



<AutomationML/>

**The Glue for Seamless
Automation Engineering**

**Whitepaper AutomationML
Part 4: AutomationML Logic**
Document Identifier: WP Logic, V 1.5.0

State: January 2017

© AutomationML consortium

Version 1.5.0 January 2017

Contact: www.automationml.org

Table of content

Table of content.....	3
List of Figures	9
List of Tables	11
1 Introduction.....	14
1.1 Scope	15
1.2 Normative references.....	15
2 Terms, definitions and abbreviations	17
2.1 Terms and definitions.....	17
2.1.1 logic information	17
2.1.2 logic model	17
2.1.3 IEC 61131-XML.....	17
2.1.4 sequencing information	17
2.1.5 sequencing model	17
2.1.6 behaviour information.....	17
2.1.7 behaviour model.....	17
2.1.8 interlocking information	17
2.1.9 interlocking model	17
2.1.10 logic objects.....	17
2.1.11 mathematical expression	17
2.1.12 state chart.....	17
2.2 Abbreviations	18
2.3 Conformity.....	18
3 Conventions	19
3.1 General	19
3.2 Conventions for the definition of the Intermediate Modelling Layer	19
3.3 Mapping between logic models and the Intermediate Modelling Layer.....	19
3.4 Mapping rules between the Intermediate Modelling Layer and IEC 61131-XML	19
4 Part 4 related extensions of AMLlibraries	20
4.1 General	20
4.2 AutomationMLBaseRoleClassLib	20
4.3 General	20
4.3.1 RoleClass InterlockingTargetGroup	21
4.3.2 RoleClass InterlockingSourceGroup	21
4.3.3 RoleClass LogicObject.....	21
4.4 AutomationMLInterfaceClassLib	22
4.4.1 General.....	22
4.4.2 InterfaceClass LogicInterface.....	23
4.4.3 InterfaceClass SequencingLogicInterface	23
4.4.4 InterfaceClass BehaviourLogicInterface	24
4.4.5 InterfaceClass SequencingBehaviourLogicInterface	24
4.4.6 InterfaceClass InterlockingLogicInterface	25
4.4.7 InterfaceClass LogicObjectInterface	25
4.4.8 InterfaceClass VariableInterface	26
4.4.9 InterfaceClass InterlockingConnector	26

4.4.10	InterfaceClass InterlockingVariableInterface	27
5	Meta informationabout the IEC 61131-XML source tool.....	28
6	Referencing IEC 61131-XML documents	29
6.1	General	29
6.2	Referencing logic information	29
7	Intermediate Modelling Layer.....	31
7.1	General	31
7.2	IML system overview	31
7.3	Definition of IML system elements.....	31
7.3.1	“Header”	31
7.3.2	“State”	32
7.3.3	“State transition”	32
7.3.4	“Activity”	33
7.3.5	“Selection divergence”	34
7.3.6	“Simultaneous divergence”	34
7.3.7	“Selection convergence”	34
7.3.8	“Simultaneous convergence”	34
7.3.9	“Event”	35
7.3.10	“Variable”	35
7.3.11	“Comment”	35
7.3.12	“Additional data”	35
8	Mapping of IML to IEC 61131-XML.....	37
8.1	General	37
8.2	AML addData schema for additional data within logic information	37
8.2.1	Declaration and usage of AML addDataSchema in IEC 61131-XML.....	37
8.2.2	XML schema description ofAMLspecific data	38
8.2.3	XML schema description of timing data	39
8.2.4	XML schema description of durations regarding timing data	40
8.2.5	XML schema description of earliest starts regarding timing data	41
8.2.6	XML schema description of latest starts regarding timing data	42
8.2.7	XML schema description of earliest ends regarding timing data	43
8.2.8	XML schema description of latest ends regarding timing data	44
8.2.9	XML schema description of delays regarding timing data	45
8.2.10	XML schema description of chart types	46
8.2.11	XML schema description of change definitions of resource states	47
8.2.12	XML schema description of interruptible actions	47
8.2.13	XML schema description of sub chartsin state charts.....	48
8.2.14	XML schema description of state types of state charts.....	49
8.2.15	XML schema description of actions types of state charts	50
8.2.16	XML schema description of resource groups of timing diagrams	51
8.2.17	XML schema description of PLC variables of timing diagrams.....	52
8.2.18	XML schema description of state status	53
8.2.19	XML schema description of action status	54
8.2.20	XML schema description of units of measurement	55
8.2.21	XML schema description of addDataBaseObject	55
8.3	Mapping of IML to IEC 61131-XML SFC	56

8.3.1	Common rules	56
8.3.2	Mapping of headers	58
8.3.3	Mapping of states	59
8.3.4	Mapping of state transitions	60
8.3.5	Mapping of activities	66
8.3.6	Mapping of selection divergences	70
8.3.7	Mapping of simultaneous divergences	71
8.3.8	Mapping of selection convergences	72
8.3.9	Mapping of simultaneous convergences	73
8.3.10	Mapping of events	74
8.3.11	Mapping of variables	75
8.3.12	Mapping of comments	77
8.3.13	Mapping of additional data	78
9	Usage of mathematical expressions in logic information	80
9.1	General	80
9.2	General integration concept	80
9.3	Mapping between MathML variables and IEC 61131-XMLElements "variable"	81
9.3.1	General	81
9.3.2	Declaration and usage of the XML schema for variable mappings for MathML in IEC 61131-XML	82
9.3.3	XML schema description of formula data	83
9.3.4	XML schema description of variable data	83
9.4	Assignment of MathML expressions to IEC 61131-XML elements	84
9.5	Assignment of results of a MathML expression to IEC 61131-XMLElements "variable"	84
10	Linking AML objects with interlocking information	86
10.1	General	86
10.2	Referencing interlocking information	86
Annex A	Logic information in AML	88
A.1	Logic information in production system engineering	88
A.2	Logic models in production system engineering	89
A.3	Storing logic models in AML	90
A.4	Storing additional information to logic models in AML	92
1.1.1	Additional logic model specific information	92
1.1.2	Meta information on file level	92
1.1.3	Mathematical expressions	92
A.5	Referencing logic information in AML	92
Annex B	Logic models in AML	93
B.1	General	93
B.2	Gantt charts	93
B.2.1	General	93
B.2.2	Model elements	93
B.2.3	Chart structuring	93
B.2.4	Amount of information	94
B.2.5	Mapping rules	94
B.3	Activity-on-node networks	94
B.3.1	General	94
B.3.2	Model elements	94

B.3.3	Network structuring	95
B.3.4	Amount of information	95
B.3.5	Mapping rules	96
B.4	Timing diagrams	96
B.4.1	General	96
B.4.2	Model elements	96
B.4.3	Diagram structuring	97
B.4.4	Amount of information	97
B.4.5	Mapping rules	97
B.5	State charts	98
B.5.1	General	98
B.5.2	Model elements	98
B.5.3	Sequential function charts	99
B.5.4	Function block diagrams	99
Annex C	Mapping of logic models	100
C.1	General	100
C.2	Mapping of Gantt charts to IML	100
C.2.1	Common rules	100
C.2.2	Mapping of the start	101
C.2.3	Mapping of bars	102
C.2.4	Mapping of bar start points	102
C.2.5	Mapping of bar end points	103
C.2.6	Mapping of successor bars	103
C.2.7	Mapping of predecessor bars	104
C.2.8	Mapping of the end	106
C.3	Mapping of activity-on-node networks to IML	107
C.3.1	Common rules	107
C.3.2	Mapping of the start of an activity-on-node network	108
C.3.3	Mapping of nodes	109
C.3.4	Mapping of node start points	110
C.3.5	Mapping of node end points	110
C.3.6	Mapping of successor nodes	110
C.3.7	Mapping of predecessor nodes	111
C.3.8	Mapping of the end of an activity-on-node network	113
C.4	Mapping of timing diagrams to IML	114
C.4.1	Common rules	114
C.4.2	Mapping of the start of a timing diagram	115
C.4.3	Mapping of the timeline	116
C.4.4	Mapping of resources	116
C.4.5	Mapping of resource states	117
C.4.6	Mapping of signals	119
C.4.7	Mapping of the end of a timing diagram	121
C.4.8	Mapping of timing diagram details	121
C.5	Mapping of state charts	123
C.5.1	Common rules	123
C.5.2	Mapping of regions	124

C.5.3	Mapping of states	125
C.5.4	Mapping of successors of states	125
C.5.5	Mapping of predecessor states	126
C.5.6	Mapping of actions	126
C.5.7	Mapping of events	128
C.5.8	Mapping of signals	128
C.5.9	Mapping of state transitions	128
C.5.10	Mapping of history connectors	135
C.5.11	Mapping of condition connectors	135
Annex D	Referencing methods for logic information	136
D.1	General.....	136
D.2	Referencing logic information expressed as logic models	136
D.2.1	Referencing logic information stored in one POU	136
D.2.2	Referencing logic information distributed throughout several POUs	137
D.2.3	Referencing interlocking information distributed throughout several IEC 61131-XML documents	137
D.3	Referencing logic information as a part of logic models	140
D.3.1	Referencing a variable	141
D.3.2	Referencing an interlocking variable	142
D.3.3	Referencing a logic object	143
D.4	Referencing logic information as a part of already referenced logic models	145
Annex E	Using mathematical expressions in logic information.....	148
E.1	General.....	148
E.2	Example description	148
E.3	Mathematical functions expressed in MathML for logic information	148
E.4	Integration positions in IEC 61131-XML for “addData” elements including MathML expressions	152
Annex F	Referencing interlocking information	153
F.1	General.....	153
F.2	Example description	153
F.3	Interlocking information	154
F.4	Referencing interlocking information without interlocking condition	154
F.5	Referencing interlocking information with interlocking condition	156
Annex G	Example for the mapping of logic models to IML	161
G.1	Mapping Examples of Gantt charts	161
G.1.1	Mapping of activities without predecessor and successor relation	161
G.1.2	Mapping of an activity sequence	161
G.1.3	Mapping of an activity sequence with divergences	162
G.1.4	Mapping of an activity sequence with convergences	163
G.2	Mapping examples of activity-on-node networks	163
G.2.1	Mapping of an activity sequence	163
G.2.2	Mapping of an activity sequence with divergences and convergences	164
G.2.3	Mapping of a complex activity sequence with divergences and convergences	166
G.3	Mapping examples of timing diagrams	167
G.3.1	Example transformation internal signal	168
G.3.2	Example External Signals	169
G.3.3	Example Signal Between two Resource States Flows	170

G.4	Mapping examples of state charts	171
G.4.1	Mapping of a simple cyclic state chart	171
G.4.2	Mapping of a state chart with states with different predecessors and successors	172
G.4.3	Mapping of a state chart with actions.....	173
G.4.4	Mapping of a state chart with a condition connector	174
G.4.5	Mapping of a state chart with simple hierarchy	175
G.4.6	Mapping of a state chart with a complex hierarchy and connectors	176
Annex H	XML representation of AML libraries	179
H.1	AutomationMLBaseRoleClassLib	179
H.2	AutomationMLInterfaceClassLib	180
Annex I	XML representation of schemata	182
I.1	AML_addData	182
I.2	MathMLinIEC61131-XML.....	186

List of Figures

Figure 1: Overview of the engineering data exchange format AML	14
Figure 2: AutomationMLBaseRoleClassLib	20
Figure 3: AutomationMLInterfaceClassLib.....	22
Figure 4: IML system entities and its relations.....	31
Figure 5: XML text of declaration of AML schema for storing additional logic information within IEC 61131-XML documents	37
Figure 6: Used parts of the IEC 61131-XML schema to represent a IML system	57
Figure 7: Mapping concept for MathML variables and IEC 61131-XML elements “variable”	81
Figure 8: XML text of the mapping of MathML variables and IEC 61131-XML elements “variable”	82
Figure 9: XML text of the declaration of the XML schema for variable mappings for MathML in IEC 61131-XML.....	82
Figure 10: XML text of the IEC 61131-XML element “data” for the MathML expression	84
Figure 11: Decision tree for the assignment of results of MathML expressions to IEC 61131- XML elements “variable”	85
<i>Figure A.1: Types of logic information in AML.....</i>	<i>88</i>
<i>Figure A.2: Logic models in AML.....</i>	<i>90</i>
<i>Figure A.3: Transformation to IEC 61131-XML</i>	<i>90</i>
<i>Figure A.4: Scope of IEC 62714-4 regarding mapping rules.....</i>	<i>91</i>
Figure B.1: Model elements of Gantt charts	93
Figure B.2: Information provided by Gantt charts	94
Figure B.3: Model elements of activity-on-node networks	95
Figure B.4: Information provided by activity-on-node networks.....	96
Figure B.5: Model elements of timing diagrams	96
Figure B.6: Information provided by timing diagrams	98
Figure B.7: Model elements of state charts	99
Figure C.1: Actions of IML for Gantt bar representation	101
Figure D.1: Referencing logic information (as SFC) stored in one POU	136
Figure D.2: XML text of the CAEX file for referencing logic information stored in one POU	136
Figure D.3: Referencing logic information distributed throughout several POUs	137
Figure D.4: Referencing interlocking information distributed throughout several IEC 61131- XML documents	138
Figure D.5: XML text of the CAEX file for referencing interlocking information distributed throughout several IEC 61131-XML documents.....	138
Figure D.6: Referencing interlocking information distributed throughout several IEC 61131- XML documents using CAEX means.....	139
Figure D.7: XML text of the CAEX file for referencing interlocking information distributed throughout several IEC 61131-XML documents using CAEX means	140
Figure D.8: Referencing a variable	141
Figure D.9: XML text of the CAEX file for referencing a variable	142
Figure D.10: Referencing an interlocking variable	142
Figure D.11: XML text of the CAEX file for referencing an interlocking variable	143
Figure D.12: Referencing a logic object.....	144
Figure D.13: XML text of the CAEX file for referencing a logic object	144
Figure D.14: Referencing a variable of an already referenced logic model	146

Figure D.15: XML text of the CAEX file for referencing a variable of an already referenced logic model.....	147
Figure E.1: Diagram of an one-way flow control valve with exhaust-air flow control.....	148
Figure E.2: Flow rate of valves	148
Figure E.3: “data” element for describing the MathML expression of the flow rate of a control valve.....	151
Figure E.4: “data” element for describing the mapping of MathML variables for calculating the flow rate of a control valve to IEC 61131-XML variables.....	151
Figure E.5. “addData” element including the MathML expression for the flow rate of a control valve and the mapping of MathML variables to IEC 61131-XML variables.....	152
Figure E.6: Integration of the “addData” element including a MathML expression about the flow rate of a control valve in a POU	152
Figure F.1: Example manufacturing system	153
Figure F.2: Example interlocking source group and interlocking target group	154
Figure F.3: Referencing interlocking information without interlocking condition.....	155
Figure F.4: XML text of the CAEX file for referencing interlocking information without interlocking condition	156
Figure F.5 – Referencing interlocking information with interlocking condition.....	157
Figure F.6 – XML text of the CAEX file for referencing interlocking information with interlocking condition	159
Figure F.7 – Linking logical interface with physical interface (extension to Figure F.5)	159
Figure F.8 – XML text of the CAEX file for linking logical interface with physical interface (extension to Figure F.6).....	160
Figure G.9: Example of an activity-on-node network.....	166
Figure G.10. Timing behaviour of the SFC derived from an activity-on-node network.....	167

List of Tables

Table 1: Abbreviations	18
Table 2: RoleClass InterlockingTargetGroup.....	21
Table 3: RoleClass InterlockingSourceGroup.....	21
Table 4: RoleClass LogicObject.....	21
Table 5: InterfaceClass LogicInterface	23
Table 6: InterfaceClass SequencingLogicInterface	23
Table 7: InterfaceClass BehaviourLogicInterface	24
Table 8: InterfaceClass SequencingBehaviourLogicInterface	24
Table 9: InterfaceClass InterlockingLogicInterface	25
Table 10: InterfaceClass LogicObjectInterface	25
Table 11: InterfaceClass VariableInterface.....	26
Table 12: InterfaceClass InterlockingConnector	26
Table 13: InterfaceClass InterlockingVariableInterface	27
Table 14: Meta information about the IEC 61131-XML source tool.....	28
Table 15: Overview of all possible combinations of execution conditions	33
Table 16: AML schema element “AML”	38
Table 17: AML schema element “Time”.....	39
Table 18: AML schema element “Duration” (within the AML schema element “Time”).....	40
Table 19: AML schema element “EarliestStart” (within the AML schema element “Time”).....	41
Table 20: AML schema element “LatestStart” (within the AML schema element “Time”).....	42
Table 21: AML schema element “EarliestEnd” (within the AML schema element “Time”).....	43
Table 22: AML schema element “LatestEnd” (within the AML schema element “Time”).....	44
Table 23: AML schema element “Delay” (within the AML schema element “Time”)	45
Table 24: AML schema element “ChartType”	46
Table 25: AML schema element “ResourceStateChangeDefinition”	47
Table 26: AML schema element “InterruptibleAction”	47
Table 27: AML schema element “StateChartSubCharts”	48
Table 28: AML schema element “StateChartStateType”	49
Table 29: AML schema element “StateChartActionType”	50
Table 30: AML schema element “TimingDiagramResourceGroup”	51
Table 31: AML schema element “TimingDiagramPLCVariable”	52
Table 32: AML schema element “StateStatus”	53
Table 33: AML schema element “ActionStatus”	54
Table 34: AML schema element “Unit”	55
Table 35: AML schema element “addDataBaseObject”	55
Table 36: Mapping of the IML system element “header” to IEC 61131-XML	58
Table 37: Example transformation IML system element “header” to IEC 61131-XML	58
Table 38: Mapping of the IML system element “state” to IEC 61131-XML.....	59
Table 39: Example transformation IML system element “state” to IEC 61131-XML	60
Table 40: Mapping of the IML system element “state transition” to IEC 61131-XML	64
Table 41: Example transformation IML system element “state transition” to IEC 61131-XML.....	65
Table 42: Mapping of the IML system element “activity” to IEC 61131-XML	69
Table 43: Example transformation IML system element “activity” to IEC 61131-XML	70
Table 44: Mapping of the IML system element “selection divergence” to IEC 61131-XML	70

Table 45: Example transformation IML system element “selection divergence” to IEC 61131-XML.....	71
Table 46: Mapping of the IML system element “simultaneous divergence” to IEC 61131-XML	71
Table 47: Example transformation IML system element “simultaneous divergence” to IEC 61131-XML	72
Table 48: Mapping of the IML system element “selection convergence” to IEC 61131-XML	72
Table 49: Example transformation IML system element “selection convergence” to IEC 61131-XML	73
Table 50: Mapping of the IML system element “simultaneous convergence” to IEC 61131-XML.....	73
Table 51: Example transformation IML system element “simultaneous convergence” to IEC 61131-XML	74
Table 52: Mapping of the IML system element “event” to IEC 61131-XML.....	74
Table 53: Example transformation IML system element “event” to IEC 61131-XML	74
Table 54: Mapping of the IML system element “variable” to IEC 61131-XML.....	76
Table 55: Example transformation IML system element “variable” to IEC 61131-XML.....	76
Table 56: Mapping of the IML system element “comment” to IEC 61131-XML.....	77
Table 57: Example transformation IML system element “comment” to IEC 61131-XML	77
Table 58: Mapping of the IML system element “additional data” to IEC 61131-XML	78
Table 59: Example transformation IML system element “additional data” to IEC 61131-XML.....	79
Table 60: Structure of the IEC 61131-XML element "addData"	80
Table 61: MathML schema element “formula”	83
Table 62: MathML schema element “variable”	83
Table B.1: Mapping of model elements of state charts to model elements of state machines	99
Table C.1: Mapping of the start of a Gantt chart to IML system elements	102
Table C.2: Mapping of Gantt chart bars to IML system elements	102
Table C.3: Mapping of a Gantt chart bar start point to IML system elements	103
Table C.4: Mapping of a Gantt chart bar end point to IML system elements	103
Table C.5: Mapping of Gantt chart successor bars to IML system elements	104
Table C.6: Mapping of Gantt chart predecessor bars to IML system elements	106
Table C.7: Mapping of the end of a Gantt chart to IML system elements	107
Table C.8: Mapping of the start of an activity-on-node network to IML system elements	109
Table C.9: Mapping of activity-on-node network nodes to IML system elements	109
Table C.10: Mapping of activity-on-node network node start points to IML system elements	110
Table C.11: Mapping of activity-on-node network node end points to IML system elements	110
Table C.12: Mapping of activity-on-node network successor nodes to IML system elements	110
Table C.13: Mapping of activity-on-node network predecessor nodes to IML system elements.....	113
Table C.14: Mapping of the end of an activity-on-node network to IML system elements.....	114
Table C.15: Mapping of the start of a timing diagram to IML system elements	115
Table C.16: Mapping of the timeline to IML system elements.....	116
Table C.17: Mapping of timing diagram resources to IML system elements.....	117
Table C.18: Mapping of timing diagram resource states to IML system elements	118
Table C.19: Mapping of timing diagram signals to IML system elements	120
Table C.20: Mapping of the end of a timing diagram to IML system elements	121
Table C.21: Mapping of timing diagram details to IML system elements	123
Table C.22: Definition of state chart headers	124
Table C.23: Mapping of state chart states to IML system elements.....	125

Table C.24: Mapping of state chart state successors to IML system elements	126
Table C.25: Mapping of state chart predecessor states to IML system elements.....	126
Table C.26: Mapping of state chart actions to IML system elements	127
Table C.27: Mapping of state chart events to IML system elements.....	128
Table C.28: Mapping of state chart signals to IML system elements	128
Table C.29: Mapping of state chart state transitions to IML system elements	134
Table C.30: Mapping of history connectors of state charts to IML system elements	135
Table C.31: Mapping of state chart condition connectors to IML system elements	135
Table G.1: Mapping of the Gantt chart example “activities without predecessor and successor relations”	161
Table G.2: Mapping of the Gantt chart example “activity sequence”	162
Table G.3: Mapping of the Gantt chart example “activity sequence with divergence”	162
Table G.4: Mapping of the Gantt chart example “activity sequence with convergences”.....	163
Table G.5: Mapping of the activity-on-node network example “activity sequence”	164
Table G.6: Mapping of the activity-on-node network example “activity sequence with divergences and convergences”	165
Table G.7: Mapping of the timing diagram example “transition from a state change to the subsequent state”	168
Table G.8: Mapping of the timing diagram example “two external signals fired with delay of three seconds”	169
Table G.9: Mapping of the timing diagram example “signal fired by one resource state and consumed by another”	170
Table G.10: Mapping of the state chart example “simple cyclic state chart”	171
Table G.11: Mapping of the state chart example “states with different predecessors and successors”	172
Table G.12: Mapping of the state chart example “state chart with actions”	173
Table G.13: Mapping of the state chart example “simple cyclic state chart”	174
Table G.14: Mapping of the state chart example “simple hierarchy”	175
Table G.15: Mapping of the state chart example “complex hierarchy and connectors”	177

1 Introduction

The data exchange format defined in IEC 62714 (Automation Markup Language (AML)) is an XML schema based data format and has been developed in order to support the data exchange between engineering tools in a heterogeneous engineering tool landscape. IEC 62714-1 gives an overview about the format.

