical sync-packet framing convention .....	88
Figure 51 – Logical <i>abort</i> -packet framing convention .....	88
Figure 52 – Selected-byte reads and writes .....	91
Figure 53 – Simplified lock model .....	92
Figure 54 – Selected-byte locks (quadlet access) .....	93
Figure 55 – Selected-byte locks (octlet access) .....	94
Figure 56 – Expected DMA read transfers .....	94
Figure 57 – Expected DMA write transfers .....	95
Figure 58 – DMA block-transfer model .....	96
Figure 59 – Time-sync on SCI .....	98
Figure 60 – Elasticity model .....	99



	Page
Figure 61 – Input-synchronizer model.....	100
Figure 62 – Idle-symbol insertion.....	100
Figure 63 – Idle-symbol deletion.....	101
Figure 64 – Fair bandwidth allocation.....	103
Figure 65 – Increasing ringlet priority.....	105
Figure 66 – Restoring ringlet priority.....	105
Figure 67 – Idle-symbol creation, fair-only node.....	106
Figure 68 – Idle-symbol creation, unfair-capable node.....	107
Figure 69 – Idle consumption, fair-only node.....	107
Figure 70 – Idle consumption, unfair-capable node.....	108
Figure 71 – Pass-transmission model (fair-only node).....	109
Figure 72 – Pass-transmission enabled.....	109
Figure 73 – Pass-transmission active.....	110
Figure 74 – Pass-transmission recovery.....	110
Figure 75 – Low/high-transmission model.....	111
Figure 76 – Low-transmission enabled.....	111
Figure 77 – Low-transmission active.....	112
Figure 78 – Low/high-transmission recovery.....	113
Figure 79 – Low/high-transmission debt repayment.....	113
Figure 80 – Low/high-transmission idle insertion.....	114
Figure 81 – High-transmission enabled.....	115
Figure 82 – Consumer send-packet queue reservations.....	116
Figure 83 – A/B age labels.....	118
Figure 84 – Response timeouts (request and no response).....	120
Figure 85 – Time-of-death discards.....	121
Figure 85 – Packet life-cycle intervals.....	121
Figure 87 – Time-of-death generation model.....	122
Figure 88 – Responder's address-error processing.....	122
Figure 89 – Response timeouts (request and no response).....	123
Figure 90 – Error-logging registers.....	124
Figure 91 – Scrubber maintenance functions.....	125
Figure 92 – Detecting lost low-go bits.....	126
Figure 93 – Producer's address-error processing.....	127
Figure 94 – Producer's echo-timeout processing.....	127
Figure 95 – Producer fatal-error recovery (optional).....	128
Figure 96 – Consumer error recovery.....	129
Figure 97 – SCI (64-bit fixed) addressing.....	129
Figure 98 – Forms of node resets.....	132
Figure 99 – Receiver synchronization and scrubber selection.....	134
Figure 100 – Reset-closure generates idle symbols.....	134
Figure 100 – Idle-closure injects go-bits in idles.....	134
Figure 101 – Initialization states.....	136
Figure 103 – Initialization states (clear option).....	137
Figure 104 – Output symbol sequence during initialization.....	138
Figure 105 – Insert-multiplexer model.....	139
Figure 106 – Nodelds after ringlet initialization and monarch selection.....	139
Figure 107 – Nodelds after emperor selection, final address assignments.....	140
Figure 108 – Flag framing convention.....	141
Figure 109 – S20 symbol encoding.....	142
Figure 110 – S20 symbol decoding.....	143
Figure 111 – S20 sync-packet encoding.....	143
Figure 112 – NODE_IDS register.....	145
Figure 113 – STATE_CLEAR fields.....	146
Figure 114 – SPLIT_TIMEOUT register-pair format.....	147
Figure 115 – ARGUMENT register-pair format.....	147
Figure 115 – CLOCK_STROBE_THROUGH format (offset 112).....	148
Figure 117 – ERROR_COUNT register (offset 384).....	149
Figure 118 – SYNC_INTERVAL register (offset 512).....	149
Figure 119 – SAVE_ID register (offset 520).....	150
Figure 120 – SLOT_ID register (offset 524).....	150

