

Specification for

ControlNet —

Part 10: Object library

ICS 25.040.40; 35.100.70

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

Committees responsible for this Draft for Development

The preparation of this Draft for Development was entrusted to Technical Committee AMT/7, Monitoring and control aspects of AMT, upon which the following bodies were represented:

- ABB Kent plc
- Cegelec Industrial Controls Ltd.
- Department of Trade and Industry
- Fisher-Rosemount Ltd.
- Foxboro Great Britain
- Institution of Electrical Engineers
- Measurement Technology Ltd.
- Phoenix Contact Ltd.
- Process Automation and Computer Systems Ltd.
- Rockwell Automation Ltd.
- Schneider Ltd.
- Siemens Automation Ltd.
- University of Warwick
- Wago Ltd.

This Draft for Development, having been prepared under the direction of the Advanced Manufacturing Technology Assembly, was published under the authority of the Standards Board and comes into effect on 15 November 1997

© BSI 07-1999

The following BSI reference relates to the work on this publication:
Committee reference AMT/7

ISBN 0 580 28677 0

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
Foreword	xiv
0 Introduction	1
1 Identity Object (class code = 0x01)	1
1.1 Scope	1
1.2 Attributes	2
1.3 Common services	6
1.4 Get_Attributes_All response	7
1.5 Object-specific services	8
1.6 Behaviour	8
2 Message Router (class code = 0x02)	11
2.1 Scope	11
2.2 Attributes	11
2.3 Common services	12
2.4 Get_Attributes_All response	13
2.5 Object-specific services	13
2.6 Behaviour	13
2.7 Service request	14
2.8 Service response	14
3 DeviceNet Object (class code = 0x03)	14
4 Assembly Object (class code = 0x04)	14
4.1 Scope	14
4.2 Revision history	14
4.3 Attributes	14
4.4 Common services	16
4.5 Object-specific services	16
4.6 Behaviour	16
4.7 Static assemblies	17
4.8 Dynamic assemblies	18
5 Connection Object (class code = 0x05)	20
6 Connection Manager Object (class code = 0x06)	20
6.1 Scope	20
6.2 Attributes	20
6.3 Common services	22
6.4 Object specific services	22
7 Register Object (class = 0x07)	22
7.1 Scope	22
7.2 Attributes	23
7.3 Common services	23
7.4 Object-specific Services	23
7.5 Behaviour	24
8 Discrete Input Point Object (class code = 0x08)	24
8.1 Scope	24
8.2 Revision history	24
8.3 Attributes	25
8.4 Common services	26
8.5 Get_Attributes_All response	26
8.6 Set_Attributes_All request	27

	Page
8.7 Object-specific services	27
8.8 Behaviour	27
8.9 Attribute access rules	29
9 Discrete Output Point Object (class code = 0x09)	29
9.1 Scope	30
9.2 Attributes	30
9.3 Common services	33
9.4 Get_Attributes_All response	34
9.5 Set_Attributes_All request	35
9.6 Object-specific services	35
9.7 Behaviour	35
9.8 Attribute access rules	38
10 Analogue Input Point Object (class code = 0x0A)	38
10.1 Scope	38
10.2 Revision history	39
10.3 Attributes	39
10.4 Common services	41
10.5 Get_Attributes_All response	41
10.6 Set_Attributes_All request	42
10.7 Object-specific Services	42
10.8 Behaviour	42
10.9 Attribute access rules	45
11 Analogue Output Point Object (class code = 0x0B)	46
11.1 Scope	46
11.2 Attributes	46
11.3 Common services	48
11.4 Get_Attributes_All response	48
11.5 Set_Attributes_All request	50
11.6 Object-specific Services	50
11.7 Behaviour	50
11.8 Attribute access rules	53
12 Reserved (class code = 0x0C)	54
13 Reserved (class code = 0x0D)	54
14 Presence Sensing Object (class code = 0x0E)	54
14.1 Scope	54
14.2 Attributes	55
14.3 Common services	57
14.4 Object-specific services	58
14.5 Behaviour	58
15 Parameter Object (class code = 0x0F)	59
15.1 Scope	59
15.2 Attributes	60
15.3 Common services	67
15.4 Get_Attributes_All response	67
15.5 Object-specific services	69
15.6 Behaviour	70
16 Parameter Group Object (class code = 0x10)	70
16.1 Scope	70

	Page
16.2 Attributes	70
16.3 Common services	72
16.4 Get_Attributes_All response	72
16.5 Object-specific services	72
16.6 Behaviour	73
17 Reserved (class code = 0x11)	73
18 Group Object (class code = 0x12)	73
18.1 Scope	73
18.2 Attributes	74
18.3 Common services	75
18.4 Get_Attributes_All response	75
18.5 Object-specific services	76
18.6 Behaviour	76
18.7 Attribute access rules	76
19 to 28 Reserved (class codes = 0x13 to 0x1C)	77
29 Discrete Input Group Object (class code = 0x1D)	77
29.1 Scope	77
29.2 Attributes	77
29.3 Common services	79
29.4 Object-specific services	80
29.5 Behaviour	80
29.6 Attribute access rules	81
30 Discrete Output Group Object (class code = 0x1E)	81
30.1 Scope	81
30.2 Attributes	82
30.3 Common services	83
30.4 Object-specific services	85
30.5 Behaviour	85
30.6 Attribute access rules	86
31 Discrete Group Object (class code = 0x1F)	86
31.1 Scope	86
31.2 Attributes	87
31.3 Common services	88
31.4 Object-specific services	89
31.5 Behaviour	89
31.6 Attribute access rules	90
32 Analogue Input Group Object (class code = 0x20)	90
32.1 Scope	90
32.2 Attributes	91
32.3 Common services	92
32.4 Object-specific services	94
32.5 Behaviour	94
32.6 Attribute access rules	95
33 Analogue Output Group Object (class code = 0x21)	95
33.1 Scope	95
33.2 Attributes	96
33.3 Common services	98
33.4 Object-specific services	100

	Page
33.5 Behaviour	100
33.6 Attribute access rules	101
34 Analogue Group Object (class code = 0x22)	101
34.1 Scope	101
34.2 Attributes	102
34.3 Common services	103
34.4 Object-specific services	105
34.5 Behaviour	105
34.6 Attribute access rules	106
35 Position Sensor Object (class code = 0x23)	106
35.1 Scope	106
35.2 Attributes	106
35.3 Common services	108
35.4 Object-specific services	108
35.5 Behaviour	108
36 Position Controller Supervisor Object (class code = 0x24)	111
36.1 Scope	111
36.2 Attributes	112
36.3 Supported services	116
36.4 State diagrams	116
37 Position Controller Object (class code = 0x25)	119
37.1 Scope	119
37.2 Attributes	119
37.3 Supported services	125
37.4 Position Controller State Diagrams	126
38 Block Sequencer Object (class code = 0x26)	128
38.1 Scope	128
38.2 Attributes	128
38.3 Supported services	129
38.4 State diagrams	129
39 Command Block Object (class code = 0x39)	130
39.1 Scope	130
39.2 Attributes	130
39.3 Command specific attribute services	131
39.4 Command block supported services	135
40 Motor Data Object (class code = 0x29)	135
40.1 Scope	135
40.2 Attributes	135
40.3 Common services	139
40.4 Object-specific services	139
40.5 Behaviour	139
41 Control Supervisor Object (class code = 0x29)	139
41.1 Scope	139
41.2 Attributes	140
41.3 Common services	142
41.4 Object specific services	142
41.5 Behaviour	142
41.6 Run/stop event matrix	143

	Page
41.7 Fault and warning codes	144
42 AC/DC Drive Object (class code = 0x2A)	148
42.1 Scope	148
42.2 Attributes	149
42.3 Common services	153
42.4 Object specific services	153
42.5 Behaviour	153
42.6 Scaling of attribute values	155
43 Acknowledge Handler Object (class code = 0x2B)	156
44 Overload Object (class code = 0x2C)	156
44.1 Scope	156
44.2 Attributes	157
44.3 Common services	158
44.4 Specific services	158
44.5 Behaviour	158
44.6 Fault and warning codes	158
45 Softstart Object (class code = 0x2D)	160
45.1 Scope	160
45.2 Attributes	160
45.3 Common services	162
45.4 Specific services	162
45.5 Behaviour	162
45.6 Fault and warning codes	165
46 Selection Object (class code = 0x2E)	166
47 to 239 Reserved (class codes = 0x2F to 0xEF)	166
240 ControlNet Object (class code = 0xF0)	166
240.1 Scope	166
240.2 Attributes	166
240.3 Services	173
240.4 Behaviour	174
240.5 Module status indicator	175
241 Keeper Object (class code = 0xF1)	175
241.1 Scope	175
241.2 Attributes	175
241.3 Common services	183
241.4 Class specific services	184
241.5 Service error codes	190
241.6 Behaviour	190
241.7 Miscellaneous notes	191
241.8 Keeper power up sequence	191
242 Scheduling Object (class code = 0xF2)	197
242.1 Scope	197
242.2 Attributes	197
242.3 Services	198
242.4 Typical Scheduling session	203
Figure 10-1 — Relationship to the ISO/OSI model	1
Figure 10-2 — Identity Object State Transition Diagram	8
Figure 10-3 — Static assembly State Transition Diagram	17

	Page
Figure 10-4 — Assembly Object State Transition Diagram	19
Figure 10-5 — Register Object State Transition Diagram	24
Figure 10-6 — for Discrete Input Object State Transition Diagram	27
Figure 10-7 — Typical input object state machine (conceptual)	28
Figure 10-8 — Discrete Input Point value behaviour	29
Figure 10-9 — Logical OR	29
Figure 10-10 — Discrete Output Point value behaviour	32
Figure 10-11 — Logical OR	32
Figure 10-12 — Discrete Output Point Object State Transition Diagram	36
Figure 10-13 — Analogue Input Point Object State Transition Diagram	43
Figure 10-14 — Typical input object state machine (conceptual)	44
Figure 10-15 — Logical OR	45
Figure 10-16 — Analogue Output Point Object State Transition Diagram	51
Figure 10-17 — Typical output object state machine (conceptual)	52
Figure 10-18 — Logical OR	54
Figure 10-19 — Presence Sensing STD	58
Figure 10-20 — Parameter Object instance example	59
Figure 10-21 — Determining engineering value from actual value	66
Figure 10-22 — Determining actual value from engineering value	66
Figure 10-23 — Determining actual value from engineering value with extended precision	66
Figure 10-24 — Determining actual value from engineering value with extended precision	66
Figure 10-25 — Parameter Object State Transition Diagram	70
Figure 10-26 — Group Objects overview	73
Figure 10-27 — State Transition Diagram	76
Figure 10-28 — Logical OR	77
Figure 10-29 — Discrete Input Group — Overview	77
Figure 10-30 — State Transition Diagram	81
Figure 10-31 — Status attribute behaviour	81
Figure 10-32 — Discrete Output Group — Overview	82
Figure 10-33 — State Transition Diagram	86
Figure 10-34 — Discrete Group Object — Overview	86
Figure 10-35 — State Transition Diagram	89
Figure 10-36 — Status attribute behaviour	90
Figure 10-37 — Analogue Input Group Object — Overview	90
Figure 10-38 — State Transition Diagram	94
Figure 10-39 — Status attribute behaviour	94
Figure 10-40 — Analogue Output Group Object — Overview	95
Figure 10-41 — State Transition Diagram	100
Figure 10-42 — Status attribute behaviour	100
Figure 10-43 — Analogue Group Object — Overview	101
Figure 10-44 — State Transition Diagram	105
Figure 10-45 — Status attribute behaviour	105
Figure 10-46 — State Transition Diagram for Position Sensor Object	108
Figure 10-47 — Conceptual illustration of the state machine for a typical input object	109
Figure 10-48 — Home input behaviour	117
Figure 10-49 — Index input behaviour	118

	Page
Figure 10-50 — Registration input behaviour	119
Figure 10-51 — Position Controller Object profile generation state diagram	126
Figure 10-52 — Servo Command output generator	127
Figure 10-53 — Direct torque mode function	128
Figure 10-54 — Block Sequencer	130
Figure 10-55 — Control Supervisor State Transition Diagram	142
Figure 10-56 — AC/DC Drive Object signal flow	153
Figure 10-57 — AC/DC torque/current signal flow	154
Figure 10-58 — Drive acceleration	155
Figure 10-59 — Torque reference	155
Figure 10-60 — Soft Starter Output Voltage Behaviour Description	164
Figure 10-61 — Keeper Object Power Up State Diagram	192
Figure 10-62 — Keeper Object operating state diagram	193
Figure 10-63 — Synchronised Network Change Processing	196
Table 10-1 — Identity Object class attributes	2
Table 10-2 — Identity Object instance attributes	3
Table 10-3 — Identity Object semantics	4
Table 10-4 — Identity Object bit definitions for Status instance Attribute	5
Table 10-5 — Fault bits	5
Table 10-6 — Identity Object faults (examples — not normative)	5
Table 10-7 — Identity Object common service	6
Table 10-8 — Object-specific parameter for Reset	6
Table 10-9 — Type bit specifications	6
Table 10-10 — Class level object/service specific reply data of Get_Attributes_All	7
Table 10-11 — Instance level object/service specific reply data of Get_Attributes_All	7
Table 10-12 — Modified instance level object/service specific reply data of Get_Attributes_All	7
Table 10-13 — Identity Object STD events	9
Table 10-14 — State Event Matrix (SEM) for Identity Object	10
Table 10-15 — Identity Object SEM states	10
Table 10-16 — Message Router Object class attributes	11
Table 10-17 — Message Router Object instance attributes	12
Table 10-18 — Message Router Object common services	12
Table 10-19 — Class level object/service specific reply data of Get_Attributes_All	13
Table 10-20 — Instance level object/service specific reply data of Get_Attributes_All	13
Table 10-21 — Message Router Object Forward_Open parameters	13
Table 10-22 — Assembly Object revision history	14
Table 10-23 — Assembly Object class attributes	14
Table 10-24 — Assembly Object instance attributes	15
Table 10-25 — Assembly Object member list, instance attribute 2, data type	15
Table 10-26 — Assembly Object common services	16
Table 10-27 — Assembly Object member list rules	17
Table 10-28 — Static assembly behaviour for SEM events	18