The goal of AML is to interconnect engineering tools from the existing heterogeneous tool landscape in their different disciplines, e.g. mechanical plant engineering, electrical design, process engineering, process control engineering, HMI development, PLC programming, robot programming, etc.

AML stores engineering information following the object oriented paradigm and allows modelling of physical and logical plant components as data objects encapsulating different aspects. An object may consist of other sub-objects and may itself be part of a larger composition or aggregation. Typical objects in plant automation comprise information on topology, geometry, kinematics, and logic, whereas logic comprises sequencing, behaviour, and control.

AML combines existing industry data formats that are designed for the storage and exchange of different aspects of engineering information. These data formats are used on “as-is” basis within their own specifications and are not branched for AML needs.

The core of AML is the top-level data format CAEX that connects the different data formats. Therefore, AML has an inherent distributed document architecture.

Figure 1 illustrates the basic AML architecture and the distribution of topology, geometry, kinematic, and logic information.

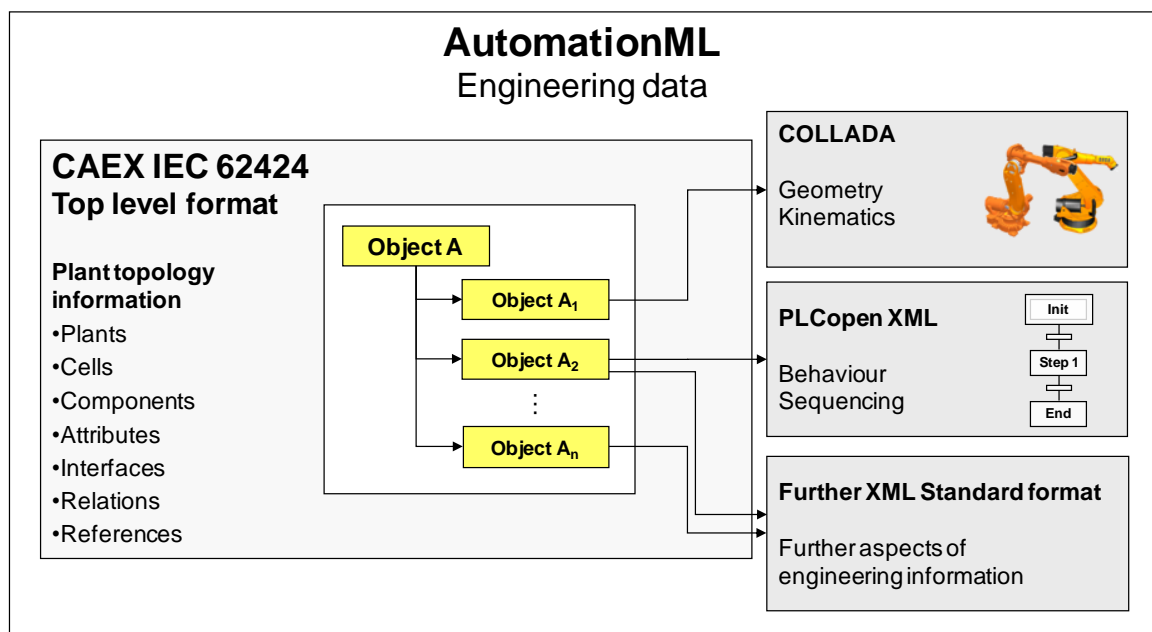


Figure 1: Overview of the engineering data exchange format AML

Due to the different aspects of AML, IEC 62714 consists of different parts focussing on different aspects.

- IEC 62714-1: Architecture and general requirements
This part specifies the general AML architecture, the modelling of engineering data, classes, instances, relations, references, hierarchies, basic AML libraries and extended AML concepts.
- IEC 62714-2: Role class libraries
This part specifies additional AML libraries.

- IEC 62714-3: Geometry and kinematics
This part specifies the modelling of geometry and kinematics information.
- IEC 62714-4: Logic
This part specifies the modelling and referencing of logic information.

Further parts may be added in the future in order to interconnect further data standards to AML.

Clause 3 specifies the normative provisions for modelling, transforming, and storing of logic models.

Clause 4 describes the logic related extensions of the role class library and interface class library.

Clause 5 defines how to store meta information about the source tool directly into the IEC 61131-XML document.

Clause 6 gives a normative description regarding referencing logic information in IEC 61131-XML documents.

Clause 7 gives a normative description of the Intermediate Modelling Layer (IML), which is used to decouple logic models from the target format, in which they are stored.

Clause 8 specifies the normative provisions and describes the additional information necessary to map the IML to IEC 61131-XML documents.

Clause 9 gives a normative description regarding integrating mathematical expressions in logic information.

Clause 10 gives a normative description regarding referencing interlocking information in IEC 61131-XML documents.

Annex A gives an informative overview of this part of the standard.

Annex B gives a normative description of the considered logic models.

Annex C specifies the normative provisions to map the logic models to IML.

Annex D describes the referencing methods for logic information.

Annex E describes the integration of mathematical expressions in logic information.

Annex F describes the referencing methods for interlocking information.

Annex G provides examples for the representation of logic models in IML.

Annex H gives an informative XML representation of the libraries defined in this part of IEC 62714.

Annex I gives a normative XML representation of the schemata defined in this part of IEC 62714.

1.1 Scope

This part of IEC 62714 specifies the integration of logic information as part of an AML model for the exchange between engineering tools in the plant automation area by means of AML.

This part does not define details of the data exchange procedure or implementation requirements for the import/export tools.

1.2 Normative references

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

IEC 62714-1, *Engineering data exchange format for use in industrial automation systems engineering: Automation markup language: Part 1: Architecture and general requirements*

IEC 62424:2008, *Representation of process control engineering - Requests in P&ID diagrams and data exchange between P&ID tools and PCE-CAE tools*

Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004 (available at <<http://www.w3.org/TR/2004/REC-xml-20040204/>>) [viewed on 2016-07-28]

IEC 61131-3, *Programmable controllers - Part 3: Programming languages*

Mathematical Markup Language (MathML) Version 2.0 (Second Edition), W3C Recommendation 21 October 2003 (available at <<https://www.w3.org/TR/MathML2/>> [viewed on 2016-07-28])

IEC 61131-10 (under work in SC 65B), *PLCopen XML Exchange Format according to IEC 61131-3*

ISO/IEC 19505, *Information technology - Object Management Group Unified Modeling Language (OMG UML)*

Unified Modeling Language (UML) Version 2.4.1, OMG (available at <<http://www.omg.org/spec/UML/2.4.1>>) [viewed on 2017-01-27]

2 Terms, definitions and abbreviations

2.1 Terms and definitions

For the purpose of this document, the terms and definitions of IEC 62714-1 apply and in addition the following terms and definitions.

2.1.1 logic information

comprises sequencing information, behaviour information, or interlocking information and its integration into the overall CAEX file.

2.1.2 logic model

represents the information about the controlled and uncontrolled behaviour of the considered system and is a part of the logic information.

2.1.3 IEC 61131-XML

represents an XML based exchange format for IEC61131-3 as defined in IEC 61131-10 (under work in SC 65B).

2.1.4 sequencing information

is logic information that describes the controlled behaviour of the system to external interactions and its integration into the overall CAEX file.

2.1.5 sequencing model

is logic model that describes the uncontrolled behaviour of the system to external interactions.

2.1.6 behaviour information

is logic information that describes the uncontrolled behaviour of the system to external interactions and its integration into the overall CAEX file.

2.1.7 behaviour model

is logic model that describes the uncontrolled behaviour of the system to external interactions.

2.1.8 interlocking information

is logic information that describes the controlled behaviour of the system to avoid unstable state of the system possibly causing harmful effects on humans and environment and its integration into the overall CAEX file.

2.1.9 interlocking model

is logic model that describes the controlled behaviour of the system to avoid unstable state of the system possibly causing harmful effects on humans and environment.

2.1.10 logic objects

tbd

2.1.11 mathematical expression

tbd

2.1.12 state chart

represents a state machine, as specified in ISO/IEC 19505, but only considering those model elements which are necessary to cover the scope of this part of IEC 62714.

Note: The considered model elements are given in B.5.

2.2 Abbreviations

For the purpose of this document, the abbreviations of IEC 62714-1 apply and in addition the abbreviations listed in Table 1.

SFC	Sequential function chart
FBD	Function block diagram
IML	Intermediate modelling layer
POU	Program organisation unit

Table 1: Abbreviations

2.3 Conformity

To claim conformity to the present document with respect to the support of AML, the requirements of clause 3, 4, 5, 6, 7, 8, 9, and 10 shall be fulfilled. In the scope of AML, an IEC 61131-XML document shall conform to the specification of PLCopen[®] XML 2.0 or 2.0.1, or to the specification of IEC 61131-10 (under work in SC 65B).

This standard does not define details of the data exchange procedure or implementation requirements for the import/export tools.

3 Conventions

3.1 General

This clause describes the conventions for the definition of the Intermediate Modelling Layer (IML): see clause 7 - and defines the mapping between the logic models according to Annex B and the IML (see Annex C) as well as the mapping rules between IML and IEC 61131-XML (see 8.3). An informative overview of the mapping concept is provided in Annex A.

3.2 Conventions for the definition of the Intermediate Modelling Layer

Each IML system consists of different IML system elements. The use of each IML element is defined in 7.3.

For each IML system element the following conventions apply:

- IML system elements are characterised by a name, an abbreviation, properties, and relations to other IML system elements according to 7.3.
- Each property and each relation has a name according to 7.3.
- The unambiguous designator of a property or a relation follows the syntax below:
 - 'IML system element abbreviation'. 'IML system element property name'
 - 'IML system element abbreviation'. 'IML system element relation name'

Note: Example: "s.ID", "s.Pre"

3.3 Mapping between logic models and the Intermediate Modelling Layer

According to this standard sequential information may be represented by directed logic models as Gantt charts, activity-on-node networks, timing diagram, and state charts, see Annex C. Regarding the logic models, the following provisions apply:

- A logic model shall be represented by an IML system following the mapping rules described in Annex C.
- Each logic model shall have an initial state.
- Each element of the logic model shall have one or more predecessors and successors (except initial and final states).

3.4 Mapping rules between the Intermediate Modelling Layer and IEC 61131-XML

Due to the equivalence of IML and SFC according to IEC 61131-3 and the capability of IEC 61131-XML to model SFC the following provision apply:

- Each IML system element with its properties and relations shall be mapped to one or more SFC elements according to the mapping rules in 8.3.

Note: An IEC 61131-XML document can contain several IML systems.

Information, which is neither given in IML nor is within the scope of this standard, but is necessary to make the document IEC 61131-XML conform, shall be generated.

4 Part 4 related extensions of AMLibraries

4.1 General

This clause defines extensions of the AML role classes and standard AML interface classes. These classes are part of the AML standard library and specific extension for this part of IEC 62714. All described attributes are part of the standard library and may be removed in the InstanceHierarchy if not needed.

4.2 AutomationMLBaseRoleClassLib

4.3 General

Note: The version of this AML base role class library is 2.2.2 and based on version x.y.z of IEC 62714-1.

Figure 2 present the normative AutomationMLBaseRoleClassLib as object tree. Details to each role class are given in 4.3.1 and 4.3.2. The XML text of AutomationMLBaseRoleClassLib is given in H.1.

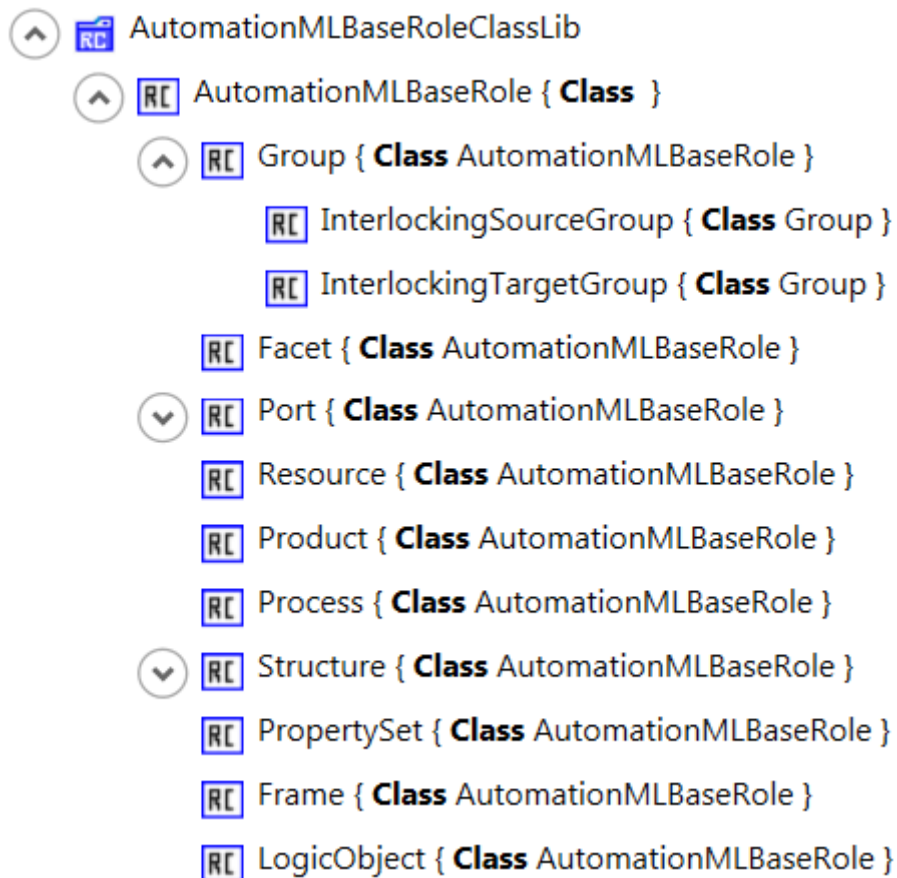


Figure 2: AutomationMLBaseRoleClassLib

4.3.1 RoleClass InterlockingTargetGroup

The role class “InterlockingTargetGroup” shall be used as specified in Table 2.

Class name	InterlockingTargetGroup
Description	The role class “InterlockingTargetGroup” shall be used for objects that group objects belonging to the same interlocking target group. AML interlocking target group objects shall reference this role. Details and examples are specified in 10 and Annex F.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/InterlockingTargetGroup
Attributes	none

Table 2: RoleClass InterlockingTargetGroup

4.3.2 RoleClass InterlockingSourceGroup

The role class “InterlockingSourceGroup” shall be used as specified in Table 3.

Class name	InterlockingSourceGroup
Description	The role class “InterlockingSourceGroup” shall be used for objects that group objects belonging to the same interlocking source group. AML interlocking source group objects shall reference this role. Details and examples are specified in 10 and Annex F.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/InterlockingSourceGroup
Attributes	none

Table 3: RoleClass InterlockingSourceGroup

4.3.3 RoleClass LogicObject

The role class “LogicObject” shall be used as specified in Table 4.

Class name	LogicObject
Description	The role class “LogicObject” shall denote a logic object. Examples are given in Annex F.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/LogicObject
Attributes	none

Table 4: RoleClass LogicObject

Note: These objects can be enriched and further referenced in CAEX.

4.4 AutomationMLInterfaceClassLib

4.4.1 General

Note: The version of this AML interface class library is 2.2.2.

Figure 3 presents the normative AutomationMLInterfaceClassLib as object tree. Details to each interface class are given in 4.4.2 to 4.4.10. The XML text of AutomationMLInterfaceClassLib is given in H.2.

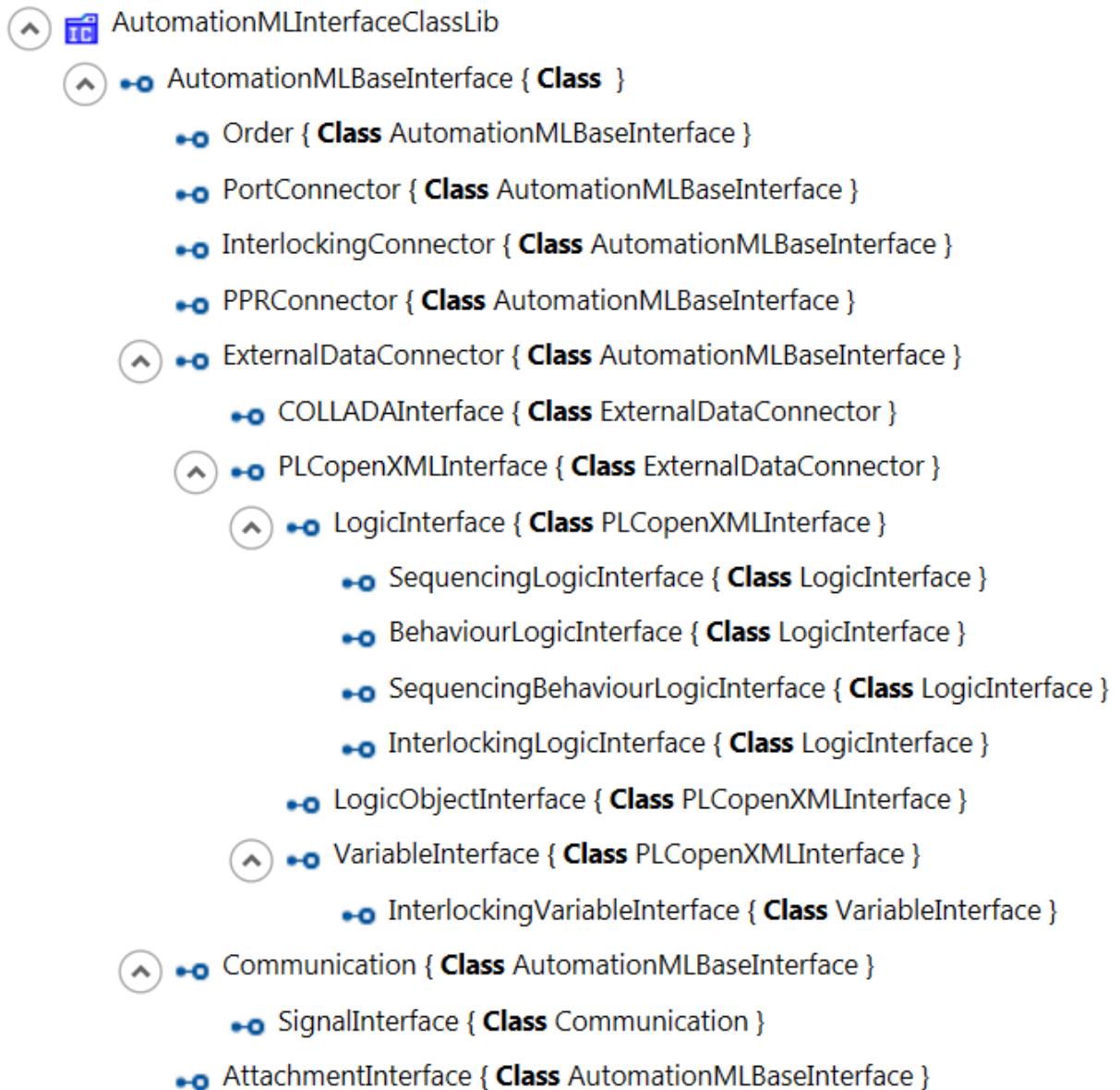


Figure 3: AutomationMLInterfaceClassLib

4.4.2 InterfaceClass LogicInterface

The interface class “LogicInterface” shall be used as specified in Table 5.

Class name	LogicInterface
Description	The interface class “LogicInterface” shall be used in order to reference logic information stored in a POU within an external IEC 61131-XML document. AML logic interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface
Attributes	none

Table 5: InterfaceClass LogicInterface

4.4.3 InterfaceClass SequencingLogicInterface

The interface class “SequencingLogicInterface” shall be used as specified in Table 6.

Class name	SequencingLogicInterface
Description	The interface class “SequencingLogicInterface” shall be used in order to reference sequencing information stored in a POU within an external IEC 61131-XML document. AML sequencing interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface/SequencingLogicInterface
Attributes	none

Table 6: InterfaceClass SequencingLogicInterface

4.4.4 InterfaceClass BehaviourLogicInterface

The interface class “BehaviourLogicInterface” shall be used as specified in Table 7.

Class name	BehaviourLogicInterface
Description	The interface class “BehaviourLogicInterface” shall be used in order to reference behaviour information stored in a POU within an external IEC 61131-XML document. AML behaviour interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface/BehaviourLogicInterface
Attributes	none

Table 7: InterfaceClass BehaviourLogicInterface

4.4.5 InterfaceClass SequencingBehaviourLogicInterface

The interface class “SequencingBehaviourLogicInterface” shall be used as specified in Table 8.

Class name	SequencingBehaviourLogicInterface
Description	The interface class “SequencingBehaviourLogicInterface” shall be used in order to reference sequencing and behaviour information stored in a POU within an external IEC 61131-XML document. AML sequencing behaviour interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface/SequencingBehaviourLogicInterface
Attributes	none

Table 8: InterfaceClass SequencingBehaviourLogicInterface

4.4.6 InterfaceClass InterlockingLogicInterface

The interface class “InterlockingLogicInterface” shall be used as specified in Table 9.

Class name	InterlockingLogicInterface
Description	The interface class “InterlockingLogicInterface” shall be used in order to reference interlocking information stored in a POU within an external IEC 61131-XML document. AML interlocking interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicInterface/InterlockingLogicInterface
Attributes	none

Table 9: InterfaceClass InterlockingLogicInterface

4.4.7 InterfaceClass LogicObjectInterface

The interface class “LogicObjectInterface” shall be used as specified in Table 10.

Class name	LogicObjectInterface
Description	The interface class “LogicObjectInterface” shall be used to reference objects within an IEC 61131-XML document. AML logic object interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/LogicObjectInterface
Attributes	none

Table 10: InterfaceClass LogicObjectInterface

Note: Within an IEC 61131-XML document logic objects can be steps, transitions, actions etc.

4.4.8 InterfaceClass VariableInterface

The interface class “VariableInterface” shall be used as specified in Table 11.

Class name	VariableInterface
Description	The interface class “VariableInterface” shall be used in order to reference a variable in an external IEC 61131-XML document on a signal level for industrial communication. AML variable interface objects shall reference this interface class. Details and examples are specified in 6 and Annex D.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/ PLCOpenXMLInterface/VariableInterface
Attributes	none

Table 11: InterfaceClass VariableInterface

4.4.9 InterfaceClass InterlockingConnector

The interface class “InterlockingConnector” shall be used as specified in Table 12.

Class name	InterlockingConnector
Description	The interface class “InterlockingConnector” shall be used in order to model relations between an interlocking source group and an interlocking target group. AML interlocking interface objects shall reference this interface class. Details and examples are specified in 10 and Annex F.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/InterlockingConnector
Attributes	none

Table 12: InterfaceClass InterlockingConnector

4.4.10 InterfaceClass InterlockingVariableInterface

The interface class “InterlockingVariableInterface” shall be used as specified in Table 13.