	Page
Figure 121 – SCI ROM format (bus_info_block).....	151
Figure 122 – ROM format, CsrOptions.....	152
Figure 123 – ROM format, LincOptions.....	153
Figure 124 – ROM format, MemoryOptions.....	154
Figure 125 – ROM format, CacheOptions .....	155
Figure 126 – DIRECTED_TARGET format.....	156
Figure 127 – Logical-to-physical address translation .....	157
Figure 128 – SCI transaction components .....	159
Figure 129 – Distributed sharing-list directory.....	160
Figure 130 – SCI coherence tags (64-byte line, 64K nodes) .....	161
Figure 131 – Prepend to ONLYP_DIRTY (pairwise capable).....	163
Figure 132 – Memory <i>mread</i> and cache-extended <i>cread</i> components .....	164
Figure 133 – Deletion of head (and exclusive) entry .....	165
Figure 134 – Cache <i>cwrite64</i> and memory-extended <i>mread</i> components .....	166
Figure 135 – ONLY_DIRTY list creation (minimal set) .....	171
Figure 136 – GONE list additions (minimal set) .....	172
Figure 137 – FRESH list additions (minimal set).....	172
Figure 138 – Only-entry deletions.....	173
Figure 139 – Tail-entry deletions .....	174
Figure 140 – FRESH list creation.....	175
Figure 141 – FRESH addition to FRESH list.....	176
Figure 142 – FRESH addition to DIRTY list .....	176
Figure 143 – DIRTY addition to FRESH list .....	177
Figure 144 – DIRTY addition to DIRTY list.....	177
Figure 145 – Head purging others .....	178
Figure 146 – ONLY_FRESH list conversion.....	179
Figure 147 – HEAD_FRESH list conversion.....	179
Figure 148 – Mid-entry deletions .....	180
Figure 149 – Head-entry deletions.....	180
Figure 150 – Robust ONLY_DIRTY deletions .....	181
Figure 151 – Checked DMA reads.....	181
Figure 152 – Checked DMA write (memory FRESH).....	182
Figure 153 – Checked DMA write (memory GONE).....	183
Figure 154 – CLEAN list creation.....	184
Figure 155 – FRESH addition to CLEAN/DIRTY list.....	185
Figure 156 – CLEAN addition to FRESH list .....	186
Figure 157 – CLEAN addition to CLEAN/DIRTY list.....	186
Figure 158 – Washing DIRTY sharing lists (prepend conflict) .....	188
Figure 159 – Flushing a FRESH list.....	190
Figure 160 – Flushing a GONE list .....	191
Figure 161 – Cleansing DIRTY sharing lists (prepend conflict) .....	192
Figure 162 – Pairwise-sharing transitions .....	193
Figure 163 – Prepending to pairwise list (HEAD_EXCL) .....	194
Figure 164 – Prepending to pairwise list (HEAD_STALE0) .....	195
Figure 165 – Two stale copies, head is valid .....	196
Figure 166 – Two stale copies, tail is valid .....	197
Figure 167 – Enqolb prepending to QOLB-locked list.....	198
Figure 168 – Deqolb tail-deletion on QOLB sharing list.....	199
Figure 169 – QOLB usage .....	199
Figure 170 – Basic mread/mwrite request.....	204
Figure 171 – Memory-access response .....	204
Figure 172 – Extended coherent memory read request.....	205
Figure 173 – Cache cread and cwrite64 requests .....	206
Figure 174 – Cache cread and cwrite64 responses .....	206
Figure 175 – Linc and component signals.....	207
Figure 176 – Linc and component queues .....	208
Figure 177 – One node's transfer-cloud model .....	209
Figure 178 – The linc packet queues .....	210
Figure 179 – Node interface structure .....	211
Figure 180 – Elasticity model .....	212