	Page
Table 10-29 — Static Assembly Object attribute access rules	18
Table 10-30 — State Event Matrix (SEM) for dynamic assembly behaviour	19
Table 10-31 — Dynamic Assembly Object attribute access rules	20
Table 10-32 — Connection Manager Object class attributes	20
Table 10-33 — Connection Manager Object instance attributes	21
Table 10-34 — Connection Manager Object common services	22
Table 10-35 — Connection Manager Object object specific services	22
Table 10-36 — Register Object class attributes	23
Table 10-37 — Register Object instance attributes	23
Table 10-38 — Thirty bit array encoding example-informative	23
Table 10-39 — Register Object common services	23
Table 10-40 — Register Object State Event Matrix	24
Table 10-41 — Discrete Input Point Object revision history	24
Table 10-42 — Discrete Input Point Object class attributes	25
Table 10-43 — Discrete Input Point Object instance attributes	25
Table 10-44 — Discrete Input Point Object common services	26
Table 10-45 — Class level object/service specific reply data of Get_Attributes_All	26
Table 10-46 — Instance level object/service specific reply data of Get_Attributes_All	26
Table 10-47 — Instance level order	27
Table 10-48 — Discrete Input Object SEM states	28
Table 10-49 — Discrete Input Point Object STD events	28
Table 10-50 — Discrete Input Point Object SEM events	29
Table 10-51 — Discrete Output Point Object class attributes	30
Table 10-52 — Discrete Output Point Object instance attributes	31
Table 10-53 — Discrete Output Point Object optional instances attributes	32
Table 10-54 — Safe state values	32
Table 10-55 — Attribute dependencies	32
Table 10-56 — DOP behaviour	33
Table 10-57 — Discrete Output Point Object common services	33
Table 10-58 — Class level object/service specific reply data for Get_Attributes_All	34
Table 10-59 — Instance level object/service specific reply data for Get_Attributes_All	34
Table 10-60 — Instance level passed attribute order for Set_Attributes_All	35
Table 10-61 — Discrete Output Point Object SEM states	36
Table 10-62 — Discrete Output Point Object SEM events	37
Table 10-63 — State Event Matrix	38
Table 10-64 — Analogue Input Point Object revision history	39
Table 10-65 — Analogue Input Point Object class attributes	39
Table 10-66 — Analogue Input Point Object instance attributes	40
Table 10-67 — Analogue Input Point Object common services	41
Table 10-68 — Class level object/service specific reply data for Get_Attributes_All	41
Table 10-69 — Instance level object/service specific reply data for Get_Attributes_All	42

	Page
Table 10-70 — Order of attributes passed in the Set_Attributes_All request	42
Table 10-71 — Analogue Input Point Object SEM states	43
Table 10-72 — Analogue Input Point Object SEM events	44
Table 10-73 — SEM for the Analogue Input Point Object	45
Table 10-74 — Analogue Output Point class attributes	46
Table 10-75 — Analogue Output Point instance attributes	46
Table 10-76 — Default attribute values	48
Table 10-77 — Analogue Output Point Object common services	48
Table 10-78 — Class level object/service specific reply data for Get_Attributes_All	48
Table 10-79 — Instance level object/service specific reply data for Get_Attributes_All	49
Table 10-80 — Order of attributes at the instance level	50
Table 10-81 — Analogue Output Point Object SEM states	51
Table 10-82 — Analogue Output Point Object SEM events	52
Table 10-83 — SEM for the Analogue Output Point Object	53
Table 10-84 — “Safe state” optional attributes	54
Table 10-85 — Attribute dependencies	54
Table 10-86 — Presence Sensing Object class attributes	55
Table 10-87 — Presence Sensing Object instance attributes	56
Table 10-88 — Common presence sensing devices output values	57
Table 10-89 — Operate Mode usage	57
Table 10-90 — Presence Sensing Object common services	57
Table 10-91 — Attribute accessibility	58
Table 10-92 — Presence Sensing Object attribute access	58
Table 10-93 — Parameter Object class attributes	60
Table 10-94 — Parameter Object class descriptor bit values	61
Table 10-95 — Parameter Object instance attributes	62
Table 10-96 — Bit definitions of instance attribute 4, descriptor instance attribute	64
Table 10-97 — Data types supported by the Parameter Object	65
Table 10-98 — Scaling formula attributes	65
Table 10-99 — Scaling link instance attributes	66
Table 10-100 — Parameter Object common services	67
Table 10-101 — Class level object/service specific reply data of Get_Attributes_All	67
Table 10-102 — Selection of stub versus full attributes	68
Table 10-103 — Instance level object/service specific reply data of Get_Attributes_All — Stub parameters	68
Table 10-104 — Instance level object/service specific reply data of Get_Attributes_All — Full parameters	68
Table 10-105 — Parameter Object : object specific service	69
Table 10-106 — Enumerated strings	69
Table 10-107 — Parameters for Get_Enum_String request	69
Table 10-108 — Parameters for Get_Enum_String response	69
Table 10-109 — SEM events for the Parameter Object	70
Table 10-110 — Parameter Group Object class attributes	71
Table 10-111 — Parameter Group Object instance attributes	71

	Page
Table 10-112 — Parameter Group Object common services	72
Table 10-113 — Class level object/service specific reply data for Get_Attributes_All	72
Table 10-114 — Instance level object/service specific reply data for Get_Attributes_All	72
Table 10-115 — SEM for the Parameter Group Object	73
Table 10-116 — Group Object class attributes	74
Table 10-117 — Group Object instance attributes	74
Table 10-118 — Group Object common services	75
Table 10-119 — Class level object/service specific reply data for Get_Attributes_All	75
Table 10-120 — Instance level object/service specific reply data for Get_Attributes_All	75
Table 10-121 — Group Object Behaviour states	76
Table 10-122 — Discrete Input Group Object class attributes	78
Table 10-123 — Discrete Input Group Object instance attributes	78
Table 10-124 — Discrete Input Group Object common services	79
Table 10-125 — Class level object/service specific reply data for Get_Attributes_All	79
Table 10-126 — Instance level object/service specific reply data for Get_Attributes_All	80
Table 10-127 — Instance level object/service specific reply data for Set_Attributes_All	80
Table 10-128 — State equivalencies	81
Table 10-129 — Discrete Output Group Object class attributes	82
Table 10-130 — Discrete Output Group Object instance attributes	83
Table 10-131 — Discrete Output Group Object common services	83
Table 10-132 — Class level object/service specific reply data for Get_Attributes_All	84
Table 10-133 — Instance level object/service specific reply data for Get_Attributes_All	84
Table 10-134 — Instance level object/service specific reply data for Set_Attributes_All	85
Table 10-135 — State equivalencies	85
Table 10-136 — Discrete Group Object class attributes	87
Table 10-137 — Discrete Group Object instance attributes	87
Table 10-138 — Discrete Group common services	88
Table 10-139 — Class level object/service specific reply data for Get_Attributes_All	88
Table 10-140 — Instance level object/service specific reply data for Get_Attributes_All	88
Table 10-141 — State equivalencies	89
Table 10-142 — Analogue Input Group Object class attributes	91
Table 10-143 — Analogue Input Group Object instance attributes	92
Table 10-144 — Analogue Input Group Object common services	92
Table 10-145 — Class level object/service specific reply data for Get_Attributes_All	93
Table 10-146 — Instance level object/service specific reply data for Get_Attributes_All	93

	Page
Table 10-147 — Instance level object/service specific reply data for Set_Attributes_All	93
Table 10-148 — State equivalencies	94
Table 10-149 — Analogue Output Group Object class attributes	96
Table 10-150 — Analogue Output Group Object instance attributes	97
Table 10-151 — Analogue Output Group Object common services	98
Table 10-152 — Class level object/service specific reply data for Get_Attributes_All	98
Table 10-153 — Instance level object/service specific reply data for Get_Attributes_All	99
Table 10-154 — Instance level object/service specific reply data for Set_Attributes_All	99
Table 10-155 — State equivalencies	100
Table 10-156 — Analogue Group Object class attributes	102
Table 10-157 — Analogue Group Object instance attributes	103
Table 10-158 — Analogue Group Object common services	103
Table 10-159 — Class level object/service specific reply data for Get_Attributes_All	104
Table 10-160 — Instance level object/service specific reply data for Get_Attributes_All	104
Table 10-161 — State equivalencies	105
Table 10-162 — Position Sensor Object class attributes	106
Table 10-163 — Position Sensor Object instance attributes	107
Table 10-164 — Position Sensor Object common services	108
Table 10-165 — SEM events	109
Table 10-166 — State Event Matrix for the Position Sensor Object	110
Table 10-167 — Resolution values	110
Table 10-168 — Resolution example	111
Table 10-169 — Resolution	111
Table 10-170 — CAM attribute characteristics	111
Table 10-171 — Position Controller Supervisor Object class attributes	112
Table 10-172 — Position Controller Supervisor Object instance attributes	113
Table 10-173 — Position Controller Supervisor Object home and index attributes	114
Table 10-174 — Position Controller Supervisor Object registration attributes	115
Table 10-175 — Position Controller Supervisor Object axis following attributes	116
Table 10-176 — Supported services, codes, and descriptions	116
Table 10-177 — Position Controller Object class attributes	120
Table 10-178 — Position Controller Object profile attributes	120
Table 10-179 — Position Controller Object feedback and tuning attributes	123
Table 10-180 — Position Controller Object limit attributes	125
Table 10-181 — Position Controller Object supported services	125
Table 10-182 — Block Sequencer Object class attributes	128
Table 10-183 — Block Sequencer Object instance attributes	129
Table 10-184 — Block Sequencer Object supported services	129

	Page
Table 10-185 — Command Block Object class attributes	130
Table 10-186 — Command Block Object instance attributes	131
Table 10-187 — Modify attribute command — 01	132
Table 10-188 — Wait equals command — 02	132
Table 10-189 — Conditional link greater than command — 03	133
Table 10-190 — Conditional link less than command — 04	133
Table 10-191 — Delay command — 06	134
Table 10-192 — Trajectory command — 07	134
Table 10-193 — Trajectory command and wait — 08-	134
Table 10-194 — Velocity change command — 09	134
Table 10-195 — Goto home command — 10	135
Table 10-196 — Goto index command — 11	135
Table 10-197 — Goto registration position command — 12	135
Table 10-198 — Command block supported services	135
Table 10-199 — Motor Data Object class attributes	136
Table 10-200 — Motor Data Object instance attributes	136
Table 10-201 — Motor class types and section references	137
Table 10-202 — AC motor instance attributes	137
Table 10-203 — DC motor instance attributes	138
Table 10-204 — Motor Data Object instance attribute abbreviations	138
Table 10-205 — Motor Data Object common services	139
Table 10-206 — Control Supervisor Object class attributes	140
Table 10-207 — Control Supervisor Object instance attributes	140
Table 10-208 — Control Supervisor Object common services	142
Table 10-209 — Control Supervisor State Event Matrix	143
Table 10-210 — Run/stop event matrix	143
Table 10-211 — Drive behaviour	144
Table 10-212 — Fault and warning codes	144
Table 10-213 — AC/DC Drive Object class attributes	149
Table 10-214 — AC/DC Drive Object instance attributes	149
Table 10-215 — AC/DC Drive Object basic units	152
Table 10-216 — AC/DC Drive Object common services	153
Table 10-217 — Overload Object class attributes	157
Table 10-218 — Overload Object instance attributes	157
Table 10-219 — Overload Object common services	158
Table 10-220 — Fault and warning codes	159
Table 10-221 — Softstart Object class attributes	160
Table 10-222 — Softstart Object instance attributes	161
Table 10-223 — Softstart Object common services	162
Table 10-224 — AC Motor Starter Device Profiles fault and warning codes	165
Table 10-225 — ControlNet Object class attributes	166
Table 10-226 — ControlNet Object instance attributes	166
Table 10-227 — TUI status flag bits	171
Table 10-228 — Channel state bits	172
Table 10-229 — ControlNet Object common services	173
Table 10-230 — ControlNet Object class specific services	174
Table 10-231 — Keeper Object class attributes	175

	Page
Table 10-232 — Keeper Object instance attributes	176
Table 10-233 — Keeper operating state definitions	180
Table 10-234 — Port status flag bit definitions	180
Table 10-235 — TUI Status Flag Bits	181
Table 10-236 — Memory requirements (in bytes) for the Keeper attributes	182
Table 10-237 — Keeper Object common services	183
Table 10-238 — Keeper Class specific services	184
Table 10-239 — Obtain_Network_Resource error codes	186
Table 10-240 — Hold_Network_Resource error codes	186
Table 10-241 — Release_Network_Resource error codes	186
Table 10-242 — Change_Start error codes	187
Table 10-243 — Change_Complete error codes	187
Table 10-244 — Change_Abort error codes	188
Table 10-245 — Get_Signature error codes	188
Table 10-246 — Get_Attribute_Fragment error codes	189
Table 10-247 — Set_Attribute_Fragment error codes	189
Table 10-248 — Wire order format of the TUI packet	190
Table 10-249 — Service error codes	190
Table 10-250 — Keeper Object operating states	191
Table 10-251 — Keeper Object State Event Matrix	194
Table 10-252 — Scheduling Object class level attributes	197
Table 10-253 — Scheduling Object instance level attributes	197
Table 10-254 — Needed service at class level and instance level	198
Table 10-255 — Status error descriptions	199
Table 10-256 — Status error descriptions for Delete	199
Table 10-257 — Status error descriptions for Kick_Timer	199
Table 10-258 — Status error descriptions for Read	200
Table 10-259 — Status error descriptions for Conditional_Write	201
Table 10-260 — Status error descriptions for Forced_Write	201
Table 10-261 — Status error descriptions for Change_Start	201
Table 10-262 — Status error descriptions for Break_Connections	202
Table 10-263 — Status error descriptions for Change_Complete	202
Table 10-264 — Status error descriptions for Restart_Connections	203

Foreword

This part of DD241 describes the Application Layer of the deterministic control network.