Class name	InterlockingVariableInterface	
Description	The interface class “InterlockingVariableInterface” shall be used in order to reference a variable which represents the results of a function describing the interlocking condition stored in POU or a POU network. AML interlocking variable interface objects shall reference this interface class. Details and examples are specified in 10 and Annex F.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/PLCopenXMLInterface/VariableInterface	
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/PLCopenXMLInterface/VariableInterface/InterlockingVariableInterface	
Attributes	SafeConditionEquals (type="xs:boolean")	<p>The attribute “SafeConditionEquals” shall be used in order to specify the result of the function describing the interlocking condition. The allowed values shall be “true” or “false” and shall indicate which value of a unique Boolean variable represents the safe state of an interlocking source group.</p> <p>The use of the attribute shall be optional. The default value shall be “true”.</p>

Table 13: InterfaceClass InterlockingVariableInterface

5 Meta information about the IEC 61131-XML source tool

In order to simplify the data exchange between a source tool and a destination tool, it is useful to store information about the source tool directly into the IEC 61131-XML document. Hence, the following provisions apply:

Note: In the engineering process it is useful to transfer the information of the project the data belongs to and the time of the export of the data. This supports the implementation process of interfaces.

- Each IEC 61131-XML document shall provide information about the tool which has written the IEC 61131-XML document.
- In a data exchange tool chain, the last participating tools shall store this information in the IEC 61131-XML document.
- This information shall be stored within attributes of the IEC 61131-XML element “fileHeader”.
- The meta information shall provide information about:
 - name of the exporting software, the writer tool;
 - vendor of the writer tool;
 - URL of the writer tool;
 - product version of the writer tool;
 - product release number of the writer tool;
 - creation date and time of the IEC 61131-XML document.
- The required information shall be stored by means of the attributes shown in Table 14.

XMLtagname	Type	Level	Example
companyName	xs:string	Mandatory	“ToolX Vendor”
companyURL	xs:string	Mandatory	“http://www.ToolX-Vendor.org”
productName	xs:anyURI	Mandatory	“ToolX”
productVersion	xs:string	Mandatory	“0.2”
productRelease	xs:string	Mandatory	“123 prealpha”
creationDateTime	xs:dateTime	Mandatory	“2011-05-25T09:30:47”

Table 14: Meta information about the IEC 61131-XML source tool

6 Referencing IEC 61131-XML documents

6.1 General

According to the distributed document structure of AML it is necessary to put the different information within the different documents in relation to each other. This clause focuses on the referencing of AML objects (within a CAEX document) with logic information stored in IEC 61131-XML documents. An informative overview of the referencing methods is provided in Annex D.

6.2 Referencing logic information

Regarding referencing logic information, the following provisions apply:

- A reference from an AML object to logic information within an IEC 61131-XML document shall be modelled by means of a CAEX ExternalInterface using the AML InterfaceClasses “LogicInterface”, “SequencingLogicInterface”, “BehaviourLogicInterface”, “SequencingBehaviourLogicInterface”, “InterlockingLogicInterface”, “LogicObjectInterface”, “VariableInterface”, “InterlockingVariableInterface”, or derivations of them, specified in 4.4.
 - Logic information shall be referenced by its URI within the attribute “refURI” of these CAEX ExternalInterfaces.
 - The value of the attribute “refURI” shall point at the logic information within the IEC 61131-XML document and shall follow the syntax: URI (which specifies the document) followed by the separating character “#” followed by the IEC 61131-XML attribute “globalID” (which specifies the logic information).

Note: It is recommended to use a universal unique identifier (UUID) for the globalID to enable a file crossing identification.

Note: Structure of the “refURI” attribute value resembles <URI>#<globalID>, e.g. file:///behaviour.xml# UUID_74e94dc8-b5a2-490c-bf0f-f0fdaa4515ef.

Note: In this part of IEC 62714, UUIDs are presented in a short form such as “UUID1”, “UUID100” etc. This serves the readability and acts as a real UUID.

Note: It is not recommended to just reference the IEC 61131-XML document itself, since the document may contain several logic models containing logic information.

- An AML object shall have one or more CAEX ExternalInterfaces of the AML InterfaceClass “LogicInterface”, “SequencingLogicInterface”, “BehaviourLogicInterface”, “SequencingBehaviourLogicInterface”, “InterlockingLogicInterface”, or a derivation of them referencing one or more logic models which contain logic information. In case that an AML object has the logic model already referenced within its parent objects no such an CAEX ExternalInterface shall not be necessary.

Note 1: If an AML object has no logic information assigned to it, no CAEX ExternalInterface of the AML InterfaceClass “LogicInterface” is permitted.

Note 2: Logic information can be stored in one POU or distributed throughout several POUs called POU network.

Note 3: Logic information can be stored within one IEC 61131-XML document or distributed throughout several IEC 61131-XML documents which can be connected by means of CAEX.

- Logic models containing logic information shall be distinguishable, i.e. an AML object shall not have more than one CAEX ExternalInterface of the same AML InterfaceClass “LogicInterface”, “SequencingLogicInterface”, “BehaviourLogicInterface”, “SequencingBehaviourLogicInterface”, or a derivation of them.
- An AML object shall have zero or more CAEX ExternalInterfaces of the AML InterfaceClass “InterlockingLogicInterface” or a derivation of it (see clause 10 and Annex F).
- The following AML InterfaceClasses shall be used for AML objects that have already referenced logic information within itself or within its parent objects. These shall be modelled as direct or indirect children of AML objects with the AML InterfaceClass

“LogicInterface”, “SequencingLogicInterface”, “BehaviourLogicInterface”, “SequencingBehaviourLogicInterface”, “InterlockingLogicInterface”, or a derivation of them.

- An AML object referencing a logic object shall have one or more ExternalInterfaces of the AML InterfaceClass “LogicObjectInterface” or a derivation of it.

Note: Within a POU of an IEC 61131-XML document, logic objects can be steps, transitions, actions etc.

- An AML object referencing a variable shall have one or more ExternalInterfaces of the AML InterfaceClass “VariableInterface” or a derivation of it.

Note: The variables can, then, be assigned, e.g. to signals of other AML objects or to variables of other POUs, by using InternalLinks.

- An AML object referencing an interlocking variable shall have one ExternalInterface of the AML InterfaceClass “InterlockingVariableInterface” or a derivation of it.

7 Intermediate Modelling Layer

7.1 General

This clause defines the Intermediate Modelling Layer (IML), as a decoupling layer between the mapping of logic models and SFCs according to IEC 61131-3 as the IEC 61131-XML representation, and describes the IML system elements.

7.2 IML system overview

An IML system consists of elements with their properties and relations as depicted in Figure 4.

Note: For further information see A.3.

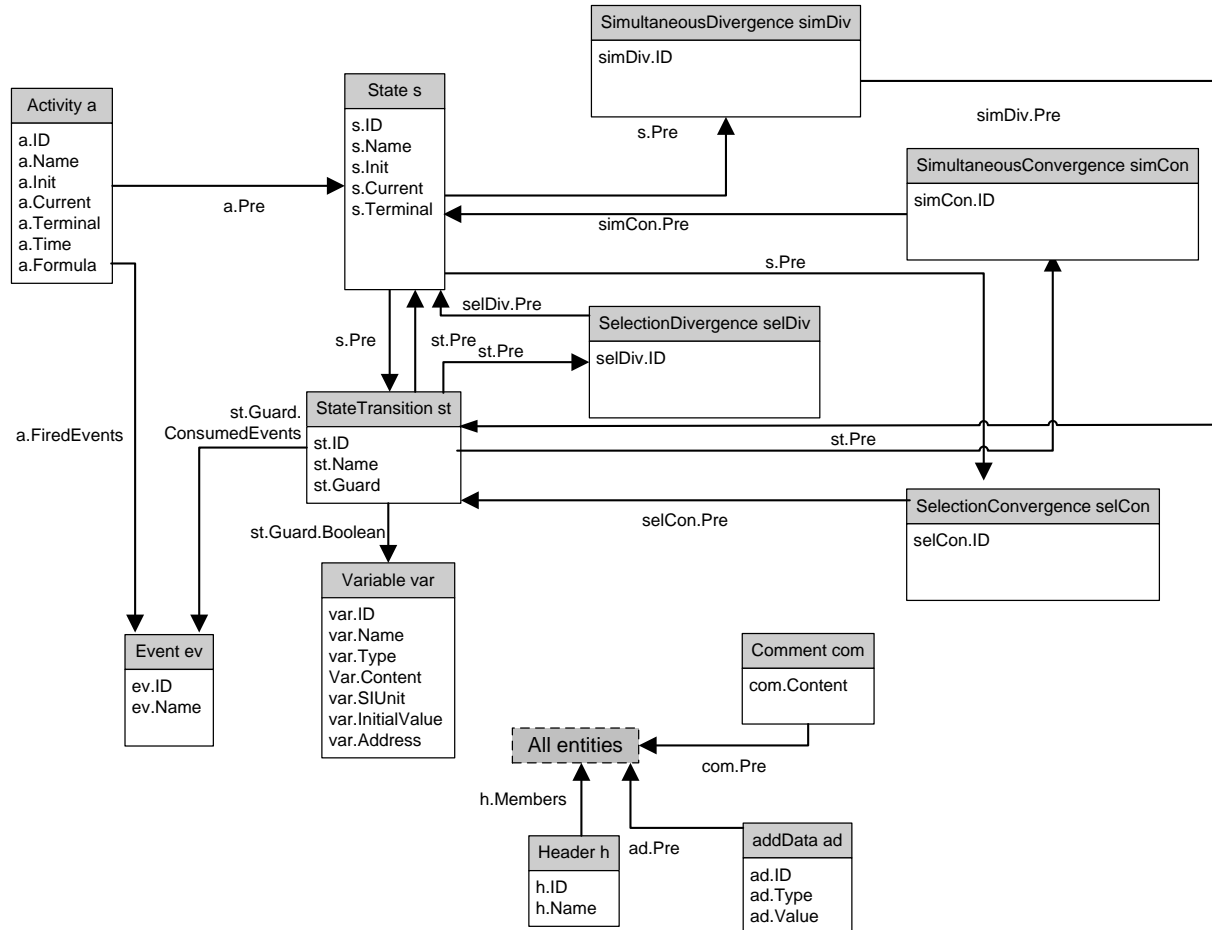


Figure 4: IML system entities and its relations

7.3 Definition of IML system elements

An IML system is a set of the following IML system elements with their corresponding properties and relations.

7.3.1 “Header”

A header “h” serves as a description of an IML system.

A header “h” shall have the following properties:

- “h.ID” shall represent the unique identification of the IML system.
- “h.Name” shall represent the unique name of the IML system.

A header “h” shall have the following relation:

- “h.Members” shall represent the set of all IML system elements of the IML system.

7.3.2 “State”

A state “s” describes a stable situation within an IML system where a dedicated set of parameters is valid. These parameters can be given by running actions, events, and values of process variables. States are reached and left via state transitions.

A state “s” shall have the following properties:

- “s.ID” shall represent the unique identification of the state “s” within an IML system.
- “s.Name” shall represent the unique name of the state “s” within an IML system.

Additionally the following provisions apply:

- Initial states shall have the property “s.Init” with a value “true”.

Note: The value “false” or the absence of the property “s.Init” indicates that the present state is not an initial state.

- Currently activated states shall have the property “s.Current” with a value “true”.

Note: The value “false” or the absence of the property “s.Current” indicates that the present state is not a currently activated state.

- Terminal states shall have the property “s.Terminal” with a value “true”.

Note: The value “false” or the absence of the property “s.Terminal” indicates that the present state is not a terminal state.

Except of the initial state all other states “s” shall have the following relation:

- “s.Pre” shall represent the set of predecessors of the state “s” covering the references to selection convergences, simultaneous divergences or state transitions.

7.3.3 “State transition”

A state transition “st” describes the transition from one state to one or more subsequent states by interpreting state transition conditions.

A state transition “st” shall have the following properties:

- “st.ID” shall represent the unique identification of the state transition “st” within an IML system.
- “st.Name” shall represent the unique name of the state transition “st” within an IML system.
- “st.Guard” shall represent one or more of the following execution conditions of the state transition “st”. An overview of all possible combinations is given in Table 15.
 - Empty guard: This represents a permanently fulfilled state transition condition.
 - One of the following:
 - “st.Guard.Boolean” shall name the unique Boolean variable giving the firing condition.
 - “st.Guard.Value” shall name a unique variable and a closed interval of valid values for the variable as necessary firing condition.

Note: The condition is “true” only if the value of the variable is within or on the borders of the interval.

- “st.Guard.Formula” shall name a Structured Text following IEC 61131-3 formula providing a Boolean value to encode the necessary firing condition.
- “st.Guard.ConsumedEvent” shall name all events that together are necessary to fire conditions.

Possible combinations	st.Guard	st.Guard.ConsumedEvent
-----------------------	----------	------------------------

of execution conditions	Empty	Boolean	Value	Formula	
Combination 1	x				
Combination 2		x			
Combination 3		x			x
Combination 4			x		
Combination 5			x		x
Combination 6				x	
Combination 7				x	x
Combination 8					x

Table 15: Overview of all possible combinations of execution conditions

Each state transition “st” shall have the following relation:

- “st.Pre” shall represent the set of predecessors of the state transition “st” covering the references to predecessor states, simultaneous convergences or selection divergences.

7.3.4 “Activity”

An activity “a” describes one or more operations related to a certain state. It is characterized by required and to be changed variables, local time properties and by possible events fired after its execution.

Note: Activities cannot be attached to state transitions.

An activity “a” shall have the following properties:

- “a.ID” shall represent a unique identification of the activity “a” within an IML system.
- “a.Name” shall represent the unique name of the activity “a” within an IML system.
- “a.Formula” shall name a Structured Text formula.

Additionally the following provisions apply:

- Initial activities shall have the property “a.Init” with a value “true”.

Note: The value “false” or the absence of the property “a.Init” indicates that the present activity is not an initial activity.

- Currently running activities shall have the property “a.Current” with a value “true”.

Note: The value “false” or the absence of the property “a.Current” indicates that the present activity is not a currently running activity.

- Terminal activities shall have the property “a.Terminal” with a value “true”.

Note: The value “false” or the absence of the property “a.Terminal” indicates that the present activity is not a terminal activity.

- For a mapping of time information the following property shall be used:

- “a.Time” shall represent the time condition of the activity “a” (using the SI unit second (s)) and shall consist of a subset of the following properties:

Note: The subset results from the transformation rules of the mapping between logic models and the Intermediate Modelling Layer.

- “a.Time.Duration” shall be a non-negative real value representing the duration of the activity “a”.
- “a.Time.Start” shall be a range [Earliest, Latest] with non-negative real values as boundaries for the start point of the activity “a”.
- “a.Time.End” shall be a range [Earliest, Latest] with non-negative real values as boundaries for the end point of the activity “a”.

- “a.Time.Delay” shall be a non-negative real value representing the delay of starting the activity “a”.

An activity “a” shall have the following relation:

- “a.Pre” shall represent the set of states to which the activity “a” relates.

Additionally the following provision applies:

- “a.FiredEvent” shall name all events fired at the end of the activity “a”.

7.3.5 “Selection divergence”

A selection divergence “selDiv” is a logical association between one predecessor state and two or more successor state transitions. The successor state transitions can be regarded as having an XOR relation.

A selection divergence “selDiv” shall have the following property:

- “selDiv.ID” shall represent the unique identifier of the selection divergence “selDiv” within an IML system.

A selection divergence “selDiv” shall have the following relation:

- “selDiv.Pre” shall represent the predecessor state of the selection divergence “selDiv”.

7.3.6 “Simultaneous divergence”

A simultaneous divergence “simDiv” is a logical association between one predecessor state transition and two or more successor states. The successor state can be regarded as having an AND relation.

A simultaneous divergence “simDiv” shall have the following property:

- “simDiv.ID” shall represent the unique identifier of the simultaneous divergence “simDiv” within an IML system.

A simultaneous divergence “simDiv” shall have the following relation:

- “simDiv.Pre” shall represent the predecessor state transition of the simultaneous divergence “simDiv”.

7.3.7 “Selection convergence”

A selection convergence “selCon” is a logical association between two or more predecessor state transitions and one successor state.

A selection convergence “selCon” shall have the following property:

- “selCon.ID” shall represent the unique identification of the selection convergence “selCon” within an IML system.

A selection convergence “selCon” shall have the following relation:

- “selCon.Pre” shall represent the set of predecessors of the selection convergence “selCon” covering the references to predecessor state transitions.

7.3.8 “Simultaneous convergence”

A simultaneous convergence “simCon” is a logical association between two or more predecessor states and one successor state transition.

A simultaneous convergence “simCon” shall have the following property:

- “simCon.ID” shall represent the unique identification of the simultaneous convergence “simCon” within an IML system.

A simultaneous convergence “simCon” shall have the following relation:

- “simCon.Pre” shall represent the set of predecessors of the simultaneous convergence “simCon” covering the references to predecessor states.

7.3.9 “Event”

An event “ev” is fired by an activity to which it is associated. Events are mainly used as triggers for state transitions.

An event “ev” shall have the following properties:

- “ev.ID” shall represent the unique identification of the event “ev” within an IML system.
- “ev.Name” shall represent the unique name of the event “ev” within an IML system.

7.3.10 “Variable”

A variable “var” is a modelling entity that characterizes states and activities. Its value can be changed by activities; it can be used as trigger conditions for state transitions or as system input and output.

A variable “var” shall have the following properties:

- “var.ID” shall represent the unique identifier of the variable “var” within an IML system.
- “var.Name” shall represent the unique name of the variable “var” within an IML system.
- “var.Type” shall be the data type of the variable “var” following the data types defined by IEC 61131-3.
- “var.Content” shall describe the use of the variable “var” as local, input, output or inout variable. Default value shall be local.

Additionally the following provisions apply:

- “var.SIUnit” shall be the measurement type of the variable “var” following the SI system of measuring units.
- “var.InitialValue” shall be the initial value of the variable “var”.
- “var.Address” shall be the physical address of the variable “var” as defined by IEC 61131-3.

7.3.11 “Comment”

A Comment “com” is used for descriptive information.

A comment “com” shall have the following property:

- “com.Content” shall be the content string of the comment “com”.

A comment “com” shall have the following relation:

- “com.Pre” shall reference the IML system element to which the comment “com” is associated.

7.3.12 “Additional data”

Additional data “addData” allows to store and exchange additional information.

Note: Examples are complex timing information, runtime information or descriptive data.

An addData “ad” shall have the following properties:

- “ad.Type” shall contain one value of the list of types for additional data as described in 8.2.

Note: The syntax of additional data is specified in a separate XML schema (see Annex I).

- “ad.Value” shall be a string representing the XML content of the addData “ad”.
- “ad.ID” shall represent the unique identification of the addData “ad” within an IML system.

An addData “ad” shall have the following relation:

- “ad.Pre” shall represent the IML system element to which the addData “ad” relates.

Note: This predecessor can be every IML system element.

8 Mapping of IML to IEC 61131-XML

8.1 General

This clause defines rules for the transformation of IML systems to IEC 61131-XML documents. AML uses the IEC 61131-XML representation of the IEC 61131-3 programming language "Sequential Function Charts" (SFC) to store all relevant logic information of the IML. But first, a schema defined is specified by AML for storing additional logic information within the IEC 61131-XML element "addData".

8.2 AML addData schema for additional data within logic information

AML exploits the additional data mechanism of IEC 61131-XML to integrate logic model specific information required by AML but not covered by the IEC 61131-XML. Hence, AML specifies an XML schema covering those additional elements and attributes. It is called AML addData schema. Annex I.1 provides an XML representation of the schema.

The declaration of the AML addData schema in IEC 61131-XML documents, enabling the usage of the AML addData schema, is described in 8.2.1. Then, the additional data can be stored within IEC 61131-XML elements "addData" according to the AML addData schema (see 8.2.2 - 8.2.21).

Note: Examples for storing additional data within IEC 61131-XML elements "addData" is given in 8.3.

8.2.1 Declaration and usage of AML addData schema in IEC 61131-XML

To use the AML addData schema in an IEC 61131-XML document, the schema shall be listed in the IEC 61131-XML element "addDataInfo" to enable the unambiguous identification of the corresponding IEC 61131-XML element "addData" contents (see Figure 5).

```
<contentHeader name="logic model">
  ...
  <addDataInfo>
    <info name="http://www.automationml.org/IEC62714-4Ed1/AML_addData.xsd"
vendor="AutomationML">
      <description>
        <xhtml:p>Schema to integrate logic model specific information required by
AML.</xhtml:p>
      </description>
    </info>
  </addDataInfo>
</contentHeader>
```

Figure 5: XML text of declaration of AML schema for storing additional logic information within IEC 61131-XML documents

8.2.2 XML schema description of AMLspecific data

Table 16 specifies the use of the AML schema element “AML”. Only one “AML” element shall exist; All information describes in this subclause shall be grouped within one “AML” element.

Diagram	<p>AML</p> <p>It comprises all of the additional, AML specific logic information.</p> <p>aml:Time</p> <p>aml:ChartType</p> <p>aml:ResourceStateChangeDefin...</p> <p>0..∞</p> <p>aml:InterruptibleAction</p> <p>aml:StateChartSubCharts</p> <p>0..∞</p> <p>aml:StateChartStateType</p> <p>aml:StateChartActionType</p> <p>aml:ImpulseDiagramResourceG...</p> <p>aml:ImpulseDiagramPLCVariable</p> <p>aml:StateStatus</p> <p>aml:ActionStatus</p> <p>aml:Unit</p>
Properties	<p>content: complex</p> <p>Followed by a “sequence”.</p>
Children	<p>aml:Time,aml:ChartType,aml:ResourceStateChangeDefinition,aml:InterruptibleAction,aml:StateChartSubCharts,aml:StateChartStateType,aml:StateChartActionType,aml:ImpluseChartResourceGroup,aml:ImpulseDiagramPLCVariable,aml:StateStatus,aml:ActionStatus, aml:Unit</p>

Table 16. AML schema element “AML”

8.2.3 XML schema description of timing data

Table 17 specifies the use of the AML schema element “Time”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex Followed by a “sequence”.
Children	aml:Duration,aml:EarliestStart,aml:LatestStart,aml:EarliestEnd,aml:LatestEnd,aml:Delay
Attributes	Attribute: ID (from type “xs:string ” with use “required”)
Type	extension of aml:addDataBaseObject

Table 17. AML schema element “Time”

8.2.4 XML schema description of durations regarding timing data

Table 18 specifies the use of the AML schema element “Duration”.