	Page
Figure 181 – A memory component's packet queues .....	214
Figure 182 – An exec component's packet queues .....	215
Figure 183 – A proc component's packet queues .....	215
Figure 184 – Type 1 module and a typical subrack .....	217
Figure 185 – Module board .....	219
Figure 186 – Module injector/ejector and top and bottom shielding .....	220
Figure 187 – Front panel arrangement, module shielding and clearances .....	221
Figure 188 – Top view of subrack .....	222
Figure 189 – Front view of subrack, left end .....	223
Figure 190 – Front view of subrack, top left detail .....	224
Figure 191 – Module power and ESD connections .....	228
Figure 192 – Backplane power pinout .....	230
Figure 193 – Backplane signal pinout .....	230
Figure 194 – Slot-position backplane wiring .....	232
Figure 195 – ECL signal voltage limits .....	234
Figure 196 – Basic timing .....	235
Figure 197 – SCI power-distribution model .....	236
Figure 198 – SCI power-control signal timing .....	237
Figure 199 – Type 18-DE-500 module extender cable .....	238
Figure 200 – Arrangement of module extender cable and connector .....	238
Figure 201 – Arrangement of module extender power cable and connector .....	239
Figure 202 – Cable-link and module signal connections contrasted .....	240
Figure 203 – Pinout of outgoing cable-link connector .....	240
Figure 204 – Pinout of incoming cable-link connector .....	241
Figure 205 – Generic eye mask .....	244
Figure 206 – Line driver with transformer isolation .....	247
Figure 207 – Line driver with capacitive coupling .....	248
Figure 208 – Receiver with transformer isolation and cable equalization .....	248
Figure 209 – Receiver with capacitive isolation and cable equalization .....	249
Figure A.1 – Simple reset .....	255
Figure A.2 – Simple reset states .....	256
Figure B.1 – Simple thru-counter implementation .....	257
Figure B.2 – Direct-register TLB-purge interlock .....	259
Figure B.3 – Coherent-TLB-purge interlock .....	260
Figure B.4 – Enqueueing messages .....	263
Figure B.5 – Dequeueing messages .....	263
Table 1 – Packet types .....	68
Table 2 – Phase field for send packets .....	71
Table 3 – Phase field for nonbusied echoes .....	73
Table 4 – Phase field for busied echoes .....	73
Table 5 – <i>status.sStat</i> status summary codes .....	76
Table 6 – Serial CRC-16 implementation .....	82
Table 7 – Parallel implementation of 16-bit CRC .....	83
Table 8 – Response-expected-subaction commands (read, write, and lock) .....	89
Table 9 – Responseless-subaction commands (move) .....	90
Table 10 – Event- and response-subaction commands .....	90
Table 11 – Subcommand values for Lock4 and Lock8 .....	92
Table 12 – Noncoherent block-transfer hints .....	97
Table 13 – Defined SCI nodeId addresses .....	130
Table 14 – Additional SCI transaction types .....	144
Table 15 – Initial nodeId values .....	146
Table 16 – Never-implemented CSR registers .....	147
Table 17 – Physical standard description .....	151
Table 18 – Interleave-control bits .....	158
Table 19 – Memory and cache update actions .....	167
Table 20 – Stable and semistable memory-tag states .....	168
Table 21 – Stable cache-tag states .....	169
Table 22 – Stable sharing lists .....	170

	Page
Table 23 – MinimalExecute Routines.....	174
Table 24 – TypicalExecute Routines.....	184
Table 25 – Readable cache states.....	200
Table 26 – FullExecute Routines .....	202
Table 27 – Coherent transaction summary .....	203
Table 28 – Module-connector part numbers.....	226
Table 29 – Backplane-fixed-connector part numbers .....	226
Table 30 – Power-connection summary .....	227
Table 31 – Main characteristics of ECL signals for SCI.....	235
Table 32 – Cable module-like connector part number .....	239
Table 33 – Cable backplane-like connector part numbers .....	239
Table 34 – Device cable-link connector (right-angle pins).....	242
Table 35 – Device cable-link connector (straight pins).....	242
Table 36 – Cable cable-link connector (sockets).....	242
Table 37 – Electrical signals at ETX .....	244
Table 38 – Electrical eye at ETX .....	245
Table 39 – Electrical signals at ERX.....	245
Table 40 – Electrical eye at ERX .....	245
Table 41 – Estimated maximum cable lengths .....	246
Table 42 – Optical eye at OTX .....	250
Table 43 – Optical eye at ORX .....	250
Table 44 – General optical requirements .....	250
Table 45 – Maximum laser spectral width .....	251
Table 46 – Typical connector properties .....	252
Table 47 – Loss budget.....	252

## **INFORMATION TECHNOLOGY – SCALABLE COHERENT INTERFACE (SCI)**

### **FOREWORD**

- 1) ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 13961 was prepared by subcommittee 26: Microprocessor systems, of ISO/IEC joint technical committee 1: Information technology.