This publication consists of the following parts:

- *Part 1: General Description;*
- *Part 2: Physical Layer and Media;*
- *Part 3: Data Link Layer;*
- *Part 4: Network and Transport Layers;*
- *Part 5: Data Management;*
- *Part 6: Object Model;*
- *Part 7: Device Profiles;*
- *Part 8: Electronic Data Sheet;*
- *Part 9: Station Management;*
- *Part 10: Object Library.*

This Draft for Development is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because the subject matter is still under development requiring wider exposure. It should be applied on this provisional basis so that information and experience of its practical application may be obtained.

A review of this Draft for Development will be carried out not later than 2 years after its Publication. Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development, will be made in an announcement in *Update Standards*.

According to the replies received the responsible BSI Committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

Observations which it is felt should receive attention before the official call for comments will be welcomed.

These should be sent to the Secretary of BSI Technical Committee AMT/7 at BSI, 389 Chiswick High Road, London, W4 4AL, giving the document reference and clause number and proposing, where possible, an appropriate revision of the text.

Summary of pages

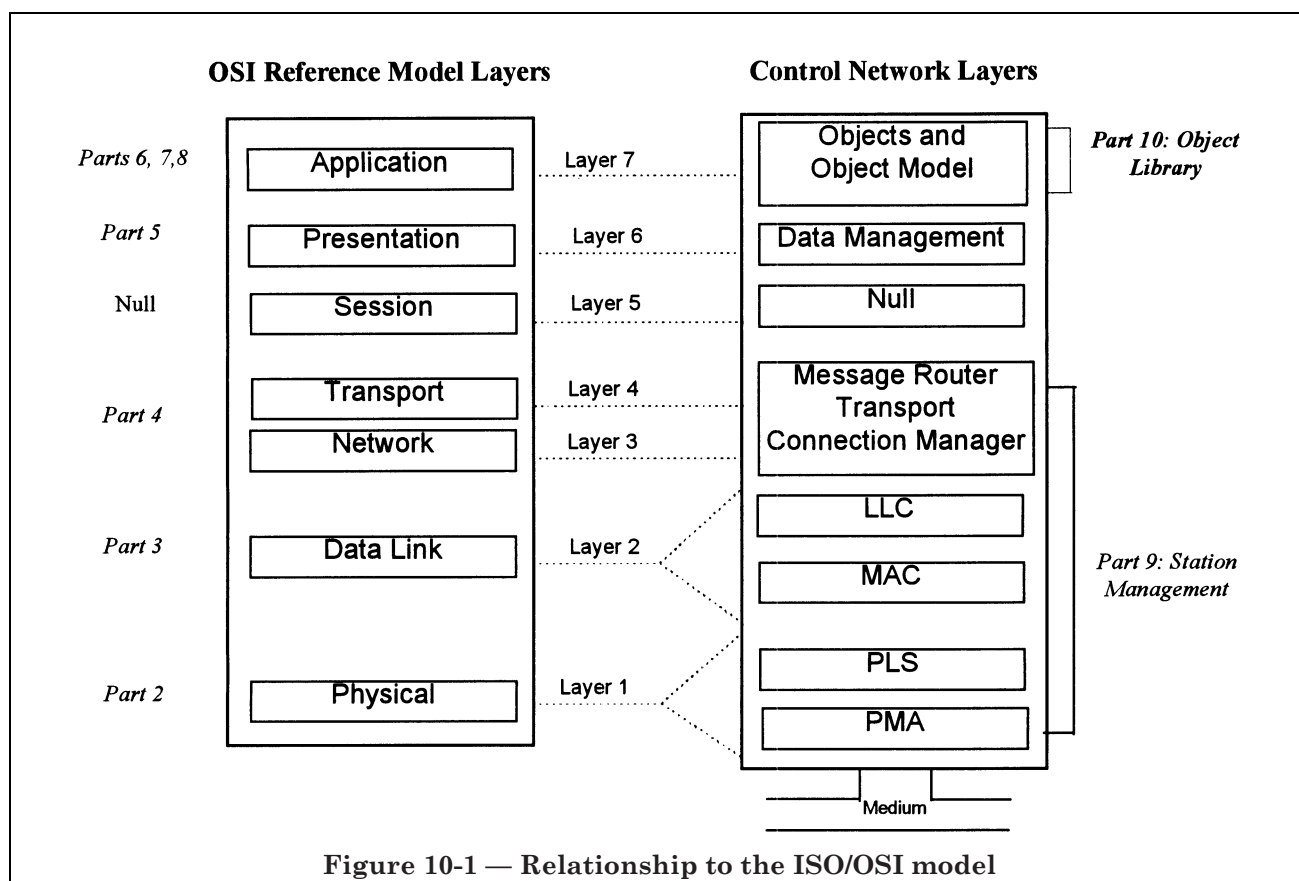
This document comprises a front cover, an inside front cover, pages i to xiv, pages 1 to 204 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

0 Introduction

In this standard, object modelling is used to represent the network visible behaviour of devices. Devices are modelled as a collection of objects. Each class of objects is a collection of related services, attributes and behaviours. Services are procedures an object performs. Attributes are characteristics of objects represented by values, which can vary. An object's behaviour is an indication of how the object responds to particular events.

Part 10 contains the object specifications. The standard uses object modelling as the way to describe system behaviour. As shown by Figure 10-1 the objects perform the Layer 7 Application functions.



1 Identity Object (class code = 0x01)

1.1 Scope

The Identity Object shall provide identification and general information about the device. Instance one of the Identity Object shall be present in all devices.

The first instance shall identify the whole device. It shall be used for electronic keying and by applications wishing to determine what nodes are on the network. Other instances are optional. They may be provided by a device to give additional information about a device and its subsystems.

1.2 Attributes

1.2.1 Class attributes

The Identity Object shall support the class attributes as specified Table 10-1.

Table 10-1 — Identity Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	First revision, value = 1
2	Conditionall ^a	Get	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device.	The largest instance number of a created object at this class hierarchy level.
6	Optional	Get	Max ID Number of Class Attributes	UINT	The attribute ID number of the last class attribute of the class definition implemented in the device.	The number of the last class attribute implemented to simplify auto determination of class's implementation by a remote terminal.
7	Optional	Get	Max ID Number of Instance Attributes	UINT	The attribute ID number of the last instance attribute of the class definition implemented in the device.	The number of the last instance attribute implemented to simplify auto determination of class's implementation by a remote terminal.
^a If multiple subcomponents can be identified using instances of the Identity Object, then this attribute shall be required . The numerically lowest available integer shall be assigned as the Instance Identifier of a newly created Identity Object Instance.						

1.2.2 Instance attributes

The Identity Object shall support the instance attributes as specified in Table 10-2.

Table 10-2 — Identity Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Required	Get	Vendor ID	UINT	Identification of each vendor by number	See Table 10-3
2	Required	Get	Device Type	UINT	Indication of general type of product	See Table 10-3
3	Required	Get	Product Code	UINT	Identification of a particular product of an individual vendor	See Table 10-3
4	Required	Get	Revision	STRUCT of	Revision of the item that this instance of the Identity Object represents	See Table 10-3
			Major Revision	USINT		See Table 10-3
			Minor Revision	USINT		See Table 10-3
5	Required	Get	Status	WORD	Summary status of device	See Table 10-3 and Table 10-4
6	Required	Get	Serial Number	UDINT	Serial number of device	See Table 10-3
7	Required	Get	Product Name	SHORT_STRING	Human readable identification	See Table 10-3
8	Optional Not Used	Get	State	USINT	Present state of the device as represented by the state transition diagram	This attribute not used. Reserved
9	Optional Not Used	Get	Configuration Consistency Value	UINT	Contents identify configuration of device	This attribute not used. Reserved
10	Optional Not Used	Get/Set	Heartbeat Interval	USINT	The nominal interval between heartbeat messages in seconds	This attribute not used. Reserved

The following instance attributes shall be used to identify with certainty the appropriate device targeted by a connection originator

- Vendor;
- Device Type;
- Product Code;
- Revision.

This collection of attributes, when kept by the connection originator, shall be referred to as a device's "electronic key".

1.2.3 Semantics

Table 10-3 — Identity Object semantics

Attribute name	Description
Vendor ID	Vendor IDs uniquely identify a particular vendor. The value zero shall not be used.
Device type	Device types shall be used to identify the device profile that a particular device is using. Device profiles define minimum requirements a device shall implement as well as common options. A listing of the presently defined Device Types can be found in part 7.
Product code	The vendor assigned Product Code shall identify a particular product within a device type. Each vendor shall assign this code to each of its products. The Product Code typically maps to one or more catalogue/model numbers. Products shall have different codes if their configuration and/or runtime options are different. Such devices present a different logical view to the network. The value zero is not valid.
Revision	The Revision attribute, which consists of Major and Minor Revisions, shall identify the <i>Revision</i> of the item the Identity Object is representing. The value zero shall not be valid for either the Major and Minor Revision fields. The Major and Minor Revision are typically displayed as major.minor. Minor revisions shall be displayed as three digits with leading zeros as necessary. The Major Revision attribute shall be limited to 7 bits. The eighth bit is reserved and shall be zero. The major revision shall be incremented by the vendor when there is a significant change to the “fit, form, or function” of the product. Any changes that effect the configuration choices available to the user (and therefore the Electronic Data Sheet) <u>require</u> incrementing the major revision. The minor revision may be used to identify changes in a product that do not effect user configuration choices, e.g. bug fixes, hardware component change, labelling change, etc.
Status	This attribute shall represent the current status of the entire device. Its value changes as the state of the device changes. The Status attribute is a WORD. The bit definitions are described in Table 1-4.
Serial number	This attribute provides a number used in conjunction with the Vendor ID to form a unique identifier for each device. Each vendor shall be responsible for guaranteeing the uniqueness of the serial number across all of its devices.
Product name	This text string may represent a short description of the product/product family represented by the product code in attribute 3. The same product code may have a variety of product name strings. The maximum number of characters in this string shall be 32.
State [optional]	Not used. Included for compatibility purposes. This attribute provides an indication of the present state of the device. The nature of a Major Unrecoverable Fault may be such that it may not be accurately reflected by the State attribute.
Configuration consistency value [optional]	Not used. Included for compatibility purposes. A product may automatically modify the <i>Configuration Consistency Value</i> whenever any non-volatile attribute is altered. A client node may, or may not, compare this value to a value within its own memory prior to system operation. The client node's behaviour, upon detection of a mismatch, is vendor specific. The <i>Configuration Consistency Value</i> may be a CRC, incrementing count or any other mechanism. The only requirement is that if the configuration is different, the <i>Configuration Consistency Value</i> shall be different to reflect the difference.
Heartbeat Interval [optional]	Not used. Included for compatibility purposed. The nominal interval between heartbeat messages in seconds.

Table 10-4 — Identity Object bit definitions for Status Instance Attribute

Bit(s):	Called:	Definition
0	Owned	A 1 (TRUE) shall indicate the device (or an object within the device) has an owner.
1		Reserved, set to 0
2	Configured	A 1 (TRUE) shall indicate the application of the device has been configured to do something different than the “out-of-box” default. This shall not include configuration of the communications.
3		Reserved, set to 0
4,5,6,7		Vendor-specific
8	Minor Recoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which is thought to be recoverable. The problem shall not cause the device to go into one of the faulted states. See Table 10-5 and Behaviour subclause 1.6.
9	Minor Unrecoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which is thought to be unrecoverable. The problem shall not cause the device to go into one of the faulted states. See Table 10-5 and Behaviour subclause 1.6.
10	Major Recoverable Fault	A 1 (TRUE) shall indicate the device detected a problem with itself, which caused the device to go into the “Major Recoverable Fault” state. See Table 10-5 and Behaviour subclause 1.6.
11	Major Unrecoverable Fault	A 1 (TRUE) indicates the device detected a problem with itself, which caused the device to go into the “Major Unrecoverable Fault” state. See Table 10-5 and Behaviour subclause 1.6.
12, 13		Reserved, shall be set to 0
14, 15		Reserved, shall be set to 0

The values of Status bits 8,9,10, and 11 shall indicate the state of the device as shown in Table 10-5.