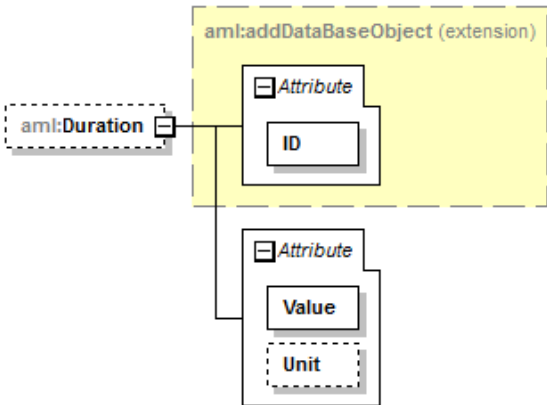
Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Value (from type “xs:decimal ” with use “required”) Attribute: Unit (from type “xs:string ” with use “optional”)
Type	extension of aml:addDataBaseObject

Table 18: AML schema element “Duration” (within the AML schema element “Time”)

8.2.5 XML schema description of earliest starts regarding timing data

Table 19 specifies the use of the AML schema element “EarliestStart”.

Diagram	
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: 1</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: Value (from type “xs:decimal ” with use “required”)</p> <p>Attribute: Unit (from type “xs:string ” with use “optional”)</p>
Type	extension of aml:addDataBaseObject

Table 19: AML schema element “EarliestStart” (within the AML schema element “Time”)

8.2.6 XML schema description of latest starts regarding timing data

Table 20 specifies the use of the AML schema element “LatestStart”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Value (from type “xs:decimal ” with use “required”) Attribute: Unit (from type “xs:string ” with use “optional”)
Type	extension of aml:addDataBaseObject

Table 20: AML schema element “LatestStart” (within the AML schema element “Time”)

8.2.7 XML schema description of earliest ends regarding timing data

Table 21 specifies the use of the AML schema element “EarliestEnd”.

Diagram	
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: 1</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: Value (from type “xs:decimal ” with use “required”)</p> <p>Attribute: Unit (from type “xs:string ” with use “optional”)</p>
Type	extension of aml:addDataBaseObject

Table 21: AML schema element “EarliestEnd” (within the AML schema element “Time”)

8.2.8 XML schema description of latest ends regarding timing data

Table 22 specifies the use of the AML schema element “LatestEnd”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Value (from type “xs:decimal ” with use “required”) Attribute: Unit (from type “xs:string ” with use “optional”)
Type	extension of aml:addDataBaseObject

Table 22: AML schema element “LatestEnd” (within the AML schema element “Time”)

8.2.9 XML schema description of delays regarding timing data

Table 23 specifies the use of the AML schema element “Delay”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: simple
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Value (from type “xs:decimal ” with use “required”) Attribute: Unit (from type “xs:string ” with use “optional”)
Type	extension of aml:addDataBaseObject

Table 23: AML schema element “Delay” (within the AML schema element “Time”)

8.2.10 XML schema description of chart types

Table 24 specifies the use of the AML schema element “ChartType”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: ChartType (from type “xs:string ”, derived by “restriction”, with use “required”) Enumeration: StateChart Enumeration: TimingDiagram Enumeration: GanttChart Enumeration: ActivityOnNodeNetwork Enumeration: SequentialFunctionChart Enumeration: FunctionBlockDiagram
Type	extension of aml:addDataBaseObject

Table 24: AML schema element “ChartType”

8.2.11 XML schema description of change definitions of resource states

Table 25 specifies the use of the AML schema element “ResourceStateChangeDefinition”.

Diagram	<p>The diagram illustrates the relationship between <code>aml:ResourceStateChangeDefinition</code> and <code>aml:addDataBaseObject (extension)</code>. <code>aml:ResourceStateChangeDefinition</code> is shown as a dashed box extending from the base class <code>aml:addDataBaseObject (extension)</code>. The base class has three attributes: <code>ID</code>, <code>DefinitionName</code>, and <code>Duration</code>. The extension class has one attribute: <code>ID</code>. The cardinality is <code>0..∞</code>.</p>
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: unbounded</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: DefinitionName (from type “xs:string ” with use “required”)</p> <p>Attribute: Duration (from type “xs:decimal ” with use “required”)</p>
Type	extension of <code>aml:addDataBaseObject</code>

Table 25: AML schema element “ResourceStateChangeDefinition”

8.2.12 XML schema description of interruptible actions

Table 26 specifies the use of the AML schema element “InterruptibleAction”.

Diagram	<p>The diagram illustrates the relationship between <code>aml:InterruptibleAction</code> and <code>aml:addDataBaseObject (extension)</code>. <code>aml:InterruptibleAction</code> is shown as a dashed box extending from the base class <code>aml:addDataBaseObject (extension)</code>. The base class has two attributes: <code>ID</code> and <code>Value</code>. The extension class has one attribute: <code>ID</code>. The cardinality is <code>1</code>.</p>
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: 1</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: Value (from type “xs:boolean ” with use “required”)</p>
Type	extension of <code>aml:addDataBaseObject</code>

Table 26: AML schema element “InterruptibleAction”

8.2.13 XML schema description of sub charts in state charts

Table 27 specifies the use of the AML schema element “StateChartSubCharts”.

Diagram	
Properties	<p>isRef 0</p> <p>minOcc 0</p> <p>maxOcc unbounded</p> <p>content complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: POURef (from type “xs:string ” with use “required”)</p>
Type	extension of aml:addDataBaseObject

Table 27: AML schema element “StateChartSubCharts”

8.2.14 XML schema description of state types of state charts

Table 28 specifies the use of the AML schema element “StateChartStateType”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: StateChartStateType (from type “xs:string ”, derived by “restriction”, with use “required”) Enumeration: HigherLevelState Enumeration: HistoryConnector Enumeration: ConditionConnector Enumeration: StateForActivity
Type	extension of aml:addDataBaseObject

Table 28: AML schema element “StateChartStateType”

8.2.15 XML schema description of actions types of state charts

Table 29 specifies the use of the AML schema element “StateChartActionType”.

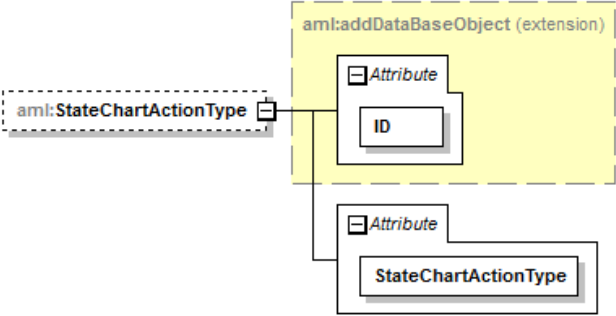
Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: StateChartActionType (from type “xs:string ”, derived by “restriction”, with use “required”) Enumeration: DoAction Enumeration: ExitAction Enumeration: EntryAction
Type	extension of aml:addDataBaseObject

Table 29: AML schema element “StateChartActionType”

8.2.16 XML schema description of resource groups of timing diagrams

Table 30 specifies the use of the AML schema element “TimingDiagramResourceGroup”.

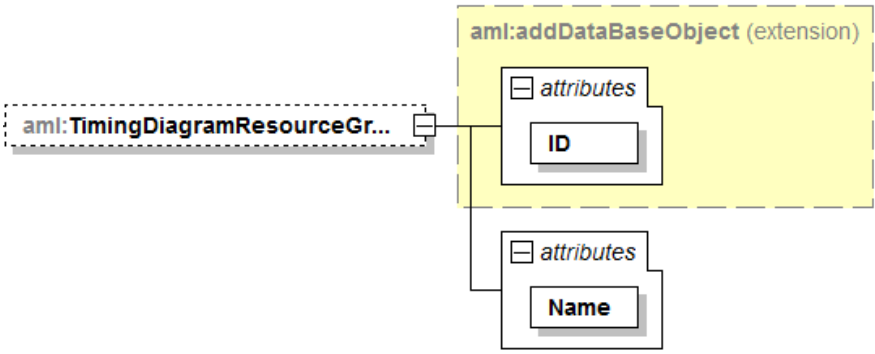
Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Name (from type “xs:string ” with use “required”)
Type	extension of aml:addDataBaseObject

Table 30: AML schema element “TimingDiagramResourceGroup”

8.2.17 XML schema description of PLC variables of timing diagrams

Table 31 specifies the use of the AML schema element “TimingDiagramPLCVariable”.

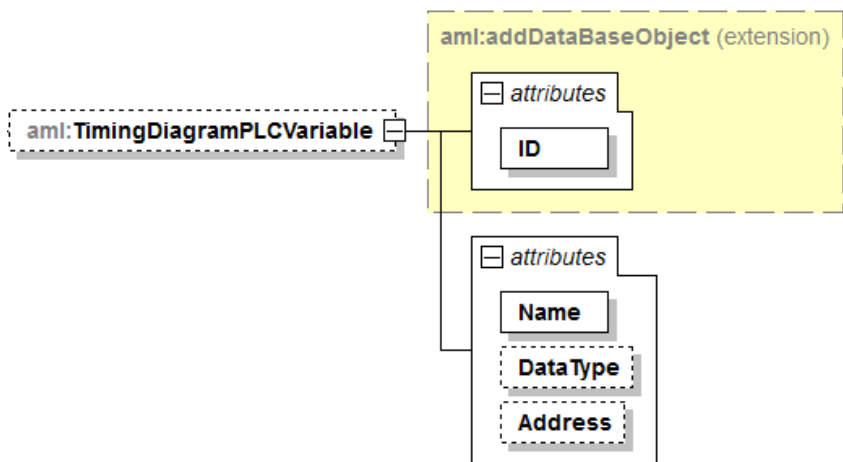
Diagram	 <p>The diagram illustrates the relationship between the <code>aml:TimingDiagramPLCVariable</code> element and the <code>aml:addDataBaseObject (extension)</code> element. The <code>aml:TimingDiagramPLCVariable</code> is connected to the <code>ID</code> attribute of the <code>aml:addDataBaseObject (extension)</code>. The <code>aml:addDataBaseObject (extension)</code> has attributes <code>ID</code>, <code>Name</code>, <code>DataType</code>, and <code>Address</code>.</p>
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: 1</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: Name (from type “xs:string ” with use “required”)</p> <p>Attribute: DataType (from type “xs:string ” with use “optional”)</p> <p>Attribute: Address (from type “xs:string ” with use “optional”)</p>
Type	extension of <code>aml:addDataBaseObject</code>

Table 31: AML schema element “TimingDiagramPLCVariable”

8.2.18 XML schema description of state status

Table 32 specifies the use of the AML schema element “StateStatus”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Current (from type “xs:boolean” with use “required”) Attribute: Terminal (from type “xs:boolean” with use “required”)
Type	extension of aml:addDataBaseObject

Table 32: AML schema element “StateStatus”

8.2.19 XML schema description of action status

Table 33 specifies the use of the AML schema element “ActionStatus”.

Diagram	<p>The diagram illustrates the relationship between the <code>aml:ActionStatus</code> element and the <code>aml:addDataBaseObject</code> extension. <code>aml:ActionStatus</code> is shown as a dashed box containing an attribute <code>ID</code>. <code>aml:addDataBaseObject</code> (extension) is shown as a solid box containing an attribute <code>ID</code> and three other attributes: <code>Initial</code>, <code>Current</code>, and <code>Terminal</code>.</p>
Properties	<p>isRef: 0</p> <p>minOcc: 0</p> <p>maxOcc: 1</p> <p>content: complex</p>
Attributes	<p>Attribute: ID (from type “xs:string ” with use “required”)</p> <p>Attribute: Initial (from type “xs:boolean” with use “required”)</p> <p>Attribute: Current (from type “xs:boolean” with use “required”)</p> <p>Attribute: Terminal (from type “xs:boolean” with use “required”)</p>
Type	extension of <code>aml:addDataBaseObject</code>

Table 33: AML schema element “ActionStatus”

8.2.20 XML schema description of units of measurement

Table 34 specifies the use of the AML schema element “Unit”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”) Attribute: Name (from type “xs:boolean” with use “required”)
Type	extension of aml:addDataBaseObject

Table 34: AML schema element “Unit”

8.2.21 XML schema description of addDataBaseObject

Table 35 specifies the use of the AML schema element “addDataBaseObject”.

Diagram	
Properties	isRef: 0 minOcc: 0 maxOcc: 1 content: complex
Attributes	Attribute: ID (from type “xs:string ” with use “required”)
Used by	aml:Time, aml:Time/Duration, aml:Time/EarliestStart, aml:Time/LatestStart, aml:Time/EarliestEnd, aml:Time/LatestEnd, aml:Time/Delay, aml:ChartType, aml:ResourceStateChangeDefinition, aml:InterruptibleAction, aml:StateChartSubCharts, aml:StateChartStateType, aml:StateChartActionType, aml:ImpulseDiagramResourceGroup, aml:ImpulseDiagramPLCVariable, aml:StateStatus, aml:ActionStatus, aml:Unit

Table 35: AML schema element “addDataBaseObject”

8.3 Mapping of IML to IEC 61131-XML SFC

Note: All tags and attributes are omitted which are mandatory for a valid IEC 61131-XML document but are not relevant for the certain transformation of IML system elements to IEC 61131-XML.

Note: Multiple vendor specific information may be merged into one IEC 61131-XML element "addData".

8.3.1 Common rules

While transforming IML to IEC 61131-XML each IML system element shall be translated to a corresponding POU element respectively SFC element by means of the mapping rules below. Information given in IML but not directly expressible by IEC 61131-XML shall be mapped to additional data as described in 8.2.

The following mapping provisions apply:

- Each IML system shall be represented by one IEC 61131-XML element "pou". All sub-elements of the IML system shall be represented within the IEC 61131-XML element "pou" (see 8.3.2).
- Each IML system element "header" shall be represented by attributes of the IEC 61131-XML element "pou" (see 8.3.2).
- Each IML system element "state" shall be represented by one IEC 61131-XML element "step" within the IEC 61131-XML element "SFC" (see 8.3.3).
- Each IML system element "state transition" shall be represented by two IEC 61131-XML elements (see 8.3.4):
 - One IEC 61131-XML element "transition" within the corresponding IEC 61131-XML element "SFC".
 - One IEC 61131-XML element "transition" within the IEC 61131-XML element "transitions" of the same "pou".

Note: The IEC 61131-XML element „transition“ (child node of the IEC 61131-XML element „transitions“) can be referenced multiple times by IEC 61131-XML elements "transition" belonging to the IEC 61131-XML element "SFC".

- Each IML system element "activity" shall be represented by two IEC 61131-XML elements (see 8.3.5):
 - One IEC 61131-XML element "action" within an IEC 61131-XML element "actionBlock" within the corresponding IEC 61131-XML element "SFC".
 - One IEC 61131-XML element "action" within the IEC 61131-XML element "actions" of the same "pou".

Note: The IEC 61131-XML element „action“ (child node of the IEC 61131-XML element „actions“) can be referenced multiple times by IEC 61131-XML elements "action" belonging to the IEC 61131-XML element "SFC".

- Each IML system element "selection divergence" shall be represented by one IEC 61131-XML element "selection divergence" within the IEC 61131-XML element "SFC" (see 8.3.6).
- Each IML system element "simultaneous divergence" shall be represented by one IEC 61131-XML element "simultaneous divergence" within the IEC 61131-XML element "SFC" (see 8.3.7).
- Each IML system element "selection convergence" shall be represented by one IEC 61131-XML element "selection convergence" within the IEC 61131-XML element "SFC" (see 8.3.8).
- Each IML system element "simultaneous convergence" shall be represented by one IEC 61131-XML element "simultaneous convergence" within the IEC 61131-XML element "SFC" (see 8.3.9).
- Each IML system element "event" shall be represented by one IEC 61131-XML element "variable" of the data type "event" within the IEC 61131-XML element "interface" (see 8.3.10).
- Each IML system element "variable" shall be represented by one IEC 61131-XML element "variable" within the IEC 61131-XML element "interface" (see 8.3.11).

- Each IML system element “comment” shall be represented by one IEC 61131-XML element “documentation” (see 8.3.12).
- Each IML system element “addData” shall be represented by one element of the AML addData schema according to the IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1 specification (see 8.3.13). Each IEC 61131-XML element shall have zero or one IEC 61131-XML element “addData”. Several IML system elements “addData” shall be combined to only one IEC 61131-XML element “addData”.
- Each property and relation of the IML system elements shall be represented either by IEC 61131-XML element attributes or by sub elements of the corresponding IEC 61131-XML element.

Note: The mapping rules are designed in a way that the combination of several IML system elements to one IEC 61131-XML element is reversible.

According to the structure of the IEC 61131-XML schema all information representing an IML system is represented either in the IEC 61131-XML elements “pou”, “interface”, “actions”, “transitions”, or “SFC” of one POU within one IEC 61131-XML document (see Figure 6).

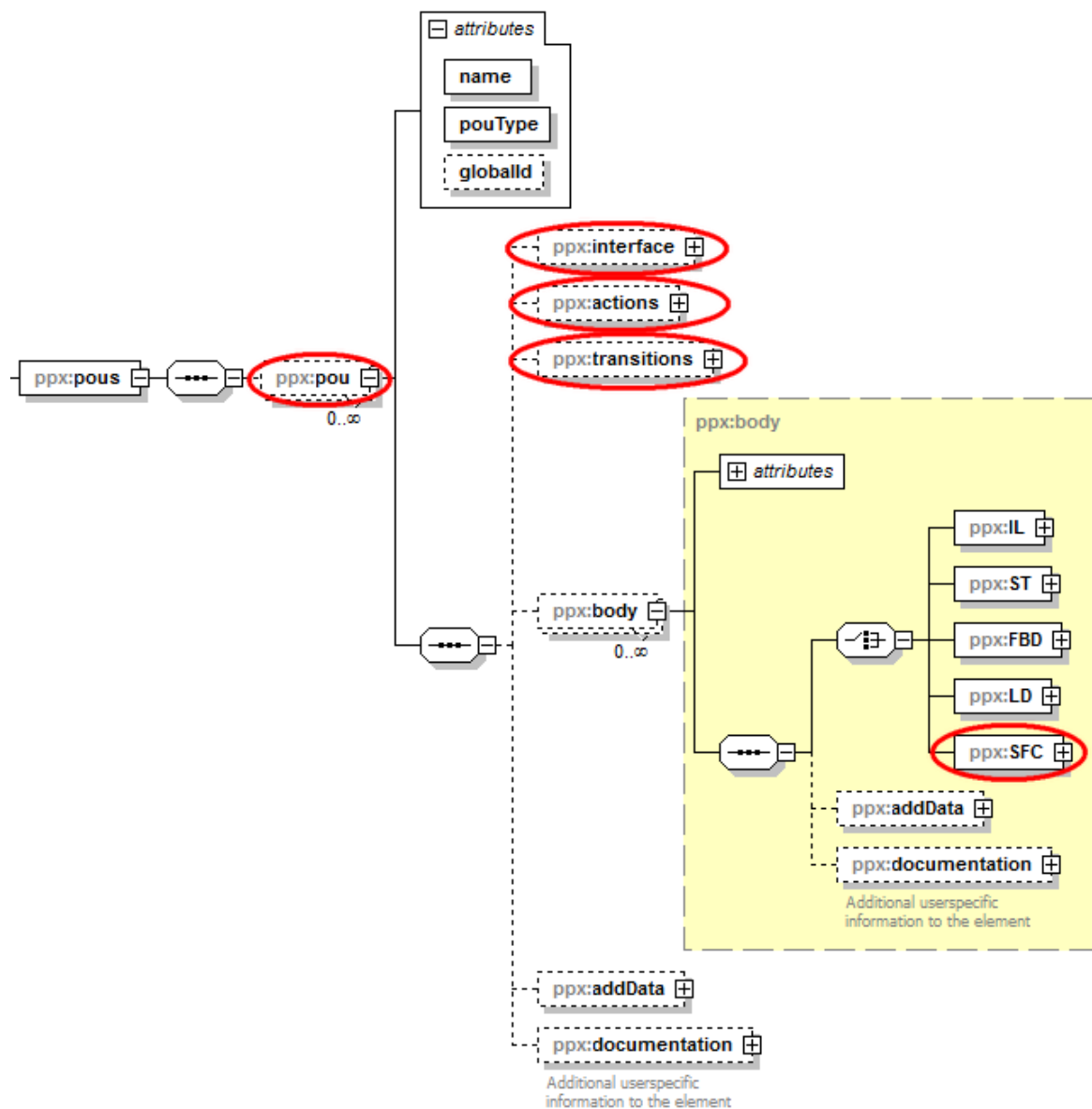


Figure 6. Used parts of the IEC 61131-XML schema to represent a IML system

8.3.2 Mapping of headers

Table 36 specifies the mapping of the properties and the relation of the IML system element “header” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
h	<pre><pou pouType="program"> ... </pou></pre>
Representation as <pou> attributes	
h.ID	globalID="h.ID"
h.Name	name="h.Name"
Representation within the IEC 61131-XML element “pou”	
h.members	<p>“h.members” (e.g. step, transition, action) shall be mapped to sub elements of the IEC 61131-XML element “pou”.</p> <pre><body> <SFC> <"h.members"> ... </"h.members"> </SFC> </body></pre>

Table 36: Mapping of the IML system element “header” to IEC 61131-XML

An example of the mapping of an IML system element “header” is given in Table 37.

IML example	Resulting IEC 61131-XML
h	<pre><pou pouType="program" globalID="UUID100" name="ExampleIML"> <body> <SFC> <step globalID="UUID101" ...> ... </step> </SFC> </body> </pou></pre>
h.ID = UUID100	
h.Name = ExampleML	
h.members = state (UUID101),	

Table 37: Example transformation IML system element “header” to IEC 61131-XML

Table 38 specifies the mapping of the properties and the relation of the IML system element “state” to anIEC 61131-XML element “step”.

Table 38: Mapping of the IML system element “state” to IEC 61131-XML

An example of the mapping of an IML system element “state” is given in Table 39.

IML example	Resulting IEC 61131-XML
s	<pre> <step localId="4" globalId="UUID104" name="IntermediateState" InitialStep="false"> <addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <StateStatus Current="true" Terminal="false"/> </AML> </data> </addData> <connectionPointIn> <connection refLocalId="3"/> </connectionPointIn> </step> </pre>
s.ID = UUID104	
s.Name = IntermediateState	
s.Init = false	
s.Current = true	
s.Terminal = false	
s.Pre = UUID103 Note: The IML system element with the globalID = UUID103, which is the predecessor of s, contains the localID = 3.	