Annexes A and B are for information only.

**IEEE Standards** documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute.

The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

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. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, 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.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.
---

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

## Foreword to IEEE Std 1596, 1998 Edition

[This foreword is not a part of ISO/IEC 13961:2000, Information technology – Scalable Coherent Interface (SCI).]

The demand for more processing power continues to increase, and apparently has no limit. One can usefully saturate the resources of any computer so easily by merely specifying a finer mesh or higher resolution for the solution of some physical problem (hydrodynamics, for example), that engineers and scientists are desperate for enormously larger computers.

To get this kind of computing power, it seems necessary to use a large number of processors cooperatively. Because of the propagation delays introduced when signals cross chip boundaries, the fastest uniprocessor may be on one chip before long. Pipelining and similar large-mainframe tricks are already used extensively on single-chip processors. Vector processors help, but are hard to use efficiently in many applications. Multiprocessors communicating by message passing work well for some applications, but not for all. The shared-memory multiprocessor looks like the best strategy for the future, but a great deal of work will be needed to develop software to use it efficiently.

It is important to support both the shared-memory and the message-passing models efficiently (and at the same time) in order to support optimal software for a wide range of problems, especially for a system that dynamically allocates processors and perhaps changes its configuration depending on the nature of its load.

SCI started from an attempt to increase the bandwidth of a backplane bus past the limits set by backplane physics in order to meet the needs of new generations of processor chips, some of which can single-handedly saturate the fastest buses. We soon learned that we had to abandon the bus structure to achieve our goals.

Backplane performance is limited by physics (distributed capacitances and the speed of light) and by a bus's one-at-a-time nature, an inherent bottleneck. To gain performance far beyond what buses and backplanes can do, one needs better signaling techniques and the concurrent use of many signaling paths.

Rather than using bused backplane wires, SCI is based on point-to-point interconnect technology. This design approach eliminates many of the physics problems and results in much higher speeds. SCI in effect simulates a bus, providing the bus services one expects (and more) without using buses.

## Committee Membership

The specification has been developed with the combined efforts of many volunteers. The following is a list of those who were members of the Working Group while the draft and final specification were compiled:

**David B. Gustavson, Chair**

**David V. James, Vice Chair**

Nagi Aboulenein	Emil N. Hahn	Phil Ponting
Knut Alnes	Horst Halling	Steve Quinton
Robert H. Appleby	Craig Hansen	Jean F. Renardy
Kurt Baty	Marit Jenssen	Randy Rettberg
Amir Behroozi	Rajeev Jog	Morten Schanke
David L. Black	Svein Erik Johansen	Gene Schramm
Andre Bogaerts	Sverre Johansen	James L. Schroeder
Paul Borrill	Ross Johnson	Tim Scott
Patrick Boyle	Anatol Kaganovich	Donald Senzig
David Brearley, Jr.	Alain Kagi	Gurindar Sohi
Charles Brill	Hans Karlsson*	Robert K. Southard
Haakon Bugge	Tom Knight	Joanne Spiller
Jan Buytaert	Michael J. Koster	Paul Sweazey
Jay Cantrell	Ernst Kristiansen	Lorne Temes
Mike Carlton	Stein Krogdahl	Manu Thapar
Fred L. Chong	Ralph Lachenmaier	John Theus
Graham Connolly	Branko Leskovar	Mike van Brunt
James R.(Bob) Davis	Dieter Linnhofer	Phil Vukovic
W. Kenneth Dawson	Robert McLaren	Anthony Waitz
Stephen R. Deiss	Mark Mellinger	Richard Walker
Gary Demos	Svein Moholt	Steve Ward
Roberto Divia	Viggy Mokkarala	Carl Warren*
Gregg Donley	John Moussouris	Steinar Wenaas
Wayne Downer	Hans Muller	Mike Wenzel
Guy Fedorkow	Klaus D. Muller	Richard J. Westmore
Peter Fenner	Ellen Munthe-Kaas	Wilson Whitehead
David Ford	Russell Nakano	Hans Wiggers
Stein Gjessing	Tom Nash	Mark Williams
Torstein Gleditsch	Steve Nelson	Philip Woest
James Goodman	Julian Olyansky	S. Y. Wong
Robert J. Greiner	Chris Parkman	Ken Wratten
Charles Grimsdale	Dan Picker	Chu-Sun Yen