Table 10-5 — Fault Bits

Value	State
Bit 8	Minor Recoverable Fault
Bit 9	Minor Unrecoverable Fault
Bit 10	Major Recoverable Fault
Bit 11	Major Unrecoverable Fault

NOTE The events that constitute a fault (recoverable or unrecoverable) shall be determined by the device implementor. The examples in Table 10-6 may help to define the various types of faults.

Table 10-6 — Identity Object faults (examples — not normative)

Fault	Example description	Recoverable	Unrecoverable
Minor Recoverable Fault (no state change)	an analogue input device is sensing an input that exceeds the configured maximum input value	Bit 8	
Minor Unrecoverable Fault (no state change)	the device’s battery backed RAM requires a battery replacement. The device shall continue to function properly until the first time power is cycled		Bit 9
Major Recoverable Fault (state changes)	the device’s configuration is incorrect or incomplete	Bit 10	
Major Unrecoverable Fault (state changes)	the device failed its ROM checksum process		Bit 11

The event that sets bits 8 or 10 shall also cause a state change, see Figure 10-2 for more information.

1.3 Common services

1.3.1 All services

The Identity Object shall provide the common services as shown in Table 10-7.

Table 10-7 — Identity Object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Required ^a	Get_Attributes_All	Returns a predefined listing of this objects attributes (See the Get_Attributes_All response definition in subclause 1.4)
0x05	Optional	Required	Reset	Invokes the Reset service for the device.
0x0E	Optional ^a	Optional ^a	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	n/a	n/a ^a	Set_Attribute_Single	Modifies an attribute.
0x11	Optional	n/a	Find_Next_Object_Instance	Causes the specified Class to search for and return a list of instance IDs of existing instances of the Identity object.

NOTE See part 6 for definitions of these common services.

^a The following are differences between DeviceNet's Identity Object and ControlNet's:

DeviceNet requires the Get_Attribute_Single service if any class attributes are supported. ControlNet does not.

DeviceNet does not require the Get_Attributes_All service, whereas ControlNet does.

DeviceNet requires the Set_Attribute_Signal service if the Heartbeat Interval attribute is supported. The ControlNet Identity Object does not support the Set_Attribute_Signal service.

1.3.2 Reset service

When the Identity Object receives a Reset request, it shall

- determine if it can provide the type of reset requested;
- respond to the request;
- attempt to perform the type of reset requested.

The Reset common service shall have the object-specific parameter specified in Table 10-8.

Table 10-8 — Object-specific parameter for Reset

Name	Type	Description of request parameters	Semantics of values
Type	USINT	Type of Reset	See Table 10-9.

The parameter Type for the Reset common service shall have the values defined as in Table 10-9.

Table 10-9 — Type bit specifications

Value	Type of Reset
0	Shall Emulate as closely as possible cycling power on the item the <i>Identity Object</i> represents. This value shall be the default if this parameter is omitted.
1	Shall Return as closely as possible to the out-of-box configuration, then shall emulate cycling power as closely as possible.
2-255	Reserved

1.4 Get_Attributes_All response

At the **class level**, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as defined in Table 10-10.

Table 10-10 — Class level object/service specific reply data of Get_Attributes_All

Attribute ID	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	Revision (low byte) Default = 1							
	1	Revision (high byte) Default = 0							
2	2	Max Instance (low byte) Default = 1							
	3	Max Instance (high byte) Default = 0							
6	4	Max ID Number of Class Attributes (low byte) Default = 0							
	5	Max ID Number of Class Attributes (high byte) Default = 0							
7	6	Max ID Number of Instance Attributes (low byte) Default = 0							
	7	Max ID Number of Instance Attributes (high byte) Default = 0							

At the **instance level**, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as defined in Table 10-11.

Table 10-11 — Instance level object/service specific reply data of Get_Attributes_All

Attribute ID	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	Vendor (low byte)							
	1	Vendor (high byte)							
2	2	Device Type (low byte)							
	3	Device Type (high byte)							
3	4	Product Code (low byte)							
	5	Product Code (high byte)							
4	6	Major Revision							
	7	Minor Revision							
5	8	Status (low byte)							
	9	Status (high byte)							
6	10	Serial Number (low byte)							
	11	Serial Number							
	12	Serial Number							
	13	Serial Number (high byte)							
7	14	Product Name length							
	15	Product Name (1st character)							
	16	Product Name (2nd character)							
	n	Product Name (last character)							

Because the length of the name is not known before issuing the Get_Attributes_All service request, implementors shall allow enough memory space to store a response up to 32 characters in length.

NOTE To maintain compatibility with DeviceNet, if any instance attributes are added at some later date, the instance Get_Attributes_All response shall be modified to include the attributes as defined in Table 10-12.

Table 10-12 — Modified instance level object/service specific reply data of Get_Attributes_All

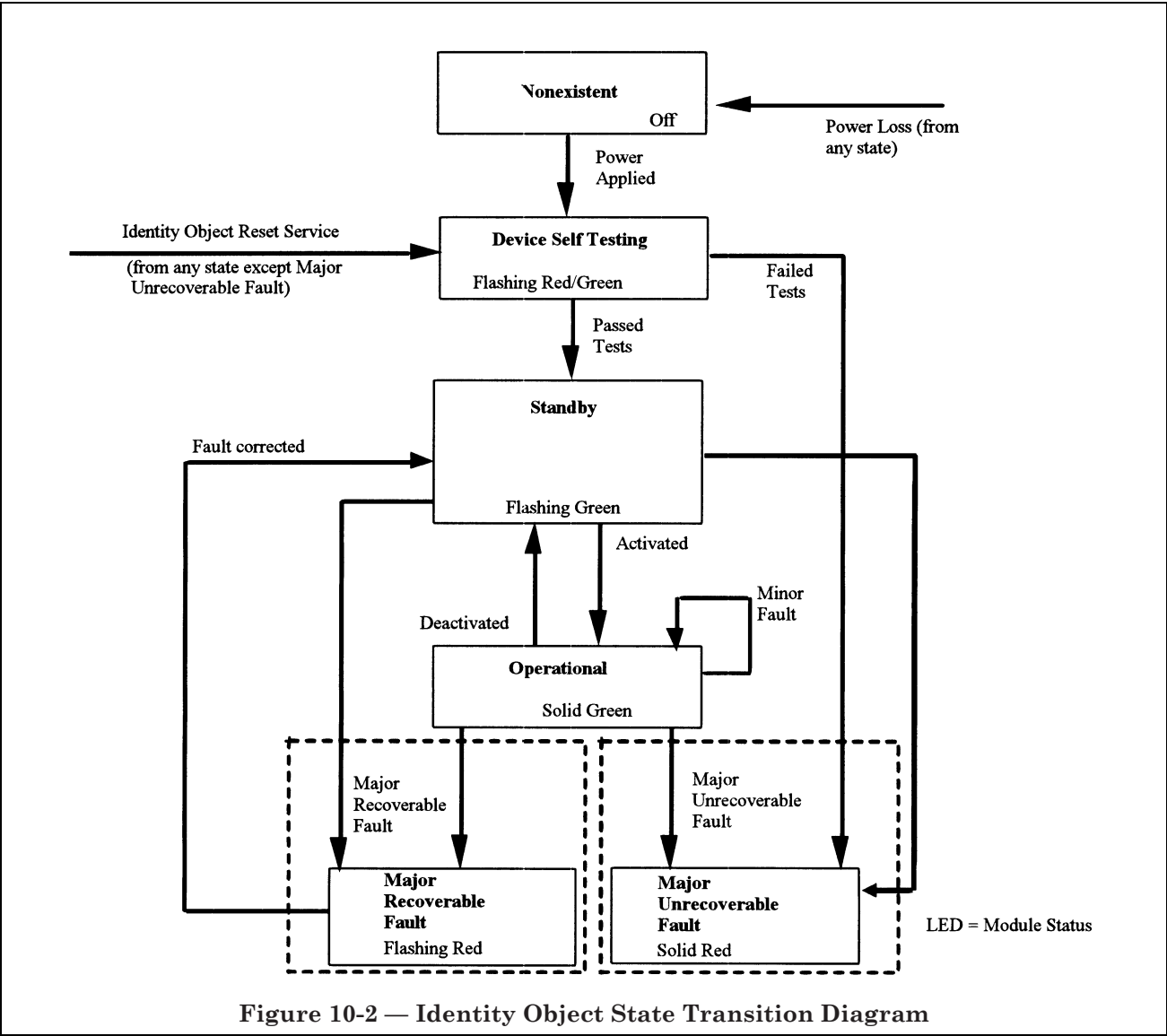
Attribute ID	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
8	n+ 1	State Default = 255							
9	n+ 2	Space reserved for attribute 9 low byte, default=0							
	n+ 3	Space reserved for attribute 9 high byte, default=0							
10	n+4	Heartbeat Interval, Default = 0							

1.5 Object-specific services

The Identity Object shall provide no object-specific services.

1.6 Behaviour

The behaviour of the Identity Object shall be as illustrated in Figure 10-2.
A device may not be able to communicate in the Major Unrecoverable Fault state. Therefore, it might not be able to report a Major Unrecoverable Fault. It shall not process a Reset service. The only exit from a Major Unrecoverable Fault shall be to cycle the device power.



The STD for the Identity Object contains the events listed in Table 10-13.

Table 10-13 — Identity Object STD events

Event	Description
Power Loss	the device is powering down
Power Applied	the device is powering up
Passed Tests	the device has successfully passed all self tests
Failed Tests	the device has failed at least one of its self tests
Activated	the device's configuration is valid and the application for which the device was designed shall now be capable of executing (communications channels may or may not yet be established)
Deactivated	the device's configuration is no longer valid and the application for which the device was designed is no longer capable of executing (communication channels may or may not still be established)
Minor fault	a fault classified as either a minor unrecoverable fault or a minor recoverable fault has occurred
Major Recoverable Fault	an event classified as Major Recoverable Fault has occurred
Major Unrecoverable Fault	an event classified as a Major Unrecoverable Fault has occurred
Fault Corrected	the major recoverable fault is corrected
Reset	response to external command
Module Status LED	lists the LED colour for each of the object states

Table 10-14 — State Event Matrix (SEM) for Identity Object

Event	Non-existent	Device self testing	Standby	Operational	Major unrecoverable Fault	Major recoverable Fault
Power Loss	Not Applicable	Transition to Non-existent	Transition to Non-existent	Transition to Non-existent	Transition to Non-existent	Transition to Non-existent
Power Applied	Transition to Device Self Testing	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Failed Tests	Not Applicable	Transition to Major Unrecoverable Fault	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Passed Tests	Not Applicable	Transition to Standby	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Deactivated	Not Applicable	Ignore Event	Ignore Event	Transition to Standby	Ignore Event	Ignore Event
Activated	Not Applicable	Ignore Event	Transition to Operational	Ignore Event	Ignore Event	Ignore Event
Major Recoverable Fault	Not Applicable	Not Applicable	Transition to Major Recoverable Fault	Transition to Major Recoverable Fault	Ignore Event	Ignore Event
Major Unrecoverable Fault	Not Applicable	Not Applicable	Transition to Major Unrecoverable Fault	Transition to Major Unrecoverable Fault	Ignore Event	Ignore Event
Minor Recoverable Fault	Not Applicable	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Ignore Event
Minor Unrecoverable Fault	Not Applicable	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Ignore Event
Fault Corrected	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Not Applicable	Transition to Standby
Reset	Not Applicable	Restart Self Tests	Transition to Device Self Testing	Transition to Device Self Testing	Ignore Event	Transition to Device Self Testing
Module Status LED	Off	Flashing Red/Green	Flashing Green	Solid Green	Solid Red	Flashing Red

The SEM for the Identity Object shall contain the states listed in Table 10-15.

Table 10-15 — Identity Object SEM states

States	Description
Non-existent	the device is without power
Device self testing	the device is executing its self tests
Standby	the device needs commissioning due to an incorrect or incomplete configuration
Operational	the device is operating in a fashion that is normal for the device
Major recoverable fault	the device has experienced a fault that is believed to be recoverable
Major unrecoverable fault	the device has experienced a fault that is believed to be unrecoverable

2 Message Router (class code = 0x02)

2.1 Scope

The Message Router Object shall provide a messaging connection point through which a client may address a service to any object class or instance residing in the physical device.

2.2 Attributes

2.2.1 Class attributes

The Message Router Object shall support the class attributes as specified in Table 10-16.

Table 10-16 — Message Router Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	First revision, value = 1
4	Optional	Get	Optional attribute list	STRUCT of	List of optional instance attributes utilised in an object class implementation	A list of attribute numbers specifying the optional instance attributes implemented in the device for this class.
			number attributes	UINT	Number of attributes in the optional attribute list.	The number of attribute numbers in the list.
			optional attributes	ARRAY of UINT	List of optional attribute numbers.	The optional attribute numbers.
5	Optional	Get	Optional service list	STRUCT of	List of optional services utilised in an object class implementation.	A list of service codes specifying the optional services implemented in the device for this class.
			number services	UINT	Number of services in the optional service list.	The number of service codes in the list.
			optional services	ARRAY of UINT	List of optional service codes	The optional service codes.
6	Optional	Get	Max ID number of class attributes	UINT	The attribute ID number of the last class attribute of the class definition implemented in the device.	The number of the last class attribute implemented to simplify auto determination of class's implementation by a remote terminal.
7	Optional	Get	Max ID number of instance attributes	UINT	The attribute ID number of the last instance attribute of the class definition implemented in the device.	The number of the last instance attribute implemented to simplify auto determination of class's implementation by a remote terminal.