Table 39: Example transformation IML system element “state” to IEC 61131-XML

8.3.4 Mapping of state transitions

Table 40 specifies the mapping of the properties and the relation of the IML system element “state transition” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
st	Representation within the IEC 61131-XML element “transitions”
	<pre> <transition> ... </transition> </pre>
	Representation within the IEC 61131-XML element “SFC”
	<pre> <transition> ... </transition> </pre>
Representation as <transition> attributes	
st.ID	Representation within the IEC 61131-XML element “transitions”
	no representation
	Representation within the IEC 61131-XML element “SFC”
	globalId=“st.ID”
st.Name	Representation within the IEC 61131-XML element “transitions”
	name=“st.Name”
	Representation within the IEC 61131-XML element “SFC”
	no representation

Representation within the IEC 61131-XML element “transition”	
st.Name	Representation within the IEC 61131-XML element “transitions”
	no representation
	Representation within the IEC 61131-XML element “SFC”
	<pre> <condition> <reference name="st.Name"/> </condition> </pre>
st.Guard	<p>Representation within the IEC 61131-XML element “transitions”</p> <p>If “st.Guard” contains an empty guard (see combination 1 in Table 15) the mapping shall be done as follows:</p> <pre> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> st.Name:=TRUE; </xhtml> </ST> </body> </pre> <p>If “st.Guard” contains “st.Guard.Boolean” and the set {e₁, ..., e_n} of “st.Guard.ConsumedEvent” is not empty (see combination 3 in Table 15) the mapping shall be done as follows:</p> <pre> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> IF (st.Guard.Boolean AND e1 AND ... en) THEN st.Name:=TRUE; (*Begin: ConsumedEvent*) e1:=False; (*other ConsumedEvents*) en:=False; (*End: ConsumedEvent*) ELSE st.Name:=FALSE; END_IF; </xhtml> </ST> </body> </pre> <p>If no set {e₁, ..., e_n} of “st.Guard.ConsumedEvent” exists (see combination 2 in Table 15) the entry “AND e1 AND ... en” within the IF expression shall be omitted as well as the entry starting from</p>

“(*Begin: ConsumedEvent*)” until “(*End: ConsumedEvent*)” within the THEN expression.

If “st.Guard” contains “st.Guard.Value” and the set {e₁, ..., e_n} of “st.Guard.ConsumedEvent” is not empty (see combination 5 in Table 15) the mapping shall be done as follows:

<body>

<ST>

<xhtml xmlns="http://www.w3.org/1999/xhtml">

IF (st.Guard.Value.low <= st.Guard.Value.var AND st.Guard.Value.var<= <= st.Guard.Value.high AND e1 AND ... en)

THEN st.Name:=TRUE;

(*Begin: ConsumedEvent*)

e1:=False;

...

en:=False;

(*End: ConsumedEvent*)

ELSE st.Name:=FALSE;

END_IF;

</xhtml>

</ST>

</body>

If no set {e₁, ..., e_n} of “st.Guard.ConsumedEvent” exists (see combination 4 in Table 15) the entry “AND e1 AND ... en” within the IF expression shall be omitted as well as the entry starting from “(*Begin: ConsumedEvent*)” until “(*End: ConsumedEvent*)” within the THEN expression.

If “st.Guard” contains “st.Guard.Formula” and the set {e₁, ..., e_n} of “st.Guard.ConsumedEvent” is not empty (see combination 7 in Table 15) the mapping shall be done as follows:

<body>

<ST>

<xhtml xmlns="http://www.w3.org/1999/xhtml">

IF (st.Guard.Formula AND e1 AND ... en)

THEN st.Name:=TRUE;

(*Begin: ConsumedEvent*)

e1:=False;

...

	<pre> en:=False; (*End: ConsumedEvent*) ELSE st.Name:=FALSE; END_IF; </xhtml> </ST> </body> </pre> <p>If no set {e1, ..., en} of “st.Guard.ConsumedEvent” exists (see combination 6 in Table 15) the entry “AND e1 AND ... en” within the IF expression shall be omitted as well as the entry starting from “(*Begin: ConsumedEvent*)” until “(*End: ConsumedEvent*)” within the THEN expression.</p> <p>“st.Guard.Formula” shall be defined as a string following the syntax of Structured Text of IEC 61131-3; providing a Boolean value as evaluation result.</p> <p>If “st.Guard” only contains the set {e1, ..., en} of “st.Guard.ConsumedEvent” (see combination 8 in Table 15) the mapping shall be done as follows:</p> <pre> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> IF (e1 AND ... en) THEN st.Name:=TRUE; e1:=False; ... en:=False; ELSE st.Name:=FALSE; END_IF; </xhtml> </ST> </body> </pre> <p>Note: For each element of “st.Guard.ConsumedEvent” the name of the IEC 61131-XML element “variable” representing it is integrated in the Structured Text body by (a) integrating its logical value in the transition firing condition and (b) afterwards set to “false”.</p> <p>Note: “(*...*)” represents a comment.</p> <p>Note: “&lt;=” represents a less-than-or-equal sign (<=)</p> <p>Representation within the IEC 61131-XML element “SFC”</p>
--	--

	no representation
st.Pre	Representation within the IEC 61131-XML element “transitions”
	no representation
	Representation within the IEC 61131-XML element “SFC”
	For an IEC 61131-XML element “el” with the local ID “el.x” named in “st.Pre” one IEC 61131-XML element “connectionPointIn” element shall be created as follows: <pre><connectionPointIn> <connection refLocalId=" el.x " /> </connectionPointIn></pre>
	Note: “st.Pre” contains the globalID of the previous IML system element. Note: LocalID's are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.

Table 40: Mapping of the IML system element “state transition” to IEC 61131-XML

An example of the mapping of an IML system element “state transition” is given in Table 41.

IML example	Resulting IEC 61131-XML
st	<pre> <transitions> <transition name="Transition3"> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> IF (c AND e) THEN Transition3:=TRUE; e:=False; ELSE Transition3:=FALSE; END_IF </xhtml> </ST> </body> </transition> </transitions> </pre>
st.ID = UUID106	
st.Name = Transition3	
st.Guard.Boolean = “c”	
st.Guard.ConsumedEvents = “e”	
<p>st.Pre = UUID105</p> <p>Note: The IML system element with the globalID = UUID105, which is the predecessor of st, contains the localID = 5.</p>	<pre> <body> <SFC> <transition localId="6" globalId="UUID106"> <condition> <reference name="Transition3"/> </condition> <connectionPointIn> <connection refLocalId="5"/> </connectionPointIn> </transition> </SFC> </body> </pre>

Table 41: Example transformation IML system element “state transition” to IEC 61131-XML

8.3.5 Mapping of activities

Table 42 specifies the mapping of the properties and relations of the IML system element “activity” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
a	Representation within the IEC 61131-XML element “actions”
	<pre><action> ... </action></pre>
	Representation within the IEC 61131-XML element “actionBlock” within the IEC 61131-XML element “SFC”
	<pre><action> ... </action></pre>
Representation as <action> attributes	
a.ID	Representation within the IEC 61131-XML element “actions”
	no representation
	Representation within the IEC 61131-XML element “SFC” within the “actionBlock”
	globalId=“a.ID”
a.Name	Representation within the IEC 61131-XML element “actions”
	name=“a.Name”
	Representation within the IEC 61131-XML element “SFC” within the “actionBlock”
	no representation
a.Time	Representation within the IEC 61131-XML element “actions”
	no representation
	Representation within the IEC 61131-XML element “actionBlock” within the IEC 61131-XML element “SFC”
	If “a.Time.Duration” is not empty the mapping shall be done as follows: qualifier=“SD” duration=“a.Time.Duration”
	If “a.Time.Duration” is empty and “a.Time.Delay” is not empty the mapping shall be done as follows: qualifier=“D” duration=“a.Time.Delay”
	If “a.Time.Duration” and “a.Time.Delay” is empty the mapping shall be done as follows: qualifier=“D” duration=“0”
	Note: Because the <action> tag only has one “qualifier” which can be used for timing information an addData information is necessary

	to allow more complex timing information representations.
Representation within the IEC 61131-XML element "action"	
a.Name	Representation within the IEC 61131-XML element "actions"
	no representation
	Representation within the IEC 61131-XML element "actionBlock" within the IEC 61131-XML element "SFC"
	<reference name="a.Name"/>
a.Time, a.Init, a.Current, a.Terminal	Representation within the IEC 61131-XML element "actions"
	no representation
	Representation within the IEC 61131-XML element "actionBlock" within the IEC 61131-XML element "SFC"
	If "a.Time.Duration" is not empty the mapping shall be done as follows:
	<pre> <addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <Time> <EarliestStart Value=" a.Time.Start.EarliestStart"/> <LatestStart Value=" a.Time.Start.LatestStart"/> <EarliestEnd Value=" a.Time.End.EarliestEnd"/> <LatestEnd Value=" a.Time.End.LatestEnd"/> <Delay Value=" a.Time.Delay"/> </Time> <ActionStatus Initial="a.Init" Current="a.Current" Terminal="a.Terminal"/> </AML> </data> </addData> </pre>
	If "a.Time.Duration" is empty the mapping shall be made as follows:
	<pre> <addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <Time> <EarliestStart Value=" a.Time.Start.EarliestStart"/> <LatestStart Value=" a.Time.Start.LatestStart"/> <EarliestEnd Value=" a.Time.End.EarliestEnd"/> </pre>

	<pre> <LatestEnd Value=" a.Time.End.LatestEnd"/> </Time> <ActionStatus Initial="a.Init" Current="a.Current" Terminal="a.Terminal"/> </AML> </data> </addData> </pre> <p>If one of the elements "a.Time.EarliestStart", "a.Time.LatestStart", "a.Time.EarliestEnd", or "a.Time.LatestEnd" is empty the corresponding tags of the IEC 61131-XML element "addData" maybe omitted.</p>
a.Formula, a.FiredEvent	Representation within the IEC 61131-XML element "actions"
	<p>If "a.Formula" and the set {e₁,...e_n} of "a.FiredEvent" are not empty the mapping shall be done as follows:</p> <pre> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> (*Begin: ChangedVariables*) a.Formula (*End: ChangedVariables*) (*Begin: FiredEvent*) e1:=true; ... en:=true; (*End: FiredEvent*) </xhtml> </ST> </body> </pre> <p>If no set {e₁, ..., e_n} of "a.FiredEvent" exists the entry starting from "(*Begin: FiredEvent*)" until "(*End: FiredEvent*)" shall be omitted.</p> <p>If "a.Formular" is empty the entry starting from "(*Begin: ChangedVariables*)" until "(*End: ChangedVariables*)" shall be omitted.</p> <p>If "a.Formular" as well as "a. FiredEvent" are empty the entry starting from "(*Begin: ChangedVariables*)" until "(*End: FiredEvent*)" shall be omitted.</p> <p>Note: "(*...*)" represents a comment.</p>
	Representation within the IEC 61131-XML element "actionBlock" within the IEC 61131-XML element "SFC"
	no representation
a.Pre	Representation within the IEC 61131-XML element "actions"
	no representation

	Representation within the IEC 61131-XML element “actionBlock” within the IEC 61131-XML element “SFC”
	<p>For an IEC 61131-XML element “el” with the local ID “el.x” named in “a.Pre” one IEC 61131-XML element “connectionPointIn” shall be created as follows:</p> <pre><connectionPointIn> <connection refLocalId="el.x"/> </connectionPointIn></pre> <p>Note: “a.Pre” contains the globalID of the previous IML system element.</p> <p>Note: LocalID's are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.</p>

Table 42: Mapping of the IML system element “activity” to IEC 61131-XML

An example of the mapping of an IML system element “activity” is given in Table 43.

IML example	Resulting IEC 61131-XML
a	<actions>
a.ID = UUID109	<action name="Action1">
a.Name = Action1	<body>
a.Init = false	<ST>
a.Current = true	<xhtml xmlns="http://www.w3.org/1999/xhtml">
a.Terminal = false	(*FiredEvent*)
a.Formula = “a:=true”	e:=true;
a.FiredEvent = {e}	(*ChangedVariables*)
a.Time.Delay = 6	a:=True
	</xhtml>
	</ST>
	</body>
	</action>
	</actions>
a.Pre = UUID107	<body>
Note: The IML system element with the globalID = UUID107, which is the predecessor of a, contains the localID = 7.	<SFC>
	<actionBlock localId="8">
	<connectionPointIn>
	<connection refLocalId="7"/>
	</connectionPointIn>
	<action localId="9" globalId="UUID109" qualifier="D" duration="6">
	<reference name="Action1"/>
	<addData>

	<pre> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <ActionStatus Initial="false" Current="true" Terminal="false"/> <Time> <Delay Value="6"/> </Time> </AML> </data> </addData> </action> </actionBlock> </SFC> </body> </pre>
--	---

Table 43: Example transformation IML system element “activity” to IEC 61131-XML

8.3.6 Mapping of selection divergences

Table 44 specifies the mapping of the property and relation of the IML system element “selection divergence” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
selDiv	<pre> <selectionDivergence> ... </selectionDivergence> </pre> within the IEC 61131-XML element “SFC”.
Representation as <selectionDivergence> attributes	
selDiv.ID	globalId="selDiv.ID"
Representation within the IEC 61131-XML element “selectionDivergence”	
selDiv.Pre	<p>For an IEC 61131-XML element “el” with the local ID “el.x” named in “selDiv.Pre” one IEC 61131-XML element “connectionPointIn” shall be created as follows:</p> <pre> <connectionPointIn> <connection refLocalId="el.x"/> </connectionPointIn> </pre> <p>Note: “selDiv.Pre” contains the globalID of the previous IML system element.</p> <p>Note: LocalID’s are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.</p>

Table 44. Mapping of the IML system element “selection divergence” to IEC 61131-XML

An example of the mapping of an IML system element “selection divergence” is given in Table 45.

IML example	Resulting IEC 61131-XML
-------------	-------------------------

selDiv	<pre><selectionDivergence localId="13" globalId="UUID113"> <connectionPointIn> <connection refLocalId="12"/> </connectionPointIn> </selectionDivergence></pre>
selDiv.ID = UUID113	
selDiv.Pre = UUID112 Note: The IML system element with the globalID = UUID112, which is the predecessor of selDiv, contains the localID = 12.	

Table 45: Example transformation IML system element “selection divergence” to IEC 61131-XML

8.3.7 Mapping of simultaneous divergences

Table 46 specifies the mapping of the property and relation of the IML system element “simultaneous divergence” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
simDiv	<pre><simultaneousDivergence> ... </simultaneousDivergence></pre> within the IEC 61131-XML element “SFC”.
Representation as <simultaneousDivergence> attributes	
simDiv.ID	globalId=“simDiv.ID”
Representation within the IEC 61131-XML element “simultaneousDivergence”	
simDiv.Pre	<p>For an IEC 61131-XML element “el” with the local ID “el.x” named in “simDiv.Pre” one IEC 61131-XML element “connectionPointIn” shall be created as follows:</p> <pre><connectionPointIn> <connection refLocalId="el.x"/> </connectionPointIn></pre> <p>Note: “simDiv.Pre” contains the globalID of the previous IML system element.</p> <p>Note: LocalID's are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.</p>

Table 46: Mapping of the IML system element “simultaneous divergence” to IEC 61131-XML

An example of the mapping of an IML system element “simultaneous divergence” is given in Table 47.

IML example	Resulting IEC 61131-XML
simDiv	<pre><simultaneousDivergence localId="15" globalId="UUID115"> <connectionPointIn> <connection refLocalId="14"/> </connectionPointIn> </simultaneousDivergence></pre>
simDiv.ID = UUID115	
simDiv.Pre = UUID114 Note: The IML system element with the globalID = UUID114, which is the predecessor of simDiv, contains the localID = 14.	

Table 47: Example transformation IML system element “simultaneous divergence” to IEC 61131-XML

8.3.8 Mapping of selection convergences

Table 48 specifies the mapping of the property and relation of the IML system element “selection convergence” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
selCon	<pre><selectionConvergence> ... </selectionConvergence></pre> within the IEC 61131-XML element “SFC”.
Representation as <selectionConvergence> attributes	
selCon.ID	globalId="selCon.ID"
Representation within the IEC 61131-XML element “selectionConvergence”	
selCon.Pre	<p>For each IEC 61131-XML element “el” with the local ID “el.x” named in “selCon.Pre” one IEC 61131-XML element “connectionPointIn” shall be created as follows:</p> <pre><connectionPointIn> <connection refLocalId="el.x"/> </connectionPointIn></pre> <p>Note: “selCon.Pre” contains the globalID’s of the previous IML system elements.</p> <p>Note: LocalID’s are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.</p>

Table 48: Mapping of the IML system element “selection convergence” to IEC 61131-XML

An example of the mapping of an IML system element “selection convergence” is given in Table 49.

IML example	Resulting IEC 61131-XML
selCon	<pre><selectionConvergence localId="18" globalId="UUID118"> <connectionPointIn> <connection refLocalId="16"/> </connectionPointIn> <connectionPointIn> <connection refLocalId="17"/> </connectionPointIn> </selectionConvergence></pre>
selCon.ID = UUID118	
selCon.Pre = UUID116, UUID117 Note: The IML system elements with the globalID = UUID116 and globalID = UUID117, which are the predecessors of selCon, contains the localID = 16 and localID = 17.	

Table 49: Example transformation IML system element “selection convergence” to IEC 61131-XML

8.3.9 Mapping of simultaneous convergences

Table 50 specifies the mapping of the property and relation of the IML system element “simultaneous convergence” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
simCon	<pre><simultaneousConvergence> ... </simultaneousConvergence></pre> within the IEC 61131-XML element “SFC”.
Representation as <simultaneousConvergence> attributes	
simCon.ID	globalId="simCon.ID"
Representation within the IEC 61131-XML element “simultaneousConvergence”	
simCon.Pre	For each IEC 61131-XML element “el” with the local ID “el.x” named in “simCon.Pre” one IEC 61131-XML element “connectionPointIn” shall be created as follows: <pre><connectionPointIn> <connection refLocalId="el.x"/> </connectionPointIn></pre> Note: “simDiv.Pre” contains the globalID’s of the previous IML system elements. Note: LocalID’s are created according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1.

Table 50: Mapping of the IML system element “simultaneous convergence” to IEC 61131-XML

An example of the mapping of an IML system element “simultaneous convergence” is given in Table 51.

IML example	Resulting IEC 61131-XML
simCon	<pre> <simultaneousConvergence localId="21" globalId="UUID121"> <connectionPointIn> <connection refLocalId="19"/> </connectionPointIn> <connectionPointIn> <connection refLocalId="20"/> </connectionPointIn> </simultaneousConvergence> </pre>
simCon.ID = UUID121	
simCon.Pre = UUID119, UUID120 Note: The IML system elements with the globalId = UUID119 and globalId = UUID120, which are the predecessors of simCon, contains the localId = 19 and localId = 20.	

Table 51: Example transformation IML system element “simultaneous convergence” to IEC 61131-XML

8.3.10 Mapping of events

Table 52 specifies the mapping of the properties of the IML system element “event” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
ev	<pre> <variable> <type> <derived name="event"/> </type> </variable> </pre>
Representation as <variable> attributes	
ev.ID	globalId="ev.ID"
ev.Name	name="ev.Name"

Table 52: Mapping of the IML system element “event” to IEC 61131-XML

An example of the mapping of an IML system element “event” is given in Table 53.

IML example	Resulting IEC 61131-XML
ev	<pre> <interface> <localVars> <variable globalId="UUID122" name="e"> <type> <derived name="event"/> </type> </variable> </localVars> </interface> </pre>
ev.ID = UUID122	
ev.Name = e	

Table 53: Example transformation IML system element “event” to IEC 61131-XML

8.3.11 Mapping of variables

Table 54 specifies the mapping of the properties of the IML system element “variable” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
var	<code><variable></code> ... <code></variable></code>
Representation as <code><variable></code> attributes	
var.ID	<code>globalId="var.ID"</code>
var.Name	<code>name="var.Name"</code>
var.Address	<code>address="var.Address"</code>
Representation within the IEC 61131-XML element “variable”	
var.Type	<code><type></code> <code><"var.Type"/></code> <code></type></code> The defined types according to IEC 61131-XML 2.0 and IEC 61131-XML 2.0.1 shall be used.
var.InitialValue	<code><initialValue></code> <code><simpleValue value="var.InitialValue"/></code> <code></initialValue></code>
var.SIUnit	<code><addData></code> <code><data</code> <code>name="http://www.automationml.org/AML_addData.xsd"></code> <code><AML></code> <code><Unit Name="var.SIUnit"/></code> <code></AML></code> <code></data></code> <code></addData></code>
“Variable” representation within the IEC 61131-XML element “interface”	
var.Content	If “var.Content” has the value “local” the mapping shall be done as follows: <code><localVars></code> ... <code></localVars></code> If “var.Content” has the value “input” the mapping shall be done as follows: <code><inputVars></code> ... <code></inputVars></code>

	<p>If “var.Content” has the value “output” the mapping shall be done as follows:</p> <pre><outputVars> ... </outputVars></pre> <p>If “var.Content” has the value “inout” the mapping shall be done as follows:</p> <pre><inOutVars> ... </inOutVars></pre> <p>The default value of “var.Content” shall be local</p>
--	---

Table 54: Mapping of the IML system element “variable” to IEC 61131-XML

An example of the mapping of an IML system element “variable” is given in Table 55.

IML example	Resulting IEC 61131-XML
var	<pre><interface> <inputVars> <variable globalId="UUID123" name="a" address="%IX0.0"> <type> <INT/> </type> <initialValue> <simpleValue value="9"/> </initialValue> <addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <Unit Name=" var.SIUnit"> </AML> </data> </addData> </variable> </inputVars> </interface></pre>
var.ID = UUID123	
var.Name = a	
var.Address = %IX0.0	
var.Type = INT	
var.InitialValue = 9	
var.SIUnit = m	
var.Content = input	

Table 55: Example transformation IML system element “variable” to IEC 61131-XML

8.3.12 Mapping of comments

Table 56 specifies the mapping of the properties and relation of the IML system element “comment” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
com	<pre> <documentation> <xhtml xmlns="http://www.w3.org/1999/xhtml"> ... </xhtml> </documentation> </pre>
Representation within the IEC 61131-XML element “documentation”	
com.Content	“com.Content”
“Documentation” representation of the IEC 61131-XML elements	
com.Pre	<p>For an IEC 61131-XML element “el” named in “com.Pre” one IEC 61131-XML element “documentation” shall be created within the IEC 61131-XML representation of “el” as follows:</p> <pre> <el globalId="com.Pre"> <documentation> ... </documentation> </el> </pre> <p>Note: “com.Pre” contains the globalID of the corresponding IML system element, to which “com” belongs to.</p>

Table 56: Mapping of the IML system element “comment” to IEC 61131-XML

An example of the mapping of an IML system element “comment” is given in Table 57.

IML example	Resulting IEC 61131-XML
com	<pre> <step globalId="UUID124"> <documentation> <xhtml xmlns="http://www.w3.org/1999/xhtml"> This is a test step. </xhtml> </documentation> </step> </pre>
com.Content = This is a test step.	
com.Pre = UUID124 Note: The comment belongs to a step with the globalID = UUID124.	

Table 57: Example transformation IML system element “comment” to IEC 61131-XML

8.3.13 Mapping of additional data

Table 58 specifies the mapping of the property and relation of the IML system element “additional data” to IEC 61131-XML.