\*deceased



The following persons were on the balloting committee that approved this document for submission to the IEEE Standards Board:

M. R. Aaron	David Hawley	Paul Rosenberg
Scott Akers	Phil Huelson	Carl Schriedekamp
Ray S. Alderman	Zoltan R. Hunor	James L. Schroeder
John Allen	Edgar Jacques	Don Senzig
Knut Alnes	David V. James	Philip Shutt
Richard P. Ames	Kenneth Jansen	Michael R. Sitzer
Bjorn Bakka	Rajeev Jog	Gurindar Sohi
David M. Barnum	Sverre Johansen	Robert K. Southard
Kurt Baty	Ross Johnson	Joanne Spiller
Harrison A. Beasley	Jack R. Johnson	David Stevenson
Amir Behroozi	Anatol Kaganovich	Robert Stewart
Janos Biri	Christopher Koehle	Paul Sweazey
David Black	Michael J. Koster	Daniel Tabak
William P. Blase	Ernst H. Kristiansen	Daniel Tarrant
Andre Bogaerts	Ralph Lachenmaier	Lorne Temes
W. C. Brantley	Glen Langdon, Jr.	Manu Thapar
David Brearley, Jr.	Gerry Laws	Michael G. Thompson
Haakon Bugge	Minsuk Lee	Chris Thomson
Kim Clohessy	Branko Leskovar	Joseph P. Trainor
Graham Connolly	Anthony G. Lubowe	Robert Tripi
Jonathan C. Crowell	Svein Moholt	Robert J. Voigt
W. Kenneth Dawson	James M. Moidel	Phil Vukovic
Stephen Deiss	James D. Mooney	Yoshiaki Wakimura
Dante Del Corso	Klaus-Dieter Mueller	Richard Walker
Stephen L. Diamond	Ellen Munthe-Kaas	Eike Waltz
Jean-Jacques Dumont	Cuong Nguyen	Carl Warren*
William P. Evertz	J. D. Nicoud	Richard J. Westmore
Guy Federkow	Dan O Connor	Hans A. Wiggers
Timothy R. Feldman	Mira Pauker	Mark Williams
Peter Fenner	Donald Pavlovich	Andrew Wilson
Gordon Force	Thomas Pittman	Joel Witt
Stein Gjessing	Steve Quinton	Ken Wratten
Andy Glew	Richard Rawson	David L. Wright
Patrick Gonia	Steven Ray	Chu Yen
James Goodman	Randy Rettburg	Oren Yuen
Charles Grimsdale	Hans Roosli	Janusz Zalewski

\*deceased

When the IEEE Standards Board approved this standard on 19 March 1992, it had the following membership:

**Marco W. Migliaro**, Chair

**Donald C. Loughry**, Vice Chair

**Andrew G. Salem**, Secretary

Dennis Bodson

Donald N. Heirman

T. Don Michael\*

Paul L. Borrill

Ben C. Johnson

John L. Rankine

Clyde Camp

Walter J. Karplus

Wallace S. Read

Donald C. Fleckenstein

Ivor N. Knight

Ronald H. Reimer

Jay Forster\*

Joseph Koepfinger\*

Gary S. Robinson

David F. Franklin

Irving Kolodny

Martin V. Schneider

Ramiro Garcia

D. N. Jim Logothetis

Terrance R. Whittemore

Thomas L. Hannan

Lawrence V. McCall

Donald W. Zipse

\*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Fernando Aldana

Satish K. Aggarwal

James Beall

Richard B. Engelman

Stanley Warshaw

This standard was approved by the American National Standards Institute on 23 October 1992. It was reaffirmed by IEEE in 1998.