2.2.2 Instance attributes

The Message Router Object shall support the instance attributes as specified in Table 10-17.

Table 10-17 — Message Router Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Object_List	STRUCT of	A list of supported objects	Structure with an array of object class codes supported by the device via the message router
			Number	UINT	Number of supported classes in the classes array	The number of class codes in the classes array
			Classes	ARRAY of UINT	List of supported class codes	The class codes supported by the device
2	Optional	Get	Number available	UINT	Maximum number of connections supported	Count of the max number of connections supported by the message router
3	Optional.	Get	Number active	UINT	Number of connections currently used by system components	Not used. Reserved for compatibility purposes Value = 0.
4	Optional.	Get	Active connections	ARRAY of UINT	A list of the connection IDs of the currently active connections	This attribute not used. Reserved for compatibility purposes

2.3 Common services

2.3.1 All services

The Message Router Object shall provide the common services listed in Table 10-18.

Table 10-18 — Message Router Object common services

Service code	Need in implementation		Service name	Description
	Class	Instance		
0x01	Optional	Optional	Get_Attributes_All	Returns the contents of all attributes
0x03	Optional	Optional	Get_Attributes_List	Returns the contents of the requested attributes
0x0E	Conditional ^a	Conditional ^a	Get_Attribute_Single	Returns the contents of the specified attribute.

NOTE See part 6 for definitions of these common services.

^a The Get_Attribute_Single service shall be required if any attributes are implemented.

2.4 Get_Attributes_All response

At the class level, the order of the attributes returned in the “object/service specific reply data” portion of the Get_Attributes_All response shall be as specified in Table 10-19.

Table 10-19 — Class level object/service specific reply data of Get_Attributes_All

Class attribute ID	Attribute name, description and default value
1	Revision default = 1
4	Optional attribute list, number of attributes default = 0
5	Optional service list, number of services default = 0
6	Max ID number of class attributes default = 0
7	Max ID number of instance attributes default = 0

Default values for all unsupported class attributes shall be inserted as shown in Table 10-19.

Table 10-20 — Instance level object/service specific reply data of Get_Attributes_All

Instance attribute ID	Attribute name, description and default value
1	Object list number, number of supported classes default = 0
2	Number available, maximum number of connections default = 0
3	Always = 0x0000t

Insert default values for all unsupported class attributes as shown in Table 10-20.

2.5 Object-specific services

The Message Router Object provides no object specific services.

2.6 Behaviour

The Message Router Object shall receive messages and

- interpret the class instance specified in a message;
- route a service to the specified object;
- interpret services directed to it;
- route a response to the correct service source.

The message router shall have one state, the run state. It shall always be running (after power-up) and shall be fixed by device design. More information concerning the message router can be found in part 4, Network and Transport Layers.

Message routers shall support 1 or more connections to them. A Forward_Open with the parameters listed in Table 10-21 shall also be supported.

Table 10-21 — Message Router Object Forward_Open parameters

Parameter	Parameter Value
O ⇒ T priority	low
O ⇒ T connection type	point-to-point
O ⇒ T fixed/variable	variable
O ⇒ T size	up to 504 bytes
T ⇒ O priority	low
T ⇒ O connection type	point-to-point
T ⇒ O fixed/variable	variable
T ⇒ O size	up to 504 bytes
client/server	server
trigger mode	ignore
transport class	class 3: verified

2.7 Service request

Interpretation of the class instance shall be performed on every service received by the message router.

Any class instance that cannot be interpreted by an implementation of a message router in a device shall report the Object_Not_Found error.

The service shall then be routed to the target object for processing. If the service is directed to the message router itself, then the message router shall process the service requests and respond accordingly.

2.8 Service response

All service responses shall be routed to the source of the service requests. The source may be one of the:

- the Unconnected Message Manager (UCMM);
- an active connection to the message router object.

All connected service responses shall be routed back across the connection from which the service request was received. All unconnected service responses (UCMM) shall be returned via the same UCMM that provided the request.

3 DeviceNet Object (class code = 0x03)

Class code 0x03 shall be reserved.

4 Assembly Object (class code = 0x04)

4.1 Scope

The Assembly Object binds attributes of multiple objects, allowing data to or from each object to be sent or received over a single connection. Assembly objects are used to produce and/or consume data to/from the network. An instance of the assembly object can both produce and consume data from the network if designed to do so.

4.2 Revision history

Table 10-22 — Assembly Object revision history

Class revision	Description of changes
01	Initial Release
02	1) Class specific service codes 0x4B (Add_Member) and 0x4C (Delete_Member) are obsolete due to the creation of the Common Services. 2) Added support of common services 0x18,19,1A and 1B. 3) Changed bit 15 of the Member Data Description sub-attribute of instance attribute 2 from representing read/write to representing most significant bit of the member data size. This allows the same Assembly Object to both produce and consume. Changed name from Member Data Description to Member Data Size.

4.3 Attributes

4.3.1 Class attributes

The Assembly Object shall support the class attributes as specified in Table 10-23.

Table 10-23 — Assembly Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Required	Get	Revision	UINT	Revision of this object	Second revision, value = 2.
2	Optional	Get	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device.	The largest instance number of a created object at this class hierarchy level.

4.3.2 Instance attributes

The Assembly Object shall support the instance attributes as specified in Table 10-24.

Table 10-24 — Assembly Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Conditional ^a	Get ^c	Number of Members in List	UINT	Number of members in attribute 2.	
2	Conditional ^a	Set for Dynamic ^c Get for Static ^c	Member List	ARRAY of STRUCT	The member list is an array of paths	This attribute has a complex data type. See Table 10-25 for details.
			Member Data Size	UINT	Size of member data (in bits).	Size in bits
			Member Path Size	UINT	Size of Member Path (in bytes).	0 = Empty Path
			Member Path	ARRAY of BYTE	Path to/from data for this member	See part 4-5 for the format of path.
3	Required	Set ^c	Data	ARRAY of BYTES	All of the member data packed into one array.	See note 2.

^a Optional for Static Assembly. Required for Dynamic Assembly.

^b Instance attribute 3, Data, shall contain all of the member data packed into one array. This data may contain many different data types. For efficiency it is best to keep this data word aligned by packing it on word boundaries and adding padding as needed. This can be accomplished by using “empty paths” (Member Path Size = 0).

^c The attribute access rules are state dependent. See Table 10-29 and Table 10-31 for access rules per state and assembly object type.

Table 10-25 — Assembly Object member list, instance attribute 2, data type

Attribute ID	Data type/ sub-attribute name	Sub-attribute data type	Description of attribute	Semantics of values
Member List	ARRAY of STRUCT		The member list is an array of each member size and origin (path)	
	Member Data Size	UINT	Size of member data (in bits)	Size in bits
	Member Path Size	UINT	Size of Member Path (in bytes)	Size in bytes
	Member Path	ARRAY of BYTE	Path to the member data. See part 4 for definition of Path.	

Each member in the Member List shall contain the above entries.

4.4 Common services

The Assembly Object shall provide the common services listed in Table 10-26.

Table 10-26 — Assembly Object common services

Service code	Need in implementation				Service name	Description of service
	Static Assembly		Dynamic Assembly			
	Class	Instance	Class	Instance		
0x08	n/a	n/a	Required	n/a	Create	Instantiates an Assembly Object within a specified class. Response contains instance number.
0x09	n/a	n/a	Optional ^a	Required	Delete	Deletes an Assembly Object and releases all associated resources.
0x0E	Required	Required	Required	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	n/a	Optional	n/a	Conditional ^b	Set_Attribute_Single	Modifies an attribute value.
0x18	n/a	Optional	n/a	Optional	Get_Member	Returns a member from the Assembly Member List.
0x19	n/a	n/a	n/a	Optional	Set_Member	Modifies a member of the Assembly Member List.
0x1A	n/a	n/a	n/a	Conditional ^c	Insert_Member	Adds a member to the Assembly Member List.
0x1B	n/a	n/a	n/a	Conditional ^c	Remove_Member	Removes a member from the Assembly Member List.
^a At the class level this service deletes all existing Assembly instances. ^b If the Set_Attribute_Single common service is not supported at the Instance level for a Dynamic Assembly, then the device shall support the Insert_Member and Remove_Member common services. ^c If the Insert_Member and Remove_Member common services are not supported at the Instance level for a Dynamic Assembly, then the device should support the Set_Attribute_Single common service. See part 6 for definitions of these common services.						

4.5 Object-specific services

The Assembly Object provides no object-specific services.

4.6 Behaviour

Assembly objects shall be either dynamic or static.

- **Dynamic** shall use assemblies member lists created and managed by the user. The member list may be altered by adding or deleting members;
- **Static** assemblies shall use member lists defined by the device profile or by the product implementor. The Instance number, number of members, and member list shall be fixed.

NOTE Static assemblies can usually be implemented entirely in ROM.

Instances of the Assembly Object shall be divided into the same ranges as specified in part 6-5 to provide for extensions to device profiles.

The behaviour of the Assembly Object shall differ by the type of Assembly. A Dynamic Assembly member list shall be created and managed by the user of the device. The manager of the dynamic assembly shall specify and maintain the member list. A Static Assembly member list shall be defined by the device manufacturer. It shall not be modified.

To provide for the ability to create and delete objects, as well as change member lists, Dynamic Assemblies shall support additional services which are not supported by Static Assemblies.

The rules listed in Table 10-27 shall apply to the member lists, instance attribute 2, of both static and dynamic assemblies.

Table 10-27 — Assembly Object member list rules

Rule number	Type of Assembly applies to		Rule description
	Static	Dynamic	
1	Yes	Yes	When an empty path (Member Path Size = 0) is used in an assembly member list, the assembly shall insert/discard the number of bits as specified in the Member Data Size sub-attribute field when producing/consuming. The assembly object shall insert if it is producing and discards if it is consuming. The assembly shall use a value of zero for all produced data which has been inserted. NOTE The use of an empty consumed path allows, for example, data destined for multiple nodes to be sent in a single message since each node can be configured to discard data in the message not intended for it.
2	Yes	Yes	The empty path shall be supported for all dynamic assemblies. Static assemblies may also contain empty paths.
3	Yes	Yes	No checking shall be done by the Assembly Object at any time to verify that the size of the member data is correct for the given member path. It shall be the responsibility of the assembly member to properly handle too much or too little data. The assembly shall be required to deliver the configured number of bits to the member.
4	Yes	Yes	No padding shall be done by the Assembly Object itself to align data from each assembly member on a byte, word, or other boundary. Empty paths may be used to provide padding if desired.

4.7 Static assemblies

Figure 10-3 — Static assembly State Transition Diagram Table 10-29 — State Event Matrix and Table 10-30 — Attribute Access Table illustrate the behaviour of Static assemblies.

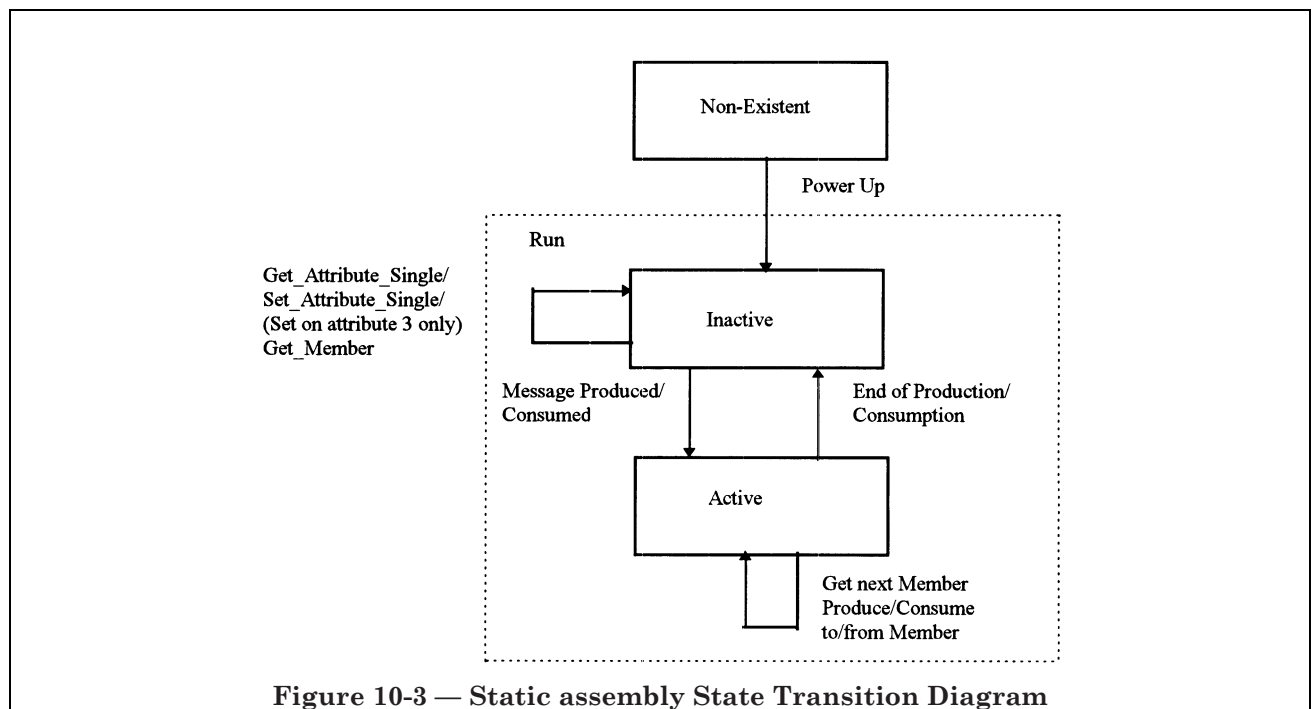


Figure 10-3 — Static assembly State Transition Diagram

Table 10-28 — Static assembly behaviour for SEM events

Event	Static Assembly Object state		
	Non-existent	Inactive	Active
Power Up	Transition to Inactive	Not applicable	Not applicable
Get_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Set_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0Cx 0x0C)
Get_Member	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Message produced/ consumed	Error: Object does not exist. (General Error Code 0x16)	Transition to Active. Begin producing/consuming from/to each member in list.	Error: Object State Conflict (General Error Code 0x0C)
End of production/ consumption	Error: Object does not exist. (General Error Code 0x16)	Error: Object State Conflict (General Error Code 0x0C)	Transition to Inactive

The attribute access rules for the static assembly shall be as shown in Table 10-29.