IML system element	Representation in IEC 61131-XML
ad	<pre><addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> ... </AML> </data> </addData></pre>
ad.ID	For further use Note: Non of the transformation rules described here in this standard need this ID.
Representation within the IEC 61131-XML element <addData>	
ad.Type	All possible ad.Value's (see 8.2) shall be created as<ad.Type> attributes.The mapping shall be done as follows: <ad.TypeattributeName="ad.Value"/>
ad.Value	
“addData” representation of the IEC 61131-XML elements	
ad.Pre	For anIEC 61131-XML element “el” named in “ad.Pre” one IEC 61131-XML element “addData” shall be created within the IEC 61131-XML representation of “el” as follows: <pre><el globalId="ad.Pre"> <addData> ... </addData> </el></pre> Note: “ad.Pre” contains the globalID of the corresponding IML system element, to which “ad” belongs to.

Table 58: Mapping of the IML system element “additional data” to IEC 61131-XML

An example of the mapping of an IML system element “additional data” is given in Table 59.

IML example	Resulting IEC 61131-XML
ad	<pre> <step globalId="UUID126"> <addData> <data name="http://www.automationml.org/AML_addData.xsd"> <AML> <StateStatus Current="true" Terminal="false"/> </AML> </data> </addData> </step> </pre>
ad.Type = StateStatus	
ad.Value = Current	
ad.Pre= UUID126 Note: The additional data belongs to a step with the globalID = UUID126.	

Table 59: Example transformation IML system element “additional data” to IEC 61131-XML

9 Usage of mathematical expressions in logic information

9.1 General

This clause defines rules for exploiting mathematical functions for logic information beyond the means of IEC 61131-3. It is described how the Mathematical Markup Language (MathML) can be integrated in IEC 61131-XML for this purpose. MathML is an XML based data format which was developed to enable an exchange of mathematical expressions.

At first, the general integration concept is described giving the essential rules. After defining a 'mapping layer' for mapping IEC 61131-XML variables to the variables defined in MathML it is described how MathML expressions are generally assigned to IEC 61131-XML elements. Additionally, an interpretation of the result of the MathML expression is given in this clause.

9.2 General integration concept

For defining mathematical expressions in the scope of AML logic information MathML version 2.0 content part shall be used. Mathematical expressions shall be given in an IEC 61131-XML element "addData". The IEC 61131-XML element "addData" shall also contain the variable assignment of variables within a mathematical expression to the IEC 61131-XML elements "variable".

For the usage of mathematical expressions in logic information the following provisions apply:

- Mathematical expressions, described within an IEC 61131-XML element "addData", shall be modelled by two IEC 61131-XML elements "data":
 - The first IEC 61131-XML element "data" shall describe the mapping between IEC 61131-XML and MathML.
 - The second IEC 61131-XML element "data" element shall describe the MathML expression itself.
- An IEC 61131-XML element shall have zero or one mathematical expression.

Note: For an unambiguous assignment of the behaviour to an IEC 61131-XML element only one mathematical expression at the IEC 61131-XML element is allowed. But the mathematical expression itself can be composed out of several formulas.

The structure of the IEC 61131-XML elements "data" within an IEC 61131-XML element "addData" is shown in Table 60.

IEC 61131-XML elements "data"	Representation in IEC 61131-XML
mapping layer	<pre><data name="http://www.automationml.org/IEC62714-4Ed1/ MathMLinIEC61131-XML.xsd"> ... </data></pre>
MathML expression	<pre><data name="http://www.w3.org/1998/Math/"> ... </data></pre>

Table 60: Structure of the IEC 61131-XML element "addData"

The mapping concept for MathML variables and IEC 61131-XML elements “variable” is based on a mapping layer. The mapping layer includes the assignment of the name of MathML variables to the IEC 61131-XML attribute “globalID” of IEC 61131-XML elements “variable” as shown in Figure 7.

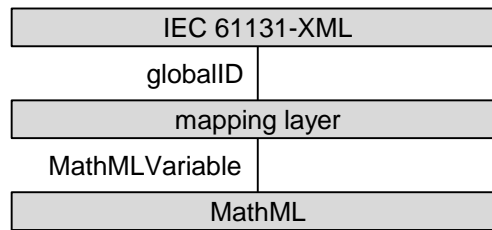


Figure 7: Mapping concept for MathML variables and IEC 61131-XML elements “variable”

The structure of the IEC 61131-XML element “data” for the mapping is defined in 9.2 and the IEC 61131-XML element “data” for MathML expressions in 9.4. For the assignment of the result of MathML expressions to a variable different cases can be distinguished, which are described in 9.5.

Mathematical expressions shall be only assigned to:

- the following IEC 61131-XML “SFC” elements:
 - action
 - pou
 - step
 - transition
 - variable
- the following IEC 61131-XML “FBD” elements:
 - block
 - FbdNetwork
 - pou
 - variable

9.3 Mapping between MathML variables and IEC 61131-XML elements “variable”

9.3.1 General

The mapping of MathML variables and IEC 61131-XML elements “variable” takes place within the mapping layer. The mapping layer shall be the first IEC 61131-XML element “data” within an IEC 61131-XML element “addData”.

For the mapping layer the following provisions apply:

- The IEC 61131-XML element “data” shall have the attribute “name” with the value “<http://www.automationml.org/IEC62714-4Ed1/MathMLinPLCopenXML.xsd>”.

Note: “<http://www.automationml.org/IEC62714-4Ed1/MathMLinPLCopenXML.xsd>” is the schema of the mapping layer.

- The IEC 61131-XML element “data” shall include one “formula” element with the following attributes:
 - “Name”: name of the MathML expression
 - “ID”: unique ID within the whole IEC 61131-XML document

Note: It is recommended to use a UUID.

- The “formula” element shall have one or more “variable” elements.
- Every “variable” element shall have the following attributes:

- “direction”: defines, that a variable is an input (“In”) or an output (“Out”) within the mathematical expression
- “refMathMLVariable”: name of the referenced MathML variable
- “refGlobalId”: IEC 61131-XML attribute “globalID” of the referenced IEC 61131-XML element “variable”

The structure of the mapping layer within an IEC 61131-XML element “addData” is shown in Figure 8.

```
<data name="http://www.automationml.org/IEC62714-4Ed1/MathMLinIEC61131-XML.xsd">
  <formula Name="integrator" ID="UUID34">
    <variable refMathMLVariable="x" refGlobalID="UUID35" direction="In"/>
    <variable refMathMLVariable="y" refGlobalID="UUID36" direction="Out"/>
  </formula>
</data>
```

Figure 8: XML text of the mapping of MathML variables and IEC 61131-XML elements “variable”

The schema for the mapping layer is defined in 9.3.3 and in I.2.

9.3.2 Declaration and usage of the XML schema for variable mappings for MathML in IEC 61131-XML

To use the XML schema for variable mappings for MathML in an IEC 61131-XML document, the schema shall be listed in the IEC 61131-XML element “addDataInfo” to enable the unambiguous identification of the corresponding IEC 61131-XML element “addData” contents (see Figure 10).

```
<contentHeader name="logic model">
  ...
  <addDataInfo>
    <info name="http://www.automationml.org/IEC62714-4Ed1/MathMLinIEC61131-XML.xsd" vendor="AutomationML">
      <description>
        <xhtml:p>Schema for variable mappings for MathML usage within IEC 61131-XML.</xhtml:p>
      </description>
    </info>
  </addDataInfo>
</contentHeader>
```

Figure 9: XML text of the declaration of the XML schema for variable mappings for MathML in IEC 61131-XML

9.3.3 XML schema description of formula data

Table 61 specifies the use of the schema element “formula”.

Diagram	
Properties	content: complex Followed by a “sequence”.
Children	formula:variable
Attributes	Attribute: Name (from type “xs:string” with use “required”) Attribute: ID (from type “xs:ID” with use “required”)

Table 61: MathML schema element “formula”

9.3.4 XML schema description of variable data

Table 62 specifies the use of the schema element “variable”.

Diagram	
Type	xs:complextyp
Properties	minOcc: 0 content: complex
Attributes	Attribute: refMathMLVariable (from type “xs:string” with use “optional”) Attribute: refGlobalID (from type “xs:ID” with use “required”) Attribute: direction (from type “xs:string”, derived by “restriction”, with use “required”) Enumeration: In Enumeration: Out

Table 62: MathML schema element “variable”

9.4 Assignment of MathML expressions to IEC 61131-XML elements

The assignment of MathML expressions to IEC 61131-XML elements shall be done via the IEC 61131-XML element “data” for MathML expressions within an IEC 61131-XML element “addData”. The IEC 61131-XML element “data” shall have the structure as shown in Figure 10 to represent a MathML expression.

```
<data name="http://www.w3.org/1998/Math/">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    ...
  </math>
</data>
```

Figure 10: XML text of the IEC 61131-XML element “data” for the MathML expression

9.5 Assignment of results of a MathML expression to IEC 61131-XML elements “variable”

Results of a MathML expression are

- the value of the mathematical expression of this MathML expression or
- the values of target variables within the mathematical expression of this MathML expression.

Within a MathML expression, all variables and results shall be determined. The following provisions are made to ensure that the MathML expressions can be evaluated:

- If the evaluation of the mathematical expression of this MathML expression results in a value, the following provisions apply:
 - Exactly one output variable assignment within the mapping layer is permitted
 - Within the output variable assignment the attribute “refMathMLVariable” shall not be included
- If the evaluation of the mathematical expression of this MathML expression results in assignments of values to target variables, the following provisions apply:
 - For each target variable an output variable assignment within the mapping layer shall be done
 - Within each output variable assignment the attribute “refMathMLVariable” shall be included
 - The equations within the mathematical expression of this MathML expression should be solvable.

Note: The assignment of variables shall be done within the mapping layer. The value of the attribute “direction” of the “variable” element shall have the value “Out” for output variables.

Figure 11 depicts the assignment of results of a MathML expression to IEC 61131-XML element "variable".

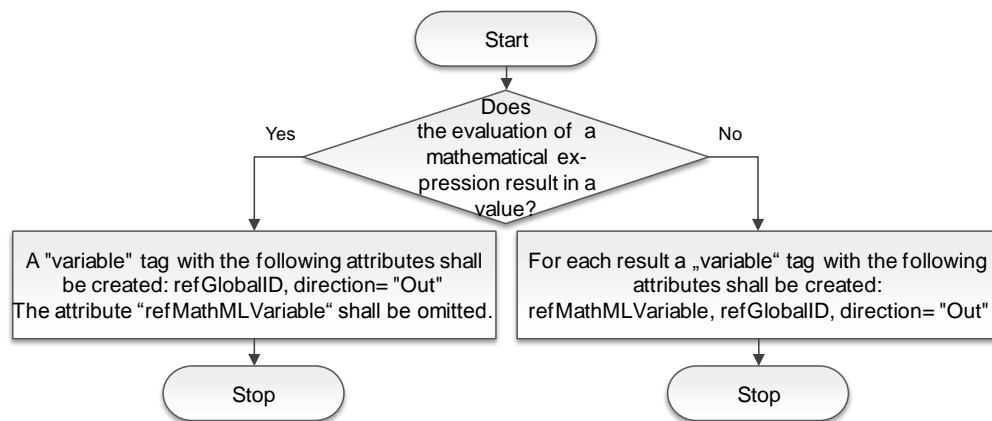


Figure 11: Decision tree for the assignment of results of MathML expressions to IEC 61131-XML elements "variable"

An informative overview of the usage of mathematical expressions in logic information is provided in Annex E.

10 Linking AML objects with interlocking information

10.1 General

Interlocking information is an important part of the engineering of production systems to ensure their safe behaviour. This clause focuses on the interlocking concept which comprises two levels of detail to model and store interlocking information in AML. An informative overview of interlocking concept is provided in Annex F.

10.2 Referencing interlocking information

On the first level of the interlocking concept, interlocking source groups and interlocking target groups are linked to describe functional dependencies among AML object groups respectively AML objects. For this, the following provisions apply:

- An interlocking target group shall be created by using the AML Group concept according to IEC 62714-1.
 - For identifying the group as an interlocking target group the role class “InterlockingTargetGroup” shall be used, which is described in 4.3.1.

Note 1: An interlocking target group is a set of objects which belongs to the same interlocking target group.

Note 2: The objects of an interlocking target group execute the actions which are necessary to ensure functional safety.

- An interlocking source group shall be created by using the AML Group concept according to IEC 62714-1.
 - For identifying the group as an interlocking source group the role class “InterlockingSourceGroup” shall be used, which is described in 4.3.2.

Note 1: An interlocking source group is a set of objects which belongs to the same interlocking source group.

Note 2: The objects of an interlocking source group indicate when actions are necessary to provide functional safety.

- Each interlocking target group and each interlocking source group shall have one CAEX ExternalInterface of the AML InterfaceClass “InterlockingConnector”, which is described in 4.4.9.
 - The CAEX ExternalInterface of the interlocking source group shall be linked to the CAEX ExternalInterface of the corresponding interlocking target group via an InternalLink.
 - The state of the interlocking source group shall affect the interlocking target group.

Note: The assignment between both groups is not restricted to a 1:1 relation. Several interlocking source groups can exist which are connected to one or more interlocking target groups.

On the second level of the interlocking concept, the simple grouping of AML objects into interlocking source groups and interlocking target groups (of the first level) is extended by a function describing the interlocking condition of interlocking source groups. For this, the following provisions apply:

- The function shall be represented by a single POU or a network of POUs.
- The function shall be expressed by Function Block Diagram (FBD).
- The FBD shall result in a unique Boolean variable describing the output respectively evaluation result of this FBD.
- This unique Boolean variable shall represent the interlocking condition of the interlocking source group which shall affect the interlocking target group.

Note: The value of the Boolean variable can be “true” or “false”. This indicates whether the safe state of the interlocking source group is represented by a “true” or “false” value.

- The unique Boolean variable shall be referenced within the interlocking source group modelling one CAEX ExternalInterface of the AML InterfaceClass “InterlockingVariableInterface”.
- This CAEX ExternalInterface shall have zero or one attribute “SafeConditionEquals” to store the Boolean value which indicates the safe state.
- The referencing between variables of POU within a POU network, which is distributed throughout several IEC 61131-XML documents, shall be done by using InternalLinks connecting the corresponding CAEX ExternalInterfaces of the AML InterfaceClass “VariableInterface” within the interlocking source group according to clause 6.

Annex A Logic information in AML

A.1 Logic information in production system engineering

This part focuses on the storage of logic information - an important aspect for electrical design, HMI development, PLC and robot control programming, for simulation purposes, and virtual commissioning.

To support different phases in the iterative production system engineering process covering different levels of detail, AML needs to be able to cover logic information from different tools and disciplines. Thus, different types of logic information belonging to a production system or to single components have to be stored. This variety of information can be differentiated into: sequencing information and behaviour information, which serve different purposes, but those can overlap depending on the point of view and utilization of elements in AML (see *Figure A.1*).

- Sequencing information is logic information describing the controlled behaviour of a system or a part of it. In its most detailed form, it is often represented by a program executed in one or more controllers, e.g. robot controller or PLC. But the development of sequences starts earlier in the engineering process: on a high level of abstraction - with models which model, e.g. the required operations of a larger scale unit (cell, line, plant etc.) and ends with highly detailed, executable programs.
- Behaviour information is logic information describing how a system or a part of it responds to the controlled behaviour (also called uncontrolled behaviour). It is often represented by a given model which reacts on external input, e.g. behaviour of a gripper or valve. Those external inputs would trigger the behaviour or signalling of the gripper's states.
- Intelligent devices may have both logic information assigned to it, sequencing information and behaviour information, e.g. robot or conveyor. From an external point of view, the devices can be controlled respectively externally triggered. From an internal point of view, the behaviour of the device could be described as sequences.

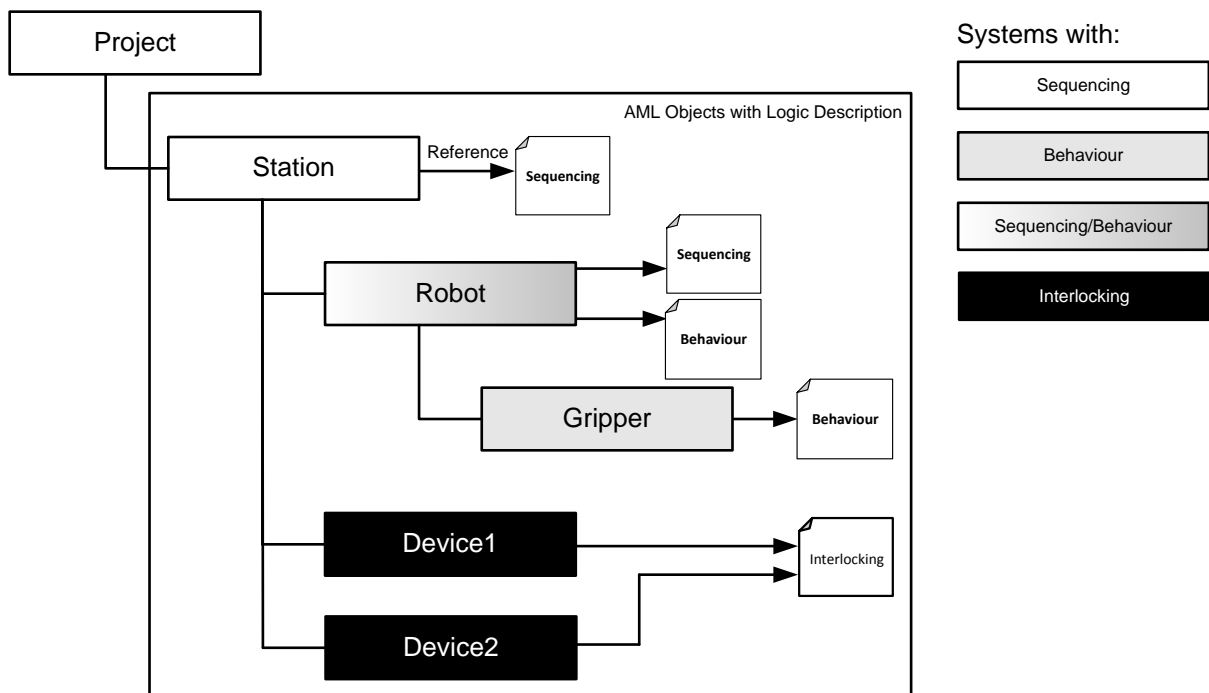


Figure A.1: Types of logic information in AML

Besides sequencing information and behaviour information, interlocking information is a third complementary type of logic information (see *Figure A. 1*). It is an important part within the engineering of production systems to ensure their safe behaviour. It affects not only the safe interaction with humans and the environment but it also helps to avoid unstable states of the systems possibly causing harmful effects on humans and environment or damages on machines and products.

A.2 Logic models in production system engineering

To be applicable within the engineering process of production systems, AML needs to support typical logic models– storing the logic information: which cover each phase of the engineering process (see *Figure A.2*). These logic models are the following:

- Gantt charts are used mainly within early phases of the engineering process to represent the timing and duration of manufacturing processes on a high level of abstraction (see B.2).
- Activity-on-node networks are applied mainly within early phases of the engineering process to represent the timing and interdependence of manufacturing processes with the capability to store timing information (see B.3).
- Timing diagrams are used mainly within the later phases of the engineering process to describe the devices with its states and their timing relationships (see B.4). Real signals are introduced and it is possible to express sequencing information.
- State charts are one of the UML language units of the ISO/IEC 19505 applicable within the entire engineering process of production systems (see B.5). Modelling of behaviour information is possible on different levels of abstraction.
- Sequential Function Charts (SFC) are one of the programming languages of the IEC 61131-3 mainly applied within later phases of production system engineering (see B.5.3). Modelling of sequencing information as well as behaviour information is possible.
- Function Block Diagrams (FBD) are one of the programming languages of the IEC 61131-3 mainly applied within later phases of production system engineering (see B.5.4). Modelling of interlocking information as well as behaviour information is possible.

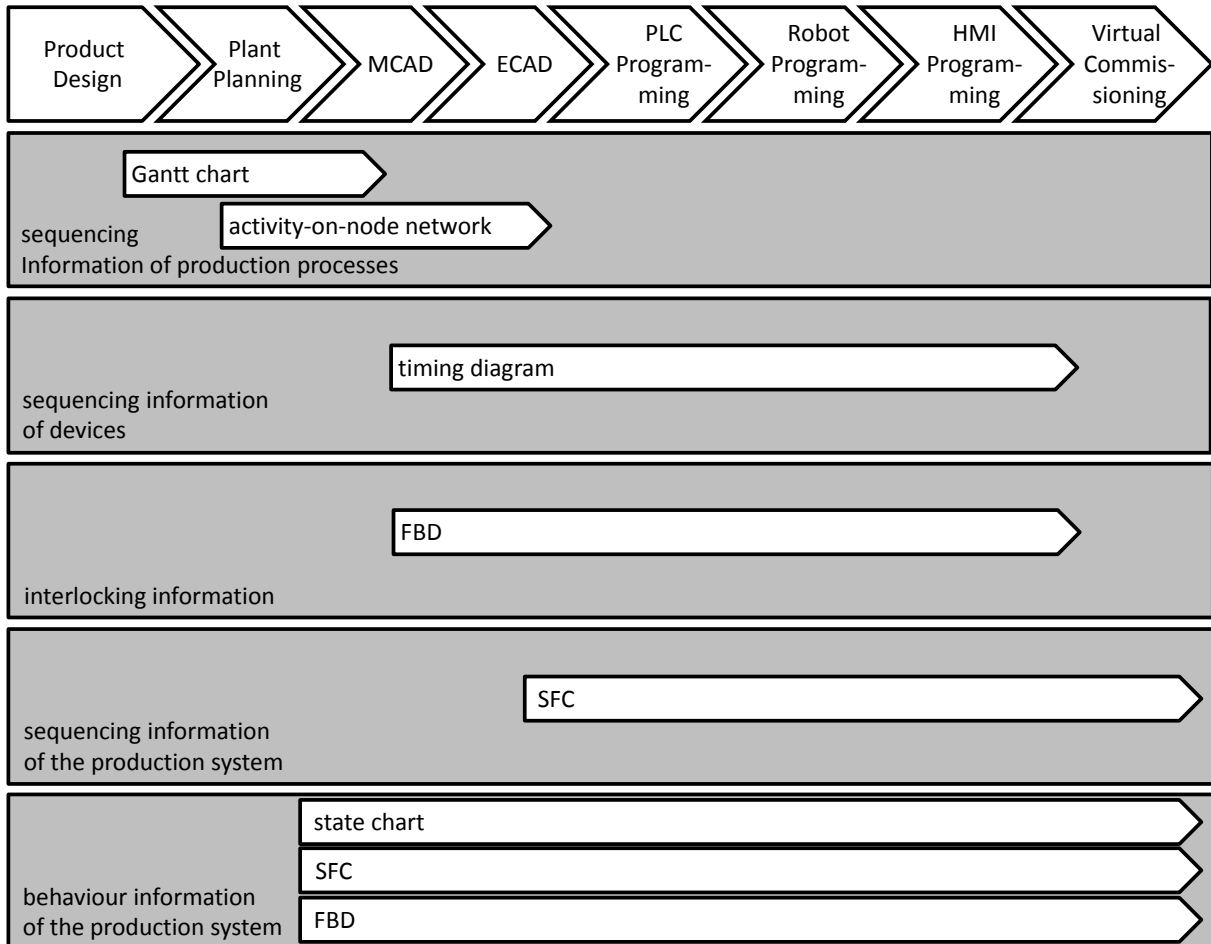


Figure A.2: Logic models in AML

A.3 Storing logic models in AML

AML uses one common data model to store the different logic models, namely Sequential Function Charts (SFC) as one of the PLC programming languages defined in IEC 61131-3. Thus, the logic models need to be translated into a SFC stored as an IEC 61131-XML representation: the target format.