# INFORMATION TECHNOLOGY – SCALABLE COHERENT INTERFACE (SCI)

## 1 Introduction

### 1.1 Document structure

This International Standard describes a communication protocol that provides the services required of a modern computer bus, but at far higher performance levels than any bus could attain. Packet protocols on unidirectional point-to-point transmission links emulate a sophisticated bus without incurring the inherent bus physics or bus contention problems.

This International Standard is partitioned into clauses that serve several distinct purposes:

**Clause 1: Introduction** provides background for understanding the Scalable Coherent Interface (SCI) protocols, and may be skipped by those already familiar with these concepts. The descriptions in this clause are somewhat simplified, and should not be considered part of the SCI specification.

**Clause 2: References, glossary and notation** defines the terminology used within this standard and lists references that are required for implementing the standard.

**Clause 3: Logical protocols and formats** defines the packets and protocols that implement transactions (like reads and writes) between SCI nodes. This clause uses text and figures as introductory material, to establish a frame of reference for the formal specification.

**Clause 4: Cache-coherence protocols** provides background information for understanding the protocols used by two or more SCI nodes to maintain coherence between cached copies of shared data. The coherence protocols contain many options. This clause describes the minimal subset of these protocols, a typical set of options that are likely to be implemented, and also the full set of protocols.

**Clause 5: C-code structure** explains the structure of the C code that defines the logical (packet symbol processing) and cache-coherence protocols. The precise specifications of the logical-level packet protocols and the cache-coherence protocols, which involve a large number of state-transition details, are expressed in C code because it is difficult to state them unambiguously in English, and so that they can be tested thoroughly under simulation.

**Clause 6: Physical layers** defines a mechanical package and several physical links that may be used to implement the logical protocols. This clause uses text and figures to specify the mechanical and electrical characteristics of several physical links.

**Annexes A and B:** These annexes describe other system-related concepts that have influenced the design of this standard. These may be useful for understanding the rationale behind some of the SCI design decisions.

**Bibliography** provides a variety of references that may be useful for understanding the terminology, notation, or concepts discussed within this standard.

**C code:** The C code is published as a text file on an IBM-format diskette. This was done for the convenience both of the casual reader of this standard, who will not delve into the details of the C code, and also of the serious user, who will wish to understand the C code thoroughly, executing it on a computer. Though the C code takes precedence over this International Standard in case of inconsistency, this International Standard provides considerable explanation and illustration to help develop an intuitive understanding that will make the C code more comprehensible.

## 1.2 SCI overview

### 1.2.1 Scope and directions

*Purpose:* To define an interface standard for very high-performance multiprocessor systems that supports a coherent shared-memory model scalable to systems with up to 64 K nodes. This standard is to facilitate assembly of processor, memory, I/O, and bus adaptor cards from multiple vendors into massively parallel systems with throughputs ranging up to more than  $10^{12}$  operations per second.

*Scope:* This standard will encompass two levels of interface, defining operation over distances less than 10 m. The *physical* layer will specify electrical, mechanical, and thermal characteristics of connectors and cards. The *logical* level will describe the address space, data transfer protocols, cache coherence mechanisms, synchronization primitives, *control* and status registers, and initialization and error recovery facilities.

The preceding statements were those submitted to and approved by the IEEE Standards Board as the definition of the SCI project. These goals have been met and exceeded: support for message-passing was added, and the operating distance is not limited to 10 m. (The intent of that limitation was to make clear that this is not yet-another Local Area Network.)

The real distinction between SCI and a network has more to do with the memory-access-based model SCI uses and the distributed cache-coherence model.

The practical operating distance depends more on the throughput and performance needed than on any absolute limit built into the specification. Very long links would yield unacceptable performance for many users (but perhaps not all).

In particular, the fibre-optic physical layer can extend the SCI paradigm over distances long enough to link a computer to its I/O devices, or to link several nearby processors. No arbitrary length limit would be appropriate, but practical considerations including the throughput requirements and the cost of transmitters and receivers will set the lengths that people consider useful.