Table 10-29 — Static Assembly Object attribute access rules

Instance attribute number	Instance attribute name	Static Assembly Object state		
		Non-existent	Inactive	Active
1	Number_of_Members	Not available	Read Only (Get)	Not available
2	Member_List	Not available	Read Only (Get)	Not available
3	Data	Not available	Read/Write (Set)	Not available

4.8 Dynamic assemblies

Figure 10-4 — Assembly Object State Transition Diagram, Table 10-30 — State Event Matrix and Table 10-31 — Attribute Access Table define the behaviour of Dynamic assemblies.

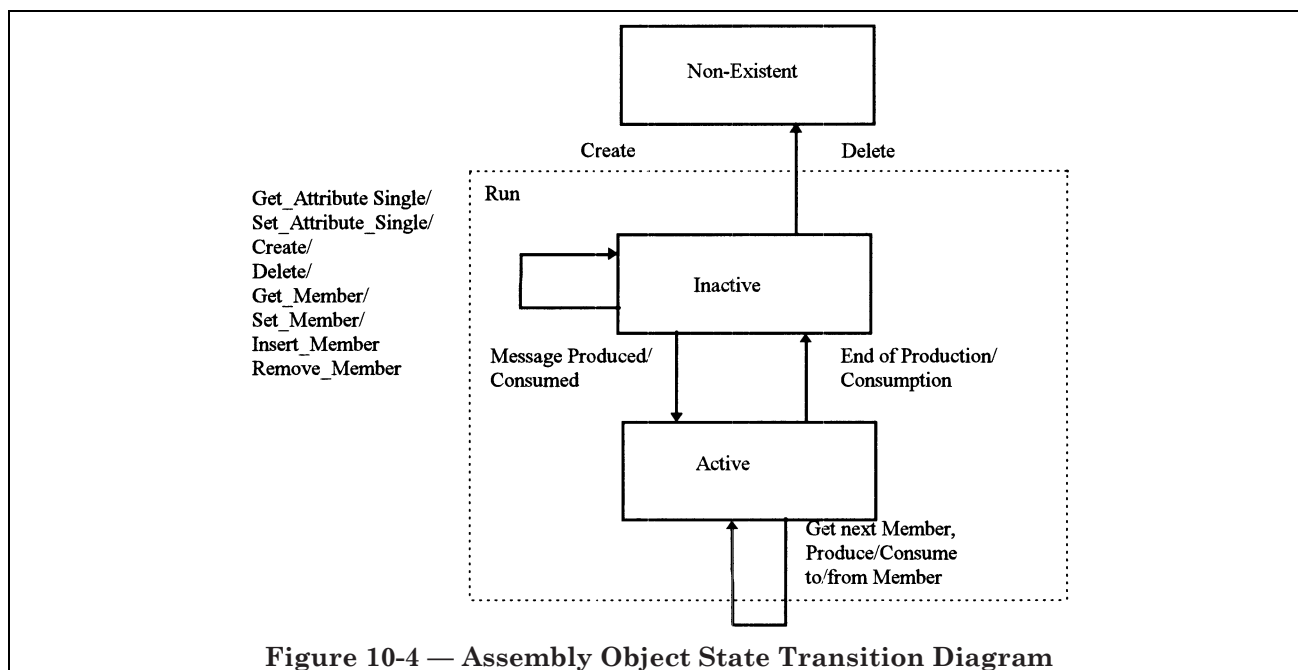


Figure 10-4 — Assembly Object State Transition Diagram

Table 10-30 — State Event Matrix (SEM) for dynamic assembly behaviour

Event	Dynamic Assembly Object state		
	Non-existent	Inactive	Active
Create	Class instantiates an Assembly Object. Transition to Inactive	Not applicable	Not applicable
Delete	Error: Object does not exist. (General Error Code 0x16)	Release all associated resources. Transition to Non-Existent	Error: Object State Conflict (General Error Code 0x0C)
Get_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Set_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Get_Member	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Set_Member	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Insert_Member	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Remove_Member	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)
Message produced/ consumed	Error: Object does not exist. (General Error Code 0x16)	Transition to Active. Begin producing/ consuming from/to each member in list.	Error: Object State Conflict (General Error Code 0x0C)
End of production/ consumption	Error: Object does not exist. (General Error Code 0x16)	Error: Object State Conflict (General Error Code 0x0C)	Transition to Inactive

The attribute access rules for the dynamic assembly shall be as shown in Table 10-31.

Table 10-31 — Dynamic Assembly Object attribute access rules

Instance attribute number	Instance attribute name	Dynamic Assembly Object state		
		Non-existent	Inactive	Active
1	Number_of_members	Not available	Read Only (Get)	Not available
2	Member_list ^a	Not available	Read/Write (Set)	Not available
3	Data	Not available	Read/Write (Set)	Not available

^a This attribute may be set by either the Insert_Member service (one member at a time) or the Set_Attribute_Single service (all members at once).

5 Connection Object (class code = 0x05)

Class code 0x05 shall be reserved.

6 Connection Manager Object (class code = 0x06)

6.1 Scope

The Connection Manager Object is used to manage the establishment and maintenance of communication connections.

6.2 Attributes

6.2.1 Class attributes

The Connection Manager Object shall support the class attributes as specified in Table 10-32.

Table 10-32 — Connection Manager Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	First revision value = 1
2	Optional	Set	Max Instance	UDINT	Maximum instance number	
4	Optional	Get	Optional Attribute List	STRUCT of	List of optional instance attributes used in an object class implementation.	A list of attribute numbers specifying the optional attributes implemented in the device for this class.
			number attributes	UINT	Number of attributes in the optional attribute list.	The number of attribute numbers in the list.
			optional attributes	ARRAY of UINT	List of optional attribute numbers.	The optional attribute numbers.

6.2.2 Instance attributes

The Connection Manager Object shall support the instance attributes as specified in Table 10-33.

Table 10-33 — Connection Manager Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Set	OpenReqs	UINT	Number of Open Requests received including Null Open Requests.	
2	Optional	Set	OpenFormat Rejects	UINT	Number of open requests rejected by this node due to format errors	
3	Optional	Set	OpenResource Rejects	UINT	Number of open requests rejected by this node	
4	Optional	Set	OpenOther Rejects	UINT	Number of open requests rejected or timed out by downstream nodes	
5	Optional	Set	CloseReqs	UINT	Number of close requests received	
6	Optional	Set	CloseFormat Rejects	UINT	Number of close requests rejected by this node due to format errors	
7	Optional	Set	CloseOther Rejects	UINT	Number of close requests rejected or timed out by downstream nodes	
8	Optional	Set	ConnTimeouts	UINT	Number of connections which have been timed out by this node after they were opened	
9	Optional	Get	Connection Entry List	Structure of	List of connections.	
			NumConnEntries	UINT	Number of Conn Array Entries (bit field). This attribute, divided by 8 and incremented for any remainder, gives the length of the array in the Conn Open Bits field.	Number of bits used in the ConnOpenBits Attribute.
			ConnOpenBits	ARRAY of BYTE	List of connection data which may be individually queried by the Get/Search Connection Data Services. Each bit represents a possible connection.	0 = No Connection. 1 = Connection Established. Query for more information.
11	Optional	Get	CpuUtilization	UINT	CPU Utilisation in tenths of a percent.	Range of 0 – 1 000 representing 0 to 100 %.
12	Optional	Get	MaxBuffSize	UDINT	Amount of buffer space originally available.	Size is in Bytes
13	Optional	Get	BufSize Remaining	UDINT	Amount of buffer space available at this time.	Size is in Bytes

NOTE 1 Instance attribute number 10 is unused.

6.3 Common services

The Identity Object shall provide the common services as specified in Table 10-34.

Table 10-34 — Connection Manager Object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Optional	Get_Attributes_All	Returns the contents of all attributes of an object
0x02	n/a	Optional	Set_Attributes_All	
0x03	Optional	Optional	Get_Attributes_List	
0x04	n/a	Optional	Set_Attributes_List	
0x0E	Optional	Optional	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	n/a	Optional	Set_Attribute_Single	Modifies an attribute value

6.4 Object specific services

The Connection Manager Object shall support the object specific services as specified in Table 10-35.

Table 10-35 — Connection Manager Object object specific services

Service name	Service code	Need in implementation		Description of service
		Class	Instance	
Open	0x4B	n/a	Obsolete	
Fwd_Open	0x4C	n/a	Obsolete	
Close	0x4D	n/a	Required	
Fwd_Close	0x4E	n/a	Required	Close a connection
Find	0x4F	n/a	Optional	
Proxy_Open	0x50	n/a	Obsolete	
Proxy_Close	0x51	n/a	Optional	
Unconnected_Send	0x52	n/a	Optional	Unconnected Send Service
Ni_Open	0x53	n/a	Required	Open a connection
Ni_Fwd_Open	0x54	n/a	Required	Open a connection
Ni_Proxy_Open	0x55	n/a	Optional	
Get_Connection_Data	0x56	n/a	Optional	Get information about a specific connection
Search_Connection_Data	0x57	n/a	Optional	Search for information about a connection

7 Register Object (class = 0x07)

7.1 Scope

The Register Object is used to address individual bits or a range of bits up to 64K bits of data. It may operate as either an producer (input) register or a consumer (output) register. A producer register object produces data onto the network. A consumer register object consumes data from the network.

7.2 Attributes

7.2.1 Class Attributes

The Register Object shall support the class attributes as specified in Table 10-36.

Table 10-36 — Register Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	First revision, value = 1

7.2.2 Instance attributes

The Register Object shall support the instance attributes as specified in Table 10-37.

Table 10-37 — Register Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Required	Get	Bad Flag	BOOL	If set to 1 then attribute 4 may contain invalid, bad or otherwise corrupt data.	0 = good 1 = bad
2	Required	Set	Direction	BOOL	Direction of data transfer	0 = Producer Register 1 = Consumer Register
3	Required	Set	Size	UNIT	Size of register data in bits	Size of data in bits.
4	Required	Conditional ^a	Data	ARRAY of BITS	Data to be transferred	See Note 2.

^a The access rule of Set shall be optional if attribute 2, Direction, = 1. If Direction = 0, the access rule shall be Get.

^b All encoded bits shall be LSO aligned.

NOTE See the encoding example in Table 10-38 below.

Table 10-38 — Thirty bit array encoding example-informative

7..... 0	15..... 8	23..... 16	29..... 24	
----------	-----------	------------	------------	--

7.3 Common services

The Register Object shall provide the common services listed in Table 10-39.

Table 10-39 — Register Object common services

Service Code	Need in implementation		Service Name	Description of service
	Class	Instance		
0x0E	Conditional ^a	Required	Get_Attribute_Single	Returns the contents of the specified attribute
0x10	n/a	Optional	Set_Attribute_Single	Modifies an attribute value

NOTE See part 6 for definitions of these common services.

^a The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

7.4 Object-specific Services

The Register Object shall provide no object-specific services.

7.5 Behaviour

Figure 10-5 — Register Object State Transition Diagram and State Event Matrix Table 10-40 shall define the behaviour of the Register Object.

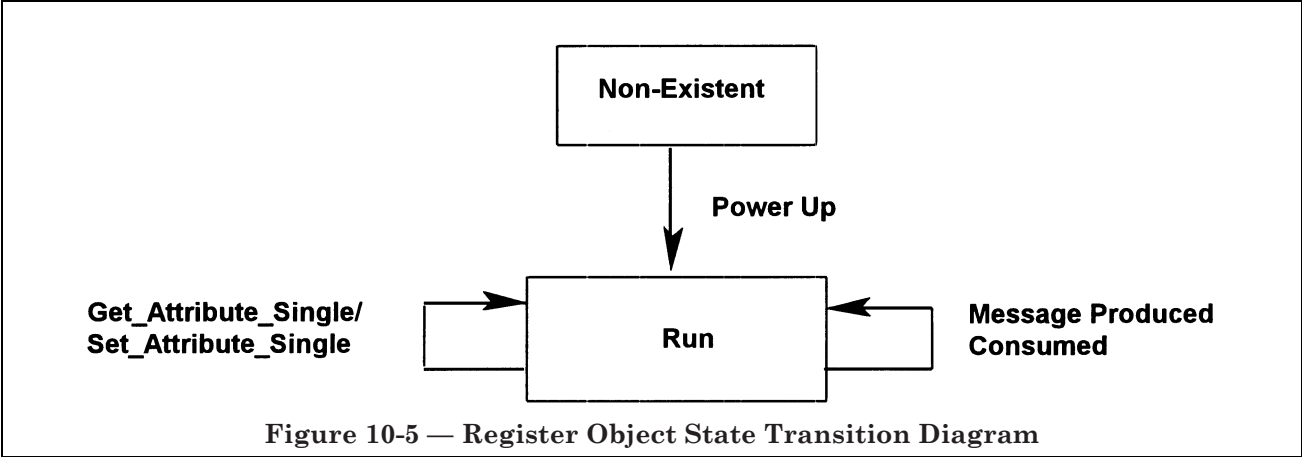


Table 10-40 — Register Object State Event Matrix

Event	State	
	Non-existent	Run
Power up	Transition to Run	Not applicable
Get_Attribute_Single	Error: Object does not exist (General Error code 0x16)	Validate/service the request. Return response.
Set_Attribute_Single	Error: Object does not exist (General Error code 0x16)	Validate/service the request. Return response.
Message produced/consumed	Error: Object does not exist (General Error code 0x16)	Apply/retrieve data

8 Discrete Input Point Object (class code = 0x08)

8.1 Scope

The Discrete Input Point (DIP) Object models discrete inputs in a product. This object may be used in applications as simple as a toggle switch or as complex as a discrete I/O control device. A Discrete Input Point Object shall produce data to the network.