To decouple the target format from the various logic models, AML defines an Intermediate Modelling Layer (IML). Normative provisions are defined in clause 3 and 7. In this way, the complex transformation rules to IEC 61131-XML need to be defined only once (see 8.3), for each logic model only transformation rules to the simpler IML need to be defined (see Annex C). This enhances the extensibility of AML for new logic models as shown in Figure A.3.

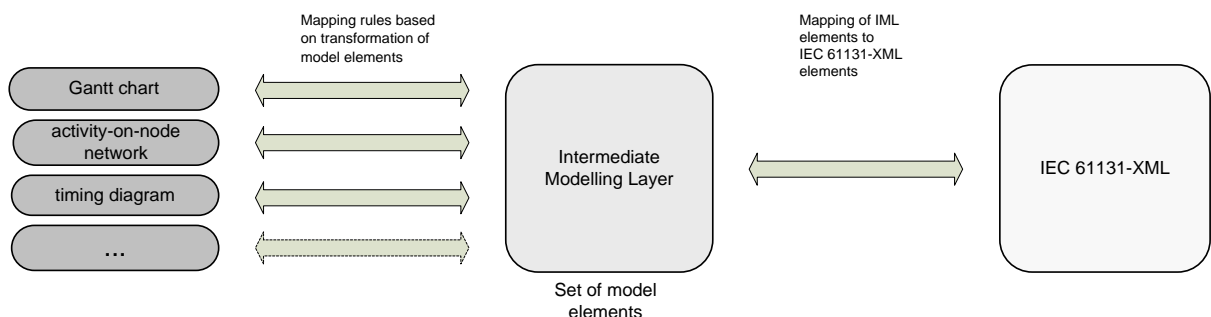


Figure A.3. Transformation to IEC 61131-XML

Using those transformation rules and the IML, not only a transformation from one logic model to IEC 61131-XML is possible. It is also possible to transform one logic model into another logic model. This is essential since the engineering process of production systems is a process of information enrichment. In the beginning, the sequencing information of a production system is only roughly described by using a Gantt chart and needs to be detailed in the next step. So, the Gantt chart is transformed into an activity-on-node network: The information from the Gantt chart is taken over and the complex timing conditions can be added now. Theoretically, this workflow can be done until the executable program.

The logic models have different modelling powers. So, the transformation from one logic model to another, e.g. from activity-on-node network to Gantt chart, will cause information loss.

All in all, this part is not about to simply store PLC programs. It is about making the logic models, emerging during the engineering process of production system, exchangeable. A re-entry of information within the subsequent engineering tool is, therefore, not necessary. The scope of this part is given in *Figure A.4*.

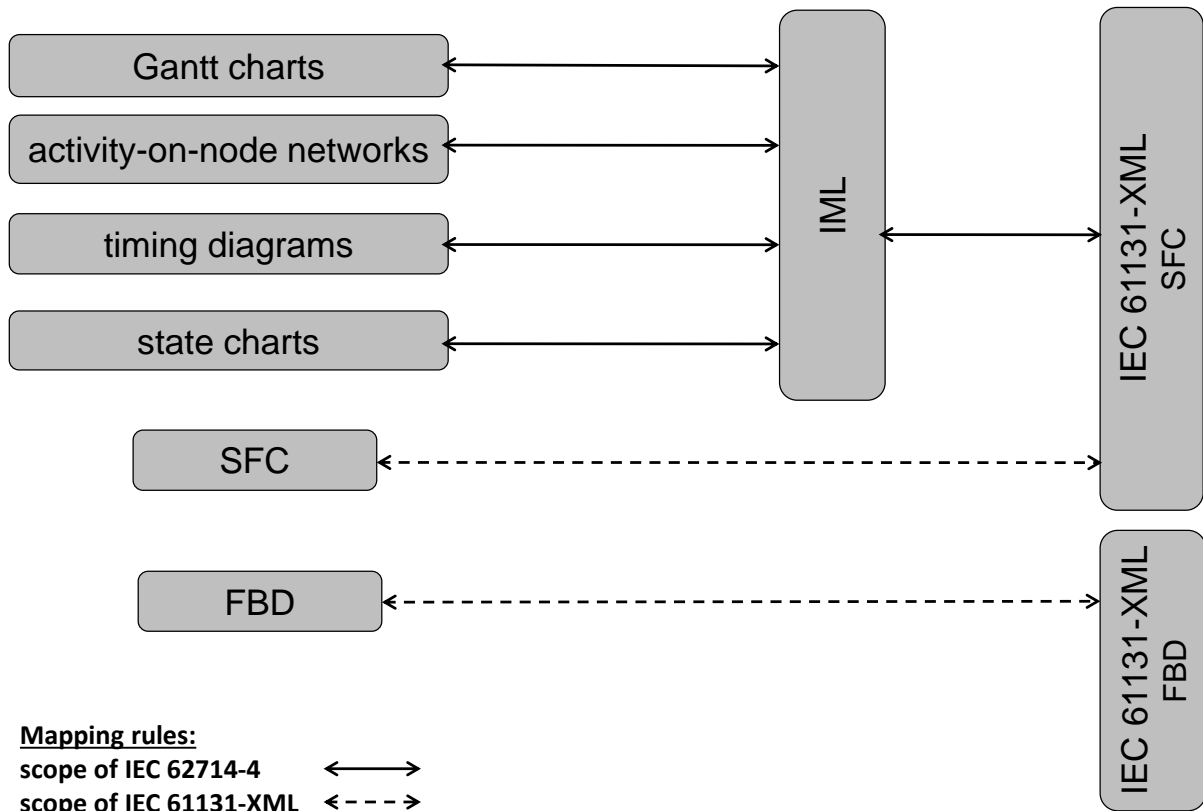


Figure A.4: Scope of IEC 62714-4 regarding mapping rules

A.4 Storing additional information to logic models in AML

To enable IEC 61131-XML to be applicable in the scope of AML, additional information needs to be covered.

1.1.1 Additional logic model specific information

Since IEC 61131-XML intends to store and exchange all relevant programming information for IEC 61131-3 PLC programming projects, and AML goes beyond this scope and use it to store logic models, additional information needs to be stored. This logic model specific information, not covered by IEC 61131-XML, is covered by integrating an additional schema: the so called AML addData schema. Normative provisions are defined in 8.2.

1.1.2 Meta information on file level

To enable a proper data exchange in the scope of AML, meta information on file level are necessary to identify and characterize the file. Normative provisions are defined in clause 5.

1.1.3 Mathematical expressions

Application dependent there can be a need to store mathematical expressions to a logic model. This goes also beyond the scope of IEC 61131-3 and, therefore, also of IEC 61131-XML. Clause 9 defines the normative provisions necessary to integrate mathematical expressions into IEC 61131-XML within the scope of AML. An informative overview is provided in Annex E.

A.5 Referencing logic information in AML

In the multi-document architecture of AML, logic information is stored in separate documents following the IEC 61131-XML data format. Modelling logic information is, therefore, divided into two parts. First, an object, which has logic information, is modelled in CAEX, the top-level format. Second, an IEC 61131-XML document has to be provided containing this logic information. Finally, a reference between the CAEX object and the IEC 61131-XML document needs to be created. Normative provisions for referencing all of the three types of information are defined in clause 4 and 6. An informative overview is provided in Annex D.

But for referencing interlocking information, additional provisions need to be considered. These are normatively defined in clause 10. An informative overview is provided in Annex D.

Annex B Logicmodels in AML

B.1 General

This clause defines the logic models which are considered in this part of IEC 62714, which information they store, and how this is mapped to the model elements.

B.2 Gantt charts

B.2.1 General

Gantt charts are a graphical representation of a schedule typically used to model partially ordered sets of sequentially and concurrently running activities comprising execution order and execution time of the activities. They do not provide means for modelling alternatives and cycles.

Note: Gantt charts are also known as bar charts.

Note: The information represented by a Gantt chart can also be represented, e.g. by a table or spreadsheet.

B.2.2 Model elements

In a Gantt chart the following model elements shall be used (see Figure B.1):

- One or more bars;
- One y-axis listing all the names of the corresponding bars;
- One time line (x-axis) realised as a global clock;
- Zero or more arrows.

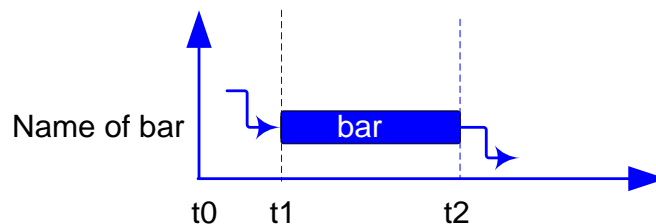


Figure B.1: Model elements of Gantt charts

B.2.3 Chart structuring

The following structuring for a Gantt chart shall be used:

- Each Gantt chart shall be represented by a coordinate system (x-axis: time; y-axis: listing of the name of the bars).
- Each bar shall have its own line in the coordinate system with its corresponding name on the y-axis.
- The timed start point and timed end point of a bar shall correspond with the x-axis. This shall represent the length of the bar. The length of the bar shall represent the duration of a bar.
- An arrow shall connect the right side of one bar with the left side of another bar. Several bars shall be connected by a branched arrow.
 - A starting bar shall be represented by a bar with an outgoing arrow.
 - An ending bar shall be represented by a bar with an incoming arrow.
 - An isolated bar shall be represented by a bar with zero outgoing and zero incoming arrows.
 - A bar, which is a part of the sequence of bars, shall have an outgoing and incoming arrow.

B.2.4 Amount of information

Gantt charts shall comprise the following information:

- Name of an activity;
- Start time of an activity;
- Finish time of an activity;
- Duration of an activity;
- Predecessor-successor-relation among activities;
 - Starting activity;
 - Ending activity;
 - Isolated activity;
 - Sequence of activities.

B.2.5 Mapping rules

For the mapping of the information to the model elements of a Gantt chart the following provisions apply (see Figure B.2):

- An activity shall be represented by a bar.
- A predecessor-successor-relation among activities shall be represented by an arrow.

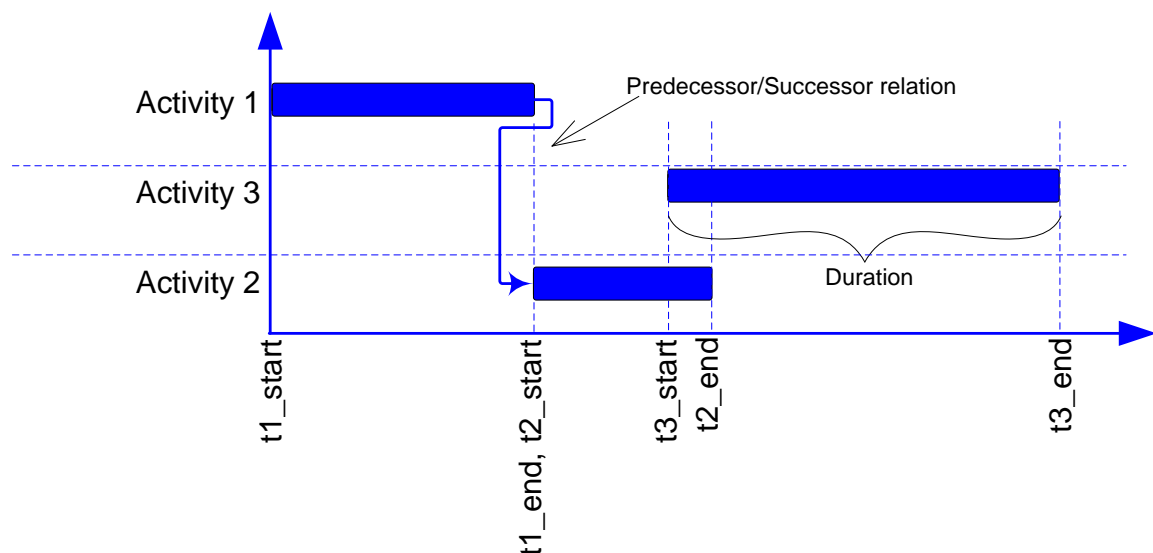


Figure B.2: Information provided by Gantt charts

B.3 Activity-on-node networks**B.3.1 General**

Activity-on-node networks are used to describe and analyse temporal and causal relations of a set of interdependent activities. The end-start-relation between the nodes states that a node can only start after the previous node has ended. Activity-on-node networks do not provide means for modelling alternatives and cycles.

B.3.2 Model elements

In an activity-on-node network the following model elements shall be used (see Figure B.3):

- One or more nodes with seven boxes;
- Zero or more arrows.

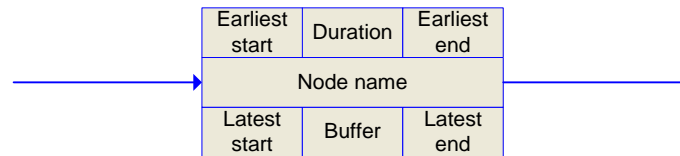


Figure B.3. Model elements of activity-on-node networks

B.3.3 Network structuring

The following structuring for an activity-on-node network shall be used:

- The earliest start of a node shall be located in the box in the upper left corner of the node.
- The duration of a node shall be located in the box in the upper center corner of the node.
- The earliest end of a node shall be located in the box in the upper right corner of the node.
- The node name shall be located in the box in the center of the node.
- The latest start shall be located in the box in the lower left corner of the node.
- The buffer shall be located in the box in the lower center corner of the node.
- The latest end shall be located in the box in the lower right corner of the node.
- An arrow shall connect the right side of one node with the left side of another node. Several nodes shall be connected by a branched arrow.
 - A starting node shall be represented by a node with an outgoing arrow.
 - An ending node shall be represented by a node with an incoming arrow.
 - An isolated node shall be represented by a node with zero outgoing and zero incoming arrows.
 - A node, which is a part of the sequence of node, shall have an outgoing and incoming arrow.

B.3.4 Amount of information

Activity-on-node networks shall comprise the following information:

- Name of an activity;
- Earliest start time of an activity;
- Latest start time of an activity;
- Earliest end time of an activity;
- Latest end time of an activity;
- Duration of an activity;
- Buffer time of an activity;
- Predecessor-successor-relations among activities;
 - Starting activity;
 - Ending activity;
 - Isolated activity;
 - Sequence of activities.

B.3.5 Mapping rules

For the mapping of the information to the model elements of a activity-on-node network the following provisions apply (see Figure B.4):

- An activity shall be represented by a node.
- A predecessor-successor-relation among activities shall be represented by an arrow.

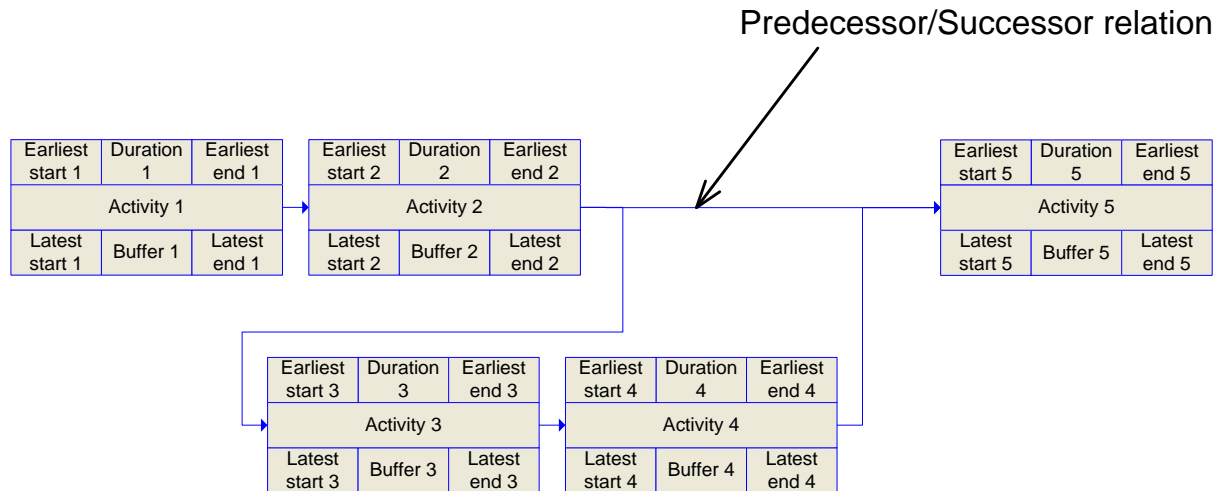


Figure B.4. Information provided by activity-on-node networks

B.4 Timingdiagrams

B.4.1 General

Timing diagrams are used to describe the temporal and causal relations between signals and system states. Furthermore, they describe how changes of system states are affected by signals and how signals are generated by states changes. Timing diagrams enable the modelling of alternatives.

B.4.2 Model elements

In a timing diagram the following model elements shall be used (see Figure B.5):

- One column for resource;
- One column for states;
- One or more lines for resources;
- Two or more lines for states;
- One or more arrows;
- One time line (x-axis) realised as a global clock;
- One or more pseudo-waveforms.

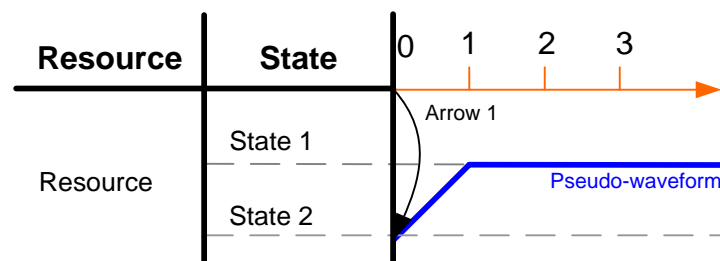


Figure B.5: Model elements of timing diagrams

B.4.3 Diagram structuring

The following structuring for a timing diagram shall be used:

- Each state shall have its own line.
- States which belong to one resource shall be grouped as one line in the resource column. One resource shall be represented by two or more states.
- Each resource shall have one pseudo-waveform.
- A pseudo-waveform shall represent a sequence of states and state changes of one resource.
- An arrow shall connect:
 - The time line with one or more pseudo-waveforms,
 - A pseudo-waveform of one resource with one or more pseudo-waveforms of other resources,
 - A pseudo-waveform of one resource with the pseudo-waveform of the same resource.
- An arrow shall induce a change of the state in one or more pseudo-waveforms.
- A change of the state of a resource shall create zero or more arrows.
- Arrows shall be created at any point at the time line.

B.4.4 Amount of information

Timing diagrams shall comprise the following information:

- Name of a resource;
- Name of a resource state;
- Name of a signal;
- Point in time when a signal occurs;
- Influence of a signal on the state of a resource;
- Duration of a resource state;
- Duration of a resource state change;
- Resource state flow;
- Delay between two subsequent external signals (called “signal time delay”);
- Duration of a resource state change or a predefined duration within a resource state (called “time delay”).

B.4.5 Mapping rules

For the mapping of the information to the model elements of a timing diagram the following provisions apply (see Figure B.6):

- A device shall be represented by a resource.
- A device state shall be represented by a state.
- A signal shall be represented by an arrow.
- A device state flow shall be presented by a pseudo-waveform.

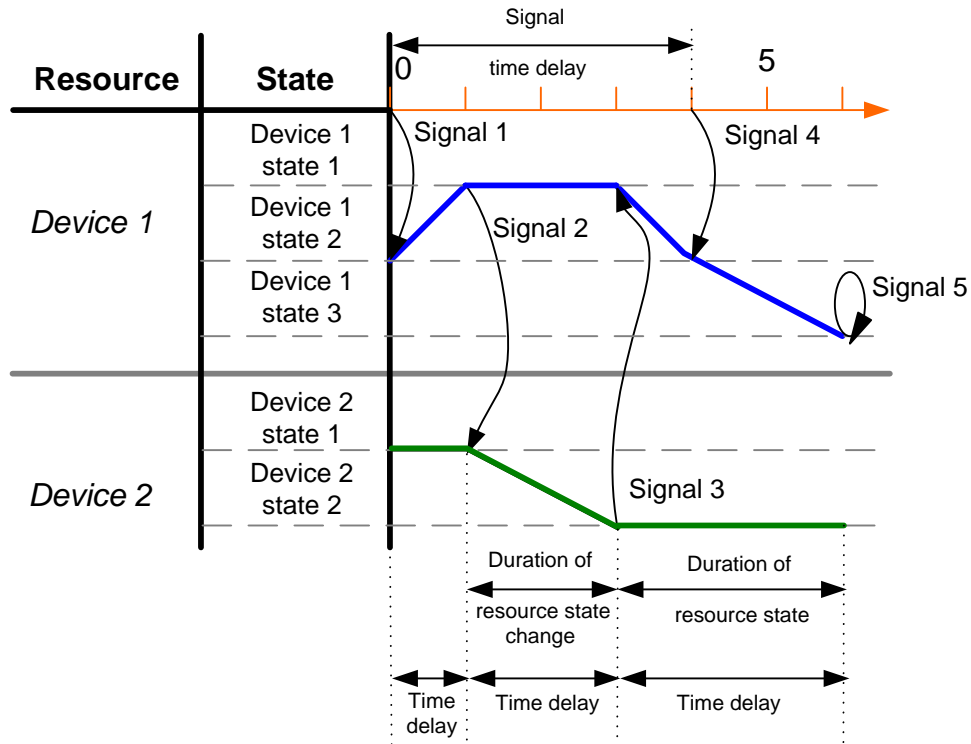


Figure B.6: Information provided by timing diagrams

B.5 State charts

B.5.1 General

This part of IEC 62714 does not need the entire scope of state machines as specified in ISO/IEC 19505. The model elements which are necessary to cover the scope of AML are listed in B.5.2 and mapped to the state machine model elements. And this model is named, in this standard, state chart.

State charts are discrete event-driven models which describe the behaviour of systems.

Chart structuring, amount of information, and mapping rules of state charts are in compliance with state machines as defined in ISO/IEC 19505.