A very-high-priority goal was that SCI be cost-effective for small systems as well as for the massively parallel ones mentioned in the purpose statement above. SCI's low pin count and simple ring implementation make medium-performance, few-processor systems easier to build with SCI than with bused backplane systems; a two-layer backplane should be sufficient, and three layers should be enough to support the optional geographical addressing mechanism. The SCI interface, complete with transceivers, fits into a single IC package that includes much of the logic needed to support the cache-coherence protocols. This economy for small systems leads to the expectation that SCI processor boards will be built in high volume, making them inexpensive enough to be assembled in large numbers for building supercomputers at low cost.

SCI also simplifies the construction of reliable systems. SCI Type 1 modules are well protected against electrostatic discharge and electromagnetic interference, and can be safely inserted while the remainder of the system remains powered. SCI supports live insertion and withdrawal by using a single supply voltage (with on-board conversion as needed) and staggered pin lengths in the connector to guarantee safe sequencing. Note, however, that system software plays an important role in live insertion or removal of a module because the resources provided by that module have to be allocated and deallocated appropriately.

In systems where several modules share a ringlet, the removal of one module interrupts all communication via that ringlet, so the resources on those modules also have to be deallocated. A similar situation arises in any system that may have multiple processors resident on one field-replaceable board: all have to be deallocated when any one is replaced. The system software for handling the deallocation and reallocation of these resources is outside SCI's scope.

Although SCI does not provide fault tolerance directly in its low-level protocols, it does provide the support needed for implementing fault-tolerant operation in software. With this recovery software, the SCI coherence protocols are robust and can recover from an arbitrary number of detected transmission failures (packets that are lost or corrupted).

The SCI paradigm removes the limits that bus structures place on throughput, but its latency is of course limited by the speed of signal propagation (less than the speed of light). Ever-increasing throughput can be expected as technology improves, but the organization of hardware and software will have to take into account the relatively constant latency (delay between request and response), which is proportional to the physical size of the system.

The last generation of buses approached the ultimate limits of performance, leading to the concept of an ultimate standard. However, the initially defined SCI physical layers are likely just the first of a series of implementations having higher or lower performance levels. The 1 Gbyte/s link speed specified for the initial ECL/copper-backplane implementation was chosen based on a combination of marketing and engineering considerations. From a marketing point of view, it was necessary to define a territory that did not disturb the markets for present 32-bit standards or present networks, and from an engineering point of view this link speed was near the edge of what available signalling technology and integrated circuit technology could support.

New technologies, such as better cables, connectors, transceivers; IC packages with more pins or higher power-dissipation capabilities; or faster ICs, could make it practical or desirable to implement SCI on new physical-layer standards. Such standards, with different link widths or bit rates, will be developed from time to time. However, packet formats and higher level coherence protocols will be the same across all these physical implementations. That should make the problem of interfacing one SCI system to another relatively simple – SCI already includes the necessary mechanisms to cope easily with speed differences.

## 2 References, glossary, and notation

### 2.1 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

EIA IS-64 (1991), *2 mm Two-Part Connectors for Use with Printed Boards and Backplanes* <sup>3)</sup>

IEC 60793-1, *Optical fibres – Part 1: Generic specification* <sup>4)</sup>

IEC 60793-2, *Optical fibres – Part 2: Product specifications*

IEEE Std 1301-1991, *IEEE Standard for a Metric Equipment Practice for Microcomputers – Coordination Document* (ANSI) <sup>5)</sup>

IEEE Std 1301.1-1991, *IEEE Standard for a Metric Equipment Practice for Microcomputers – Convection-Cooled with 2 mm Connectors* (ANSI)

ISO/IEC 13213:1994 [ANSI/IEEE Std 1212, 1994 Edition], *Information technology – Microprocessor systems – Control and Status Registers (CSA) Architecture for microcomputer buses* <sup>6)</sup>

ISO/IEC 9899:1990, *Programming languages – C*

---

<sup>3)</sup> EIA publications are available from Global Engineering, 1990 M Street NW, Suite 400, Washington, DC, 20036, USA.

<sup>4)</sup> IEC publications are available from IEC Customer Service Centre, Case postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42<sup>nd</sup> Street, 13<sup>th</sup> Floor, New York, NY 10036, USA.

<sup>5)</sup> IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

<sup>6)</sup> ISO publications are available from the ISO Central Secretariat, Case postale 56, 1 rue de Varembé, CH-1211 Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the American National Standards Institute.