The Discrete Input Point interface shall be to real input points such as a switch or screw terminal. The input shall be sampled and the data is stored in this object’s VALUE attribute. A sample of the discrete input value shall be triggered via an external command (input change-of-state, cyclic data trigger, etc.)

8.2 Revision history

Table 10-41 — Discrete Input Point Object revision history

Revision	Description of changes
01	Initial definition
02	IDLE state removed from object behaviour

8.3 Attributes

8.3.1 Class attributes

The Discrete Input Point object shall support the class attributes as specified in Table 10-42.

Table 10-42 — Discrete Input Point Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Required	Get	Revision	UINT	Revision of this object	The current value assigned to this attribute is two(-02). If updates that require an increase in this value are made, then the value of this attribute increases by 1.
2	Optional	Get	Max Instance	UDINT	Maximum instance number of an object currently created in this class level of the device.	The largest instance number of a created object at this class hierarchy level.
6	Optional	Get	Max ID Number of Class Attributes	UINT	The attribute ID number of the last class attribute of the class definition implemented in the device.	The number of the last class attribute implemented to simplify auto determination of class's implementation by a remote terminal.
7	Optional	Get	Max ID Number of Instance Attributes	UINT	The attribute ID number of the last instance attribute of the class definition implemented in the device.	The number of the last instance attribute implemented to simplify auto determination of class's implementation by a remote terminal.

8.3.2 Instance attributes

The Discrete Input Point Object shall support the instance attributes specified in Table 10-43.

Table 10-43 — Discrete Input Point Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Number of attributes	USINT	Number supported in this product	
2	Optional	Get	Attribute list	UDINT	List of attributes supported in this product	
3	Required	Get	Value	BOOL	Input point value	0 = off 1 = on
4	Optional	Get	Status	BOOL	Input point status	0 = OK 1 = product specific alarm or status
5	Optional	Set	Off_On delay	UINT	filter time for off to on transition 0 – 65,535 microseconds ^a	The default value is 0.
6	Optional	Set	On_Off delay	UINT	filter time for off to on transition 0 – 65,535 microseconds ^b	The default value is 0.

^a The input shall be on for the amount of filter time specified by the OFF_ON DELAY attribute before the ON state is recorded in the VALUE attribute.

^b The input shall be off for the amount of filter time specified by the ON_OFF DELAY attribute before the OFF state is recorded in the VALUE attribute.

8.4 Common services

The Discrete Input Point Object shall provide the common services as specified in Table 10-44.

Table 10-44 — Discrete Input Point Object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Required	Get_Attributes_All	Returns a predefined listing of this objects attributes (see clause 8.5).
0x02	Optional	Required	Get_Attributes_All	Modifies the value of a list of attributes (see clause 8.6).
0x0E	Required	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	n/a	Optional	Get_Attribute_Single	Modifies an attribute value.
NOTE See part 6 for definitions of these common services.				

8.5 Get_Attributes_All response

At the class level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as specified in Table 10-45.

Table 10-45 — Class level object/service specific reply data of Get_Attributes_All

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low byte), Default = 0							
1	Revision (high byte) Default = 0							
2	Max Instance (low byte) Default = 0							
3	Max Instance (high byte) Default = 0							
4	Max ID Number of Class Attributes (low byte) Default = 1							
5	Max ID Number of Class Attributes (high byte) Default = 0							
6	Max ID Number of Instance Attributes (low byte) Default = 3							
7	Max ID Number of Instance Attributes (high byte) Default = 0							

At the Instance level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as specified in Table 10-46.

Table 10-46 — Instance level object/service specific reply data of Get_Attributes_All

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Number of Attributes Default = 0							
1	Attribute List (attribute #1)							
n	Attribute List (attribute #m)							
.n + 1	0	0	0	0	0	0	0	Value
.n + 2	0	0	0	0	0	0	0	Status Default = 0
.	Off_On delay (low byte) Default =0							
.	Off_On delay (high byte) Default =0							
.	Off_On delay (low byte) Default =0							
.	Off_On delay (low byte) Default =0							

If the instance attribute “Number of Attributes” is not supported, the default value of zero shall be inserted in its place and no members of the “Attribute List” attribute shall follow.

8.6 Set_Attributes_All request

At the instance level, the order of attributes passed in the Set_Attributes_All request shall be as specified in Table 10-47.

Table 10-47 — Instance level order

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	off_on delay (low byte)							
1	off_on delay (high byte)							
2	On_Off delay (low byte)							
3	On_Off delay (low byte)							

The Set_Attributes_All service shall be supported only if all settable attributes shown above are implemented as settable.

8.7 Object-specific services

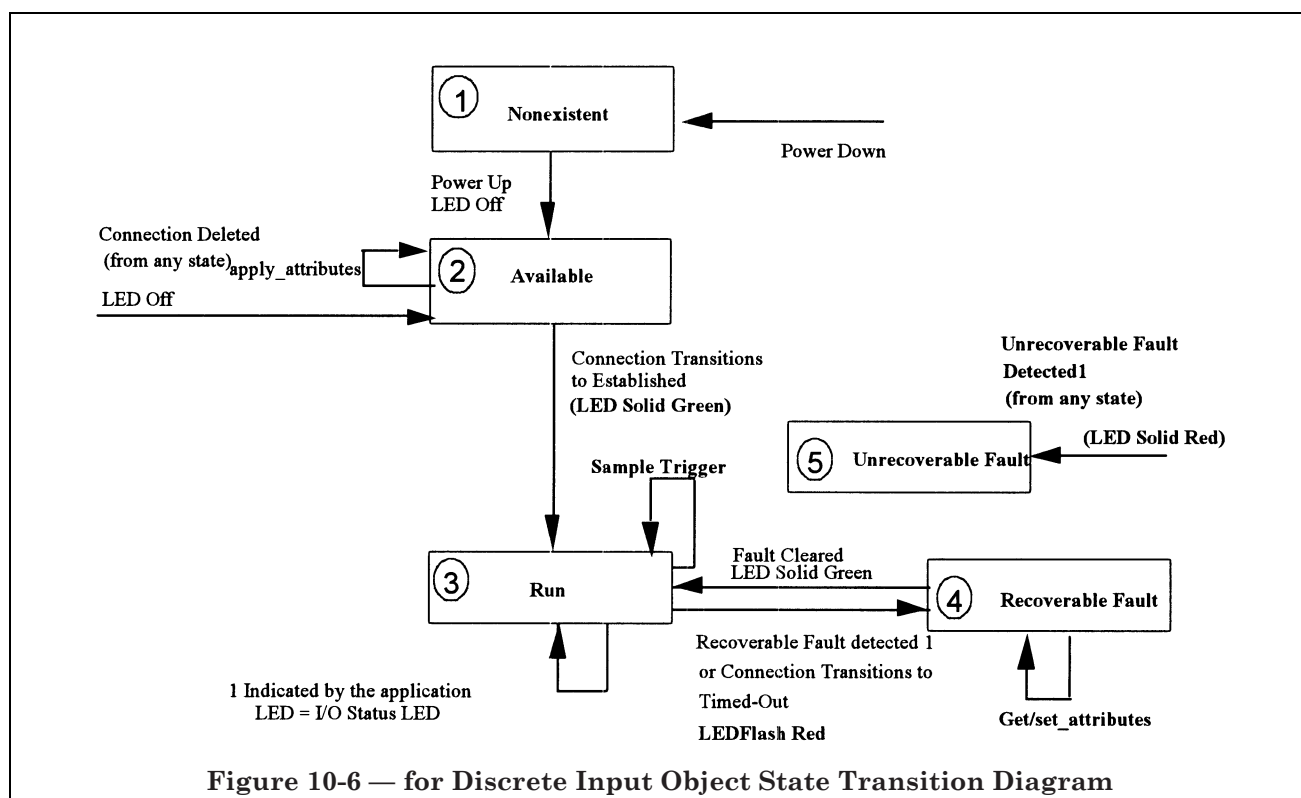
The Discrete Input Point Object shall provide no object-specific services.

8.8 Behaviour

The behaviour of the Discrete Input Point shall be as defined in Figure 10-6. A subset of the states and events may be supported in an application, but the behavior shall be consistent.

Table 10-48 and Table 10-50 list all pertinent events and the corresponding action which shall be taken while in each state.

Events may occur simultaneously, but the Fault events shall have priority if they occur simultaneously with other events.



The SEM shall contain the states as specified in Table 10-48.

Table 10-48 — Discrete Input Object SEM states

States	Description
Non-existent	the device without power
Available	waiting for a connection, power-up discrete input point defaults are set
Run	sensing data from its input and transmitting the data
Recoverable fault	a recoverable fault has occurred
Unrecoverable fault	an unrecoverable fault has occurred

The SEM for the Discrete Input Point Object shall contain the events as specified in Table 10-49.

Table 10-49 — Discrete Input Point Object STD events

Event	Description
Sample trigger	A change of state, cyclic timer trigger, application trigger
Connection deleted	I/O connection deleted
Apply_Attributes	The apply service of the I/O connection object the Discrete Input Point object shall be connected to. NOTE The application is responsible for validating the connection object's attributes.
Fault cleared	The application clearing a detected fault
Connection transitions to Established	I/O connection transitions to Established
Connection transitions to timed out state	I/O connections transitions to Timed-out

Figure 10-7 is a conceptual definition of the state machine for a typical input object (producing application). The events listed above are represented by the dotted line labelled “state changed.”

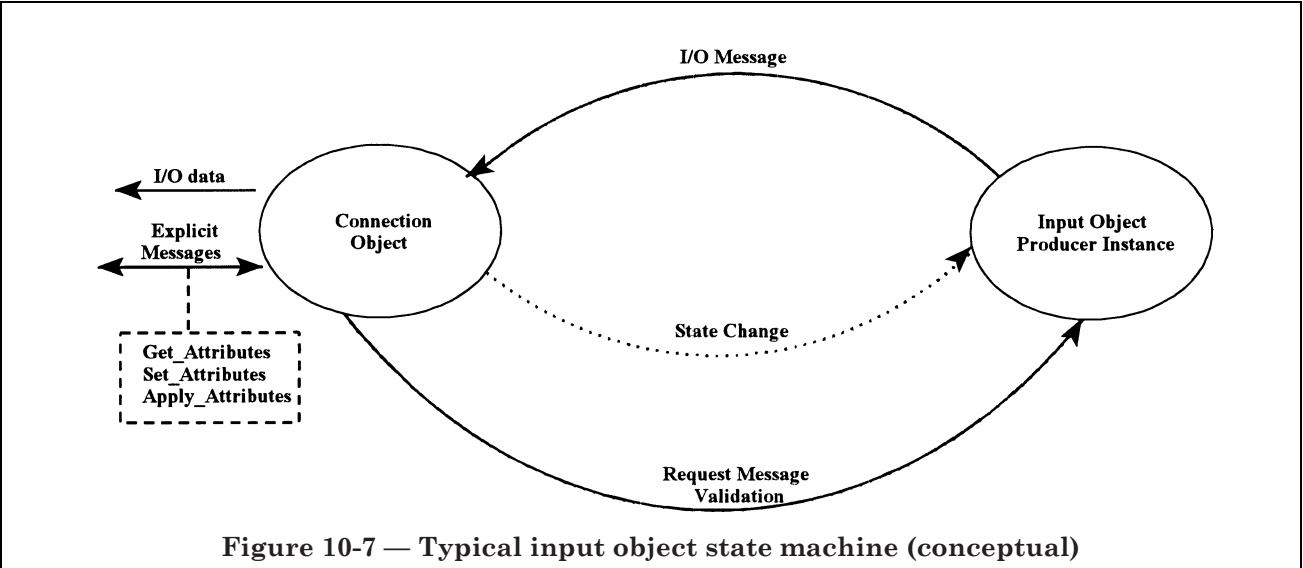


Table 10-50 — Discrete Input Point Object SEM events

Event	Non-existent	Available	State run	Recoverable Fault	Unrecoverable fault
Sample Trigger	Not Applicable	Ignore event	Sample data, Send data	Ignore Event	Ignore Event
Apply attributes	Not Applicable	Verify attributes, return results	Return error (Object State Conflict)	Return error (Object State Conflict)	Ignore Event
Connection detected	Not Applicable	Ignore Event	Transition to available	Transition to available	Ignore Event
Connections Transitions to Established	Not Applicable	Transition to Run	Ignore Event	Ignore event	Ignore Event
Connection Transitions to Timed Out state	Not Applicable	Ignore Event	Transition to recoverable fault	Ignore Event	Ignore Event
Fault Cleared	Not Applicable	Not Applicable	Not Applicable	Transition to Run	Ignore Event
Get_Attribute	Return Error (Object does not exist)	Return value	Return value	Return value	Ignore Event
Set_Attribute	Return Error (Object does not exist)	Accept value	Accept value	Accept value	Ignore Event
I/O Status LED	Off	Off	Solid Green	Flash Red	Solid Red

8.9 Attribute access rules

Except in the Non-existent and Unrecoverable Fault states, all attributes are gettable or settable according to their access rules. Because the only required Instance attribute is value, the only required behaviour of a Discrete Input Point is that it indicates a Boolean value of OFF or ON. The optional attributes shall either provide more information about the discrete input point or alter the behaviour of the input point.

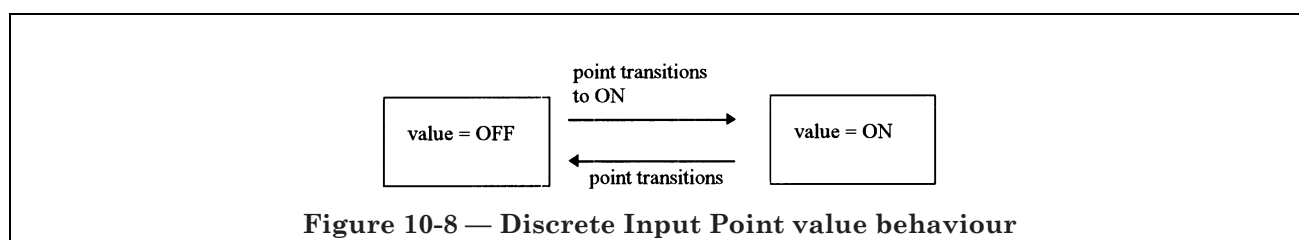


Figure 10-8 — Discrete Input Point value behaviour

The optional status instance attributes shall be simply a logical OR of all possible failure or alarm condition for the point.

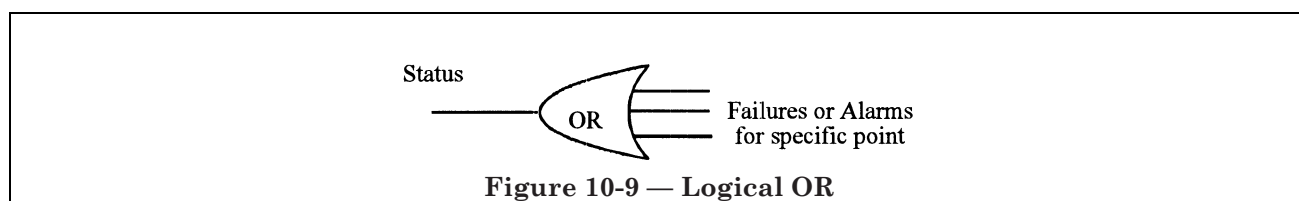


Figure 10-9 — Logical OR

9 Discrete Output Point Object (class code = 0x09)

9.1 Scope

A Discrete Output Point (DOP) shall model discrete outputs in a product. This object may be used in applications such as discrete I/O control devices, relays, switches, etc. An output shall consume data from the network.

The Discrete Output Point interface is to real output points such as a relay or LED. The output shall be read from this object's VALUE attribute and applied to the output terminal (e.g. screw terminal).

9.2 Attributes

9.2.1 Class attributes

The Discrete Output Object shall support the class attributes as specified in Table 10-51.

Table 10-51 — Discrete Output Point Object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	Revision, value = 1
2	Optional	Get	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device.	The largest instance number of a created object at this class hierarchy level.
6	Optional	Get	Max ID Number of Class Attributes	UINT	The attribute ID number of the last class attribute of the class definition shall be implemented in the device.	The number of the last class attribute shall be implemented to simplify auto determination of class's implementation by a remote terminal.
7	Optional	Get	Max ID Number of Instance Attributes	UINT	The attribute ID number of the last instance attribute of the class definition shall be implemented in the device.	The number of the last instance attribute shall be implemented to simplify auto determination of class's implementation by a remote terminal.

9.2.2 Instance attributes

The Discrete Output Point Object shall support the instance attributes specified in Table 10-52.

Table 10-52 — Discrete Output Point Object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
1	Optional	Get	Number of Attributes	USINT	Number of attributes supported in this product	
2	Optional	Get	Attribute List	ARRAY OF USINT	List of attributes supported in this product	
3	Required	Set	Value	BOOL	Output point value	0 = off; 1 = on
4	Optional	Get	Status	BOOL	Output point status	0 = OK 1 = failure or alarm
5	Optional	Set	Fault Action	BOOL	Action taken on output's value in Recoverable Fault state	0 = Fault Value attribute 1 = hold last state
6	Optional	Set	Fault Value	BOOL	User-defined value for use with Fault State attribute	0 = off 1 = on
7	Optional	Set	Idle Action	BOOL	Action taken on output's value in Recoverable Fault state	0 = Idle Value attribute 1 = hold last state
8	Optional	Set	Idle Value	BOOL	User-defined value for use with Idle State attribute	0 = off 1 = on
9	Optional	Set	Run_Idle_Command	BOOL	Generates the Receive_Idle or Receive_Ready_to_Run event	0 = Receive_Idle 1 = Receive_Ready_to_Run
10	Optional	Set	Flash	BOOL	Flash output at periodic rate if point is ON	0 = no flash; 1 = flash
11	Optional	Set	Flash Rate	USINT	Flash Rate for Flash attribute	unsigned positive integer indicating frequency in Hz, e.g. 1 = 1Hz
12	Optional	Get	Object State	USINT	State of the object	1 = Non-existent 2 = Available 3 = Idle 4 = Ready 5 = Run 6 = Recoverable Fault 7 = Unrecoverable Fault 255 = Reserved

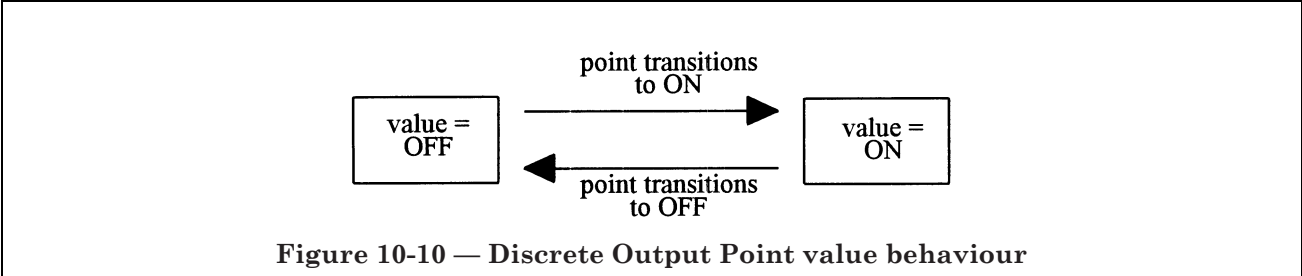
Optional attributes shall either provide more information about the discrete output point or alter the behaviour of the output point. If the optional instance attributes specified in Table 10-53 are not supported, the default attribute values shall indicate the required behaviour for instances of this object class.

Table 10-53 — Discrete Output Point Object optional instances attributes

Attribute ID	Name	Default value
5	Fault Action	0
6	Fault Value	0
7	Idle Action	0
8	Idle Value	0
10	Flash	0

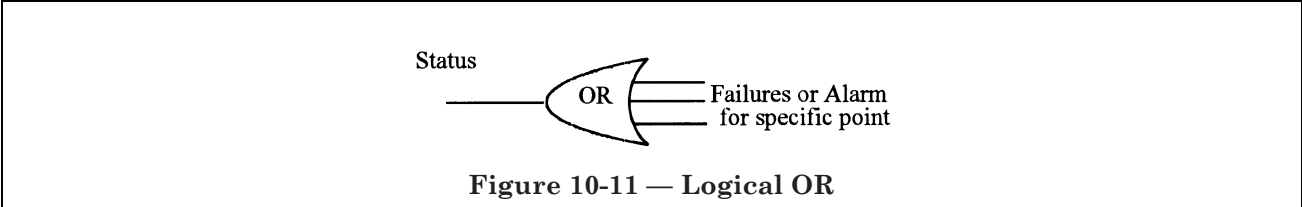
9.2.3 Value

The only required Instance attribute is **Value**. The required behaviour of a Discrete Output Point’s value is that it outputs a Boolean value of OFF or ON.



9.2.4 Status

The **Status** attribute shall simply be a logical OR of all possible failure or alarm conditions for the point.



9.2.5 Fault and idle attributes

These optional attributes define a safe state for the DOP when in the Idle or Recoverable Fault states:

- Fault Action;
- Fault Value;
- Idle Action;
- Idle Value.

Table 10-54 — Safe state values

This attribute pair:	Defines:
Fault Action and Fault Value	the value of the DOP in the Fault state.
Idle Action and Idle Value	the value of the DOP in the Idle state.

The dependencies exist among these attributes as specified in Table 10-55.

Table 10-55 — Attribute dependencies

If this attribute is supported:	Then this attribute should also be supported by definition:
Fault Action	Fault Value (although this attribute could be defined to always be OFF).
Idle Action	Idle Value (although this attribute could be defined to always be OFF).

The “Action” attributes shall dictate what the DOP shall do upon entering that state. The DOP may either hold its value in the last state or update it to the value stored in the corresponding Fault or Idle Value attribute.

Upon entering the recoverable_fault state the DOP shall behave according to Table 10-56.

Table 10-56 — DOP behaviour

	Fault_State = 0	Fault_State = 1
Fault_Value = 0	DOP uses the value in the Fault_Value attribute (0) to update its value.	DOP leaves Value in last state. Fault_Value attribute has no affect.
Fault_Value = 1	DOP uses the value in the Fault_Value attribute (1) to update its value.	DOP leaves Value in last state. Fault_Value attribute has no affect.

The Idle attributes follow similar behaviour.

There shall be one deviation from the behaviour specified in the table above. If the DOP enters the Recoverable_Fault state from the Idle state in response to the I/O connection transitioning to Timed Out, the DOP's value shall go unchanged. This shall be shown in the DOP's State Transition Diagram.

9.2.6 Run_Idle command

The **Run_Idle command** attribute causes the Receive_Ready_to_Run or Receive_Idle event to be sent to the DOP. Refer to the DOP's State Transition Diagram to see the resulting transitions. This attribute shall only have effect when the Discrete Output object is in the **Idle**, **Ready**, or **Run** states. While in the **Available** or **Recoverable Fault** state, an attempt to set this attribute shall result in an Object_State_Conflict error. A read (Get_Attribute_Single) of this attribute shall result in a zero being returned always.

9.2.7 Flash

The **Flash** attribute modifies the behaviour of the DOP such that when the DOP is in the ON state, the output flashes OFF and ON at a periodic rate.

9.2.8 Flash Rate

The **Flash Rate** attribute modifies the behaviour of the DOP in that when the DOP is in the ON state and the Flash attribute is ON, it sets the periodic flash rate by an unsigned positive integer.

9.3 Common services

The Discrete Output Object shall provide the common services as specified in Table 10-57.

Table 10-57 — Discrete Output Point Object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below)
0x02	n/a	Optional	Set_Attributes_All	Modifies the contents of the attributes of the class or object.
0x10	n/a	Required	Set_Attribute_Single	Modifies an attribute value.
0x0E	Optional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
NOTE See part 6 for definitions of these common services				
^a The Get_Attribute_Single service is REQUIRED if any attributes are implemented.				

9.4 Get_Attributes_All response

At the class level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as specified in Table 10-58.

Table 10-58 — Class level object/service specific reply data for Get_Attributes_All

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low byte) Default = 1							
1	Revision (high byte) Default = 0							
2	Max Instance (low byte) Default = 0							
3	Max Instance (high byte) Default = 0							
4	Max ID Number of Class Attributes (low byte) Default = 0							
5	Max ID Number of Class Attributes (high byte) Default = 0							
6	Max ID Number of Instance Attributes (low byte) Default = 3							
7	Max ID Number of Instance Attributes (high byte) Default = 0							

Insert default values for all unsupported attributes.

At the instance level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get_Attributes_All response shall be as specified in Table 10-59.

Table 10-59 — Instance level object/service specific reply data for Get_Attributes_All

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Number of Attributes Default = 0							
1	Attribute List (attribute #1)							
n	Attribute List (attribute #m)							
n+1	0	0	0	0	0	0	0	Value
n+2	0	0	0	0	0	0	0	Status Default = 0
·	0	0	0	0	0	0	0	Fault State Default = 0
·	0	0	0	0	0	0	0	Fault Value Default = 0
·	0	0	0	0	0	0	0	Idle State Default = 0
·	0	0	0	0	0	0	0	Idle Value Default = 0
·	·	·	·	·	·	·	·	·
·	0	0	0	0	0	0	0	Flash Default = 0
·	Flash Rate Default = 0							
·	Object State Default = 255							

If the instance attribute “Number of Attributes” is not supported, the default value of zero shall be inserted in its place and no members of the “Attribute List” attribute shall follow. Insert default values for all unsupported attributes.