B.5.2 Model elements

In a state chart the following model elements shall be used (see Table B.1 and Figure B.7):

State chart model elements	State machine model elements
State	State
State transition	Transition
Action	Activity
Guard	Constraint
Event	Trigger
Signal	Trigger
History connector	Deep history/shallow history pseudo state
Condition connector	Join/fork/junction/choice pseudo state
Initial state	Initial pseudo state
Terminal state	Terminate pseudo state
Entry action	Activity

Do action	Activity
Exit action	Activity
Region	region

Table B.1: Mapping of model elements of state charts to model elements of state machines

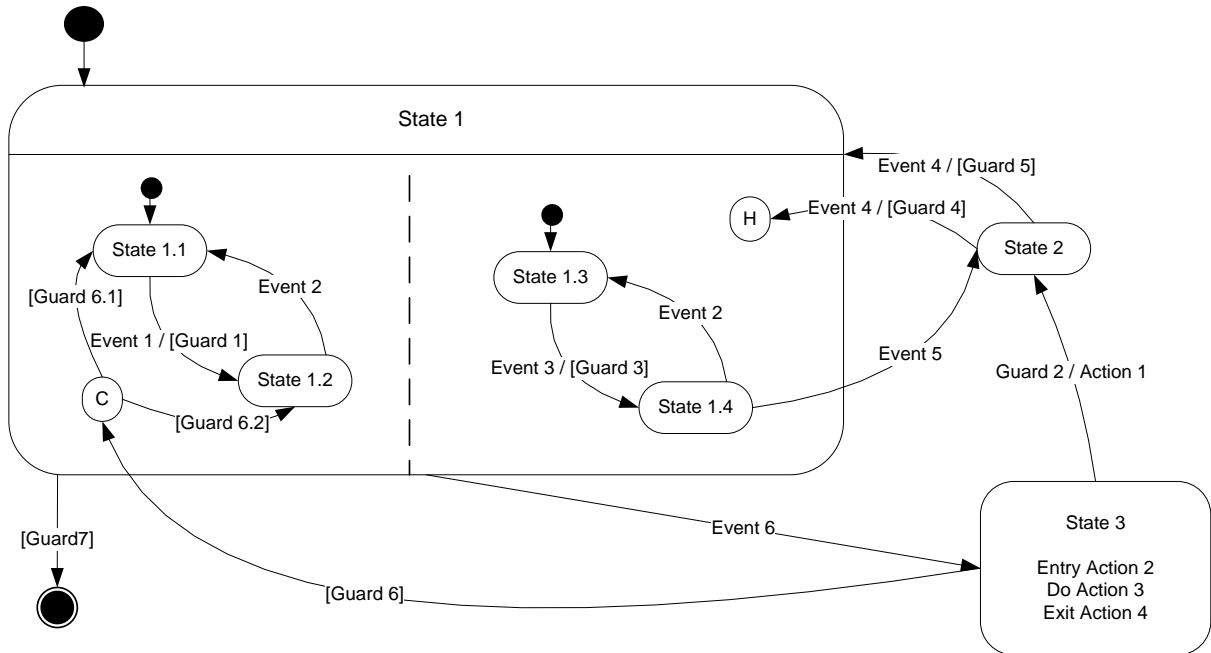


Figure B.7: Model elements of state charts

B.5.3 Sequential function charts

Sequential Function Charts (SFC) describe causal and temporal dependencies of sequences of system states and state transitions. The modelling of complex sequences with loops and conditional execution is possible.

This part of IEC 62714 shall use SFCs as specified in IEC 61131-3.

B.5.4 Function block diagrams

Function Block Diagrams (FBD) describe logical networks of function blocks respectively how input and output variables are connected functionally.

This part of IEC 62714 shall use FBDs as specified in IEC 61131-3. The usage of function blocks specified in IEC 61131-3 shall be allowed.

Annex C Mapping of logic models

C.1 General

This clause defines rules for the mapping of logic models to IML systems. For each logic model element mapping a textual representation in the IML and a graphical representation as a SFC is given. The SFC representation is not binding and only for clarification. Examples are given in Annex G.

C.2 Mapping of Gantt charts to IML

C.2.1 Common rules

For mapping a Gantt chart to an IML system, the following provisions apply:

- Each Gantt chart shall be mapped to an IML system.
- For the start of a Gantt chart the following provision applies:
 - For each IML representation of a Gantt chart, one initial “state” (named “InitialStep”) shall be created, followed by a “state transition” with condition “true” as a link to all further elements, see C.2.2.
- For bars in a Gantt chart the following provisions apply:
 - Each Gantt bar shall be represented in an IML system by a state with two activities associated to it and a successor state transition, see C.2.3.
 - The first activity shall represent directly the time behaviour of the related Gantt bar and may be delayed in execution in order to start “synchronously” to the appropriate Gantt bar (see red line in Figure C.1). Hence, the activity shall have a definition for its earliest start point, see C.2.4. The naming convention for this action is: “DA_” + name of the Gantt bar.
 - The second activity shall be responsible for enabling the transition to “synchronously” deactivate the state according to the end of the Gantt bar. Hence, the activity shall contain a definition for the duration in order to delay the start of the activity and shall store it for synchronization reasons, see C.2.5. The naming convention for this activity is: “TA_” + name of the Gantt bar.
- For predecessor and successor relations in a Gantt chart the following provisions apply:
 - If a Gantt bar has two or more successor bars, it shall be connected to them by a simultaneous divergence, see C.2.6.
 - If a Gantt bar has two or more predecessors, they shall be connected to them by a simultaneous convergence. Between this simultaneous convergence and the predecessor states there shall be synchronization states with no activities and a successor transition with a “true” condition, see C.2.7.
 - If there are no explicit relations between Gantt bars, i. e. they are concurrent, the order of associated IML states shall be parallel and they shall have no ordering relation based on state-state transition sequences, see C.2.6 and C.2.7.
 - If predecessor/successor relations between Gantt bars are defined, the resulting structure shall be a sequence of IML states and state transitions, see C.2.6 and C.2.7.
 - As Gantt charts typically have a global time, while IML activities use local time starting at activation of the state, a conversion between the two time systems is needed. For bars with no predecessors the reference time point shall be the start point of the Gantt chart. For bars with predecessors, the reference time point shall be the end of the latest predecessor bar, see C.2.4.
- For the end of a Gantt chart the following provision applies:
 - Each Gantt chart shall have a unique termination state, see C.2.8.

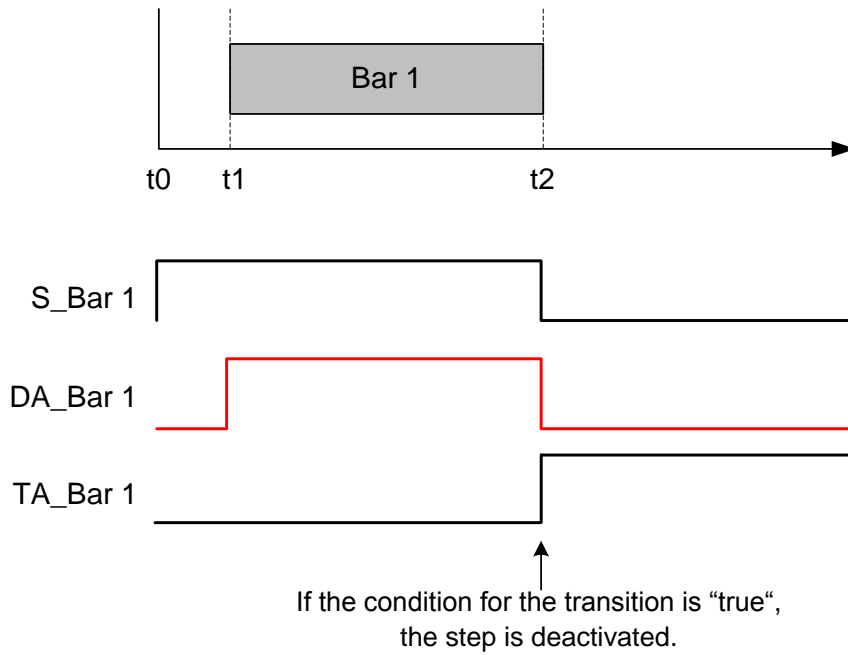


Figure C.1: Actions of IML for Gantt bar representation

C.2.2 Mapping of the start

Table C.1 specifies the mapping of properties and relations of the start of a Gantt chart to IML system elements.

IML system element	Graphical IML representation as SFC
General: state s with s.ID = <some ID> s.Name = "InitialStep" s.Init = true state transition st with st.ID = <some ID> st.Name = "InitialTransition" st.Guard.Formula = true st.Pre = s.ID addData ad with ad.ID = <some ID> ad.Type = ChartType ad.Value = "Gantt" ad.Pre = s.ID	
Case A: (one predecessor bar) Do nothing	

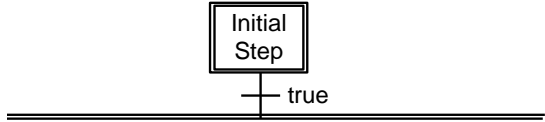
<p>Case B: (more than one predecessor bar) simultaneous divergence simDiv with simDiv.ID = <some ID> simDiv.Pre = st.ID</p>	
---	--

Table C.1: Mapping of the start of a Gantt chart to IML system elements

C.2.3 Mapping of bars

Table C.2 specifies the mapping of Gantt chart bars to IML system elements.

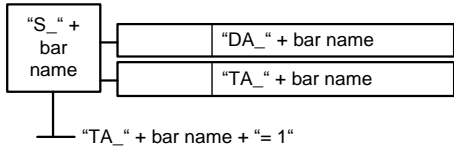
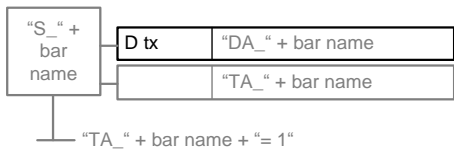
IML system element	Graphical IML representation as SFC
<p>General: state s with s.ID = <some ID> s.Name = "S_" + bar name s.Init = false activity a₁ with a₁.ID = <some ID> a₁.Name = "DA_" + bar name a₁.Pre = s.ID activity a₂ with a₂.ID = <some ID> a₂.Name = "TA_" + bar name a₂.Pre = s.ID state transition st with st.ID = <some ID> st.Name = "T_" + bar name st.Guard.Formula = a₂.name st.Pre = s.ID</p>	

Table C.2: Mapping of Gantt chart bars to IML system elements

C.2.4 Mapping of bar start points

Table C.3 specifies the mapping of a Gantt chart bar start point to IML system elements.

IML system element	Graphical IML representation as SFC
<p>Case A: (no predecessor bar) activity a₁ with a₁.Time.Start.Earliest = Bar.startpoint</p>	<p>Case A: tx=0</p> 
<p>Case B: (one predecessor bar)</p>	<p>Case B: tx=Bar.startpoint - predecessorBar.endpoint</p>

<p>activity a_1 with</p> $a_1.\text{Time.Start.Earliest} = \text{Bar.startpoint} - \text{predecessorBar.endpoint}$	
<p>Case C:</p> <p>(more than one predecessor bar)</p> <p>activity a_1 with</p> $a_1.\text{Time.Start.Earliest} = \text{Bar.startpoint} - \max(\text{predecessorBar.endpoint})$	<p>Case C: $\text{tx} = \text{Bar.startpoint} - \max(\text{predecessorBar.endpoint})$</p>

Table C.3: Mapping of a Gantt chart bar start point to IML system elements

C.2.5 Mapping of bar end points

Table C.4 specifies the mapping of a Gantt chart bar end point to IML system elements.

IML system element	Graphical IML representation as SFC
<p>Case A:</p> <p>(no predecessor bar)</p> <p>activity a_2 with</p> $a_2.\text{Time.Duration} = \text{Bar.endpoint}$	<p>Case A: $\text{ty} = \text{Bar.endpoint}$</p>
<p>Case B:</p> <p>(one predecessorBar)</p> <p>activity a_2 with</p> $a_2.\text{Time.Duration} = \text{Bar.endpoint} - \text{predecessorBar.endpoint}$	<p>Case B: $\text{ty} = \text{Bar.endpoint} - \text{predecessorBar.endpoint}$</p>
<p>Case C:</p> <p>(more than one predecessor bar)</p> <p>activity a_2 with</p> $a_2.\text{Time.Duration} = \text{Bar.endpoint} - \max(\text{predecessorBar.endpoint})$	<p>Case C: $\text{ty} = \text{Bar.endpoint} - \max(\text{predecessorBar.endpoint})$</p>

Table C.4: Mapping of a Gantt chart bar end point to IML system elements

C.2.6 Mapping of successor bars

Table C.5 specifies the mapping of Gantt chart bar successor activities to IML system elements.

IML system element	Graphical IML representation as SFC
<p>Case A:</p> <p>(no or only one successor bar)</p> <p>Do nothing</p>	<p>Case A:</p>

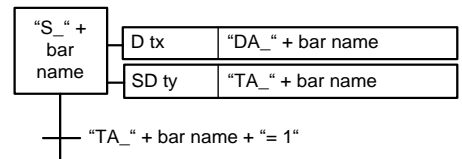
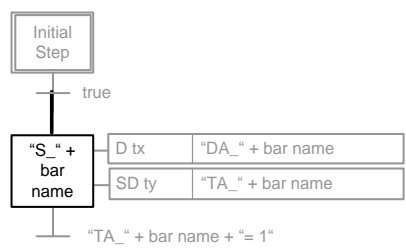
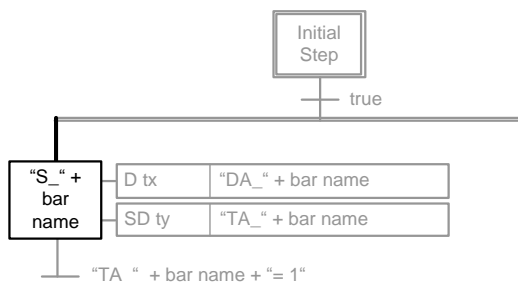
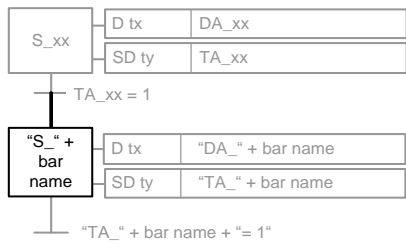
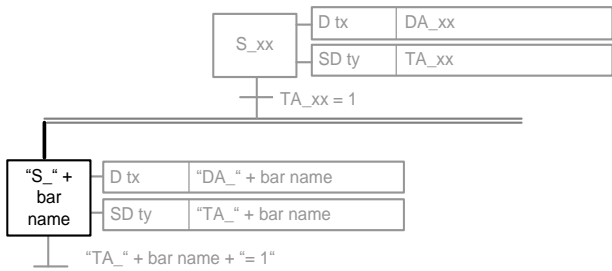
<p>Case B:</p> <p>(more than one successor bar)</p> <p>simultaneous divergence simDiv with</p> <p>simDiv.ID = <some ID></p> <p>simDiv.Pre = st.ID</p>	<p>Case B:</p> 
---	--

Table C.5: Mapping of Gantt chart successor bars to IML system elements

C.2.7 Mapping of predecessor bars

Table C.6 specifies the mapping of Gantt chart predecessor bars to IML system elements.

IML system element	Graphical IML representation as SFC
<p>Case A:</p> <p>(the only bar in the diagram with no predecessor bars)</p> <p>state s with</p> <p>s.Pre = st.ID of the initial state transition</p>	<p>Case A:</p> 
<p>Case B:</p> <p>(no predecessor bar and at least one other bar in the Gantt chart with no predecessor)</p> <p>state s with</p> <p>s.Pre = simDiv.ID of the initial simultaneous divergence</p>	<p>Case B:</p> 
<p>Case C:</p> <p>(only one predecessor bar and no other bar with the same predecessor bar)</p> <p>state s with</p> <p>s.Pre = st.ID of the predecessor state transition generated for the predecessor bar</p>	<p>Case C:</p> 
<p>Case D:</p> <p>(only one predecessor bar and this predecessor bar has more than one successor bar)</p> <p>state s with</p> <p>s.Pre = simDiv.ID of the predecessor simultaneous divergence generated for the predecessor bar</p>	<p>Case D:</p> 
Case E:	Case E:

(more than one predecessor bar, the predecessor bar has only one successor)

Synchronization state:

state s_i with

$s_i.ID = \langle \text{some ID} \rangle$

$s_i.Init = \text{false}$

$s_i.Name = \text{"SyS_"} + \text{bar name} + \text{"_"} + i$

$s_i.Pre = st.ID$ of the predecessor state transition generated for the predecessor bar

Note: For each predecessor step (if there is more than one) one synchronization step is created. They shall be numbered to ensure unique naming.

simultaneous convergence **simCon** with

$simCon.ID = \langle \text{some ID} \rangle$

$simCon.Pre = \{ \dots, s_i.ID \}$

state transition **st** with

$st.ID = \langle \text{some ID} \rangle$

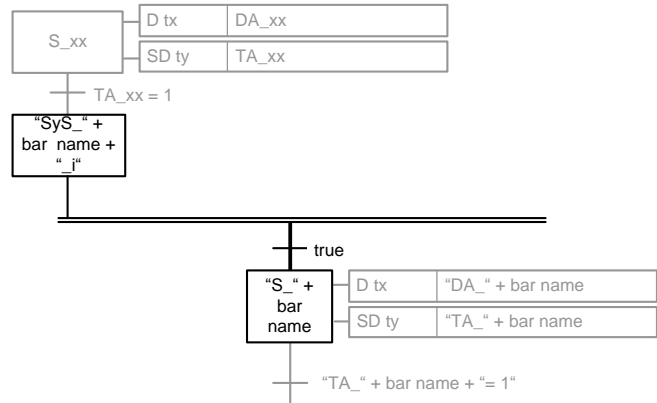
$st.Name = \text{"SyT_"} + \text{bar name}$

$st.Guard.Boolean = \text{true}$

$st.Pre = simCon.ID$

state s_{i+1} with

$s_{i+1}.Pre = st.ID$



Case F:

(more than one predecessor bar, the predecessor bar has more than one successor bar)

Synchronization state:

state s_i with

$s_i.ID = \langle \text{some ID} \rangle$

$s_i.Init = \text{false}$

$s_i.Name = \text{"SyS_"} + \text{bar name} + \text{"_"} + i$

$s_i.Pre = simDiv.ID$ of the predecessor simultaneous divergence generated for

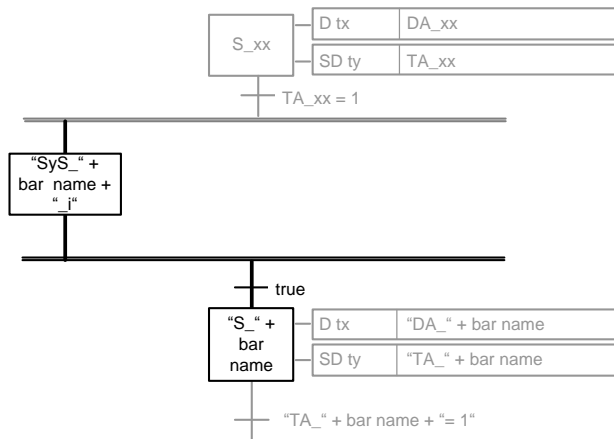
the predecessor bar

Note: For each predecessor state (if there is more than one) one synchronization state is created. They shall be numbered to ensure unique naming.

simultaneous convergence **simCon** with

$simCon.ID = \langle \text{some ID} \rangle$

Case F:

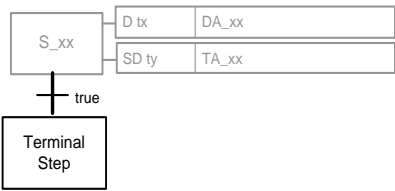
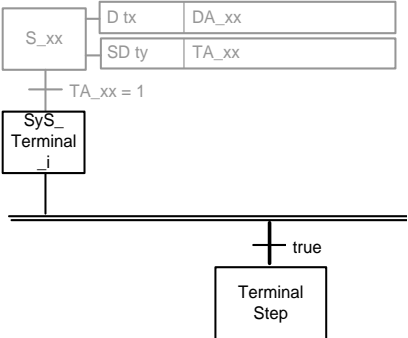


<p>simCon.Pre = { ..., s_i.ID }</p> <p>state transition st with</p> <p>st.ID = <some ID></p> <p>st.Name = "SyT_" + bar name</p> <p>st.Guard.Boolean = true</p> <p>st.Pre = simCon.ID</p> <p>state s_{i+1} with</p> <p>s_{i+1}.Pre = st.ID</p>	
---	--

Table C.6: Mapping of Gantt chart predecessor bars to IML system elements

C.2.8 Mapping of the end

Table C.7 specifies the mapping of the end of a Gantt chart to IML system elements.

IML system element	Graphical IML representation as SFC
<p>Case A:</p> <p>(One predecessor state transition for the terminal state)</p> <p>state transition st with</p> <p>st.ID = <some ID></p> <p>st.Name = "TerminalTransition"</p> <p>st.Guard.Boolean = true</p> <p>state s with</p> <p>s.ID = <some ID></p> <p>s.Init = false</p> <p>s.Name = "TerminalStep"</p> <p>s.Pre = st.ID</p>	<p>Case A:</p> 
<p>Case B:</p> <p>(One simultaneous convergence for all synchronisation states)</p> <p>Synchronization state:</p> <p>state s_i with</p> <p>s_i.ID = <some ID></p> <p>s_i.Init = false</p> <p>s_i.Name = "SyS_Terminal" + "_" + i</p> <p>s_i.Pre = st.ID of the predecessor state transition generated for the predecessor bar</p> <p><i>Note: For each state with no successor state (if there is more than one) one synchronization state is created. They shall be numbered to ensure unique naming. All synchronization states are</i></p>	<p>Case B:</p> 

predecessors of the final simultaneous convergence.

simultaneous convergence **simCon** with

simCon.ID = <some ID>

simCon.Pre = {..., s_i.ID}

state transition **st** with

st.ID = <some ID>

st.Name = "TerminalTransition"

st.Guard.Boolean = true

st.Pre = simCon.ID

Terminal State:

state **s_{i+1}** with

s_{i+1}.ID = <some ID>

s_{i+1}.Init = false

s_{i+1}.Name = "TerminalStep"

s_{i+1}.Pre = st.ID

Table C.7: Mapping of the end of a Gantt chart to IML system elements

C.3 Mapping of activity-on-node networks to IML

C.3.1 Common rules

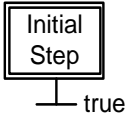
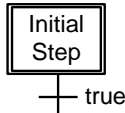
For mapping an activity-on-node network to an IML system, the following provisions apply:

- Each activity-on-node network shall be mapped to an IML system.
- For the start of an activity-on-node network the following provision applies:
 - For each IML system representing an activity-on-node network, an initial IML system element "State" (named "InitialStep") shall be created, followed by an IML system element "State Transition" with condition "true" as link to all further elements, see C.3.2.
- For nodes in an activity-on-node network the following provisions apply:
 - Each node of the activity-on-node network shall be represented by the IML system element "State" with two IML system elements "Activity" associated to it and a successor IML system element "State Transition", see C.3.3 and C.3.4.
 - The first IML system element "Activity" shall represent directly the timing behaviour of the associated node. Hence, the IML system element "Activity" shall have a definition for its earliest start point, see C.3.4. The naming convention is: "DA_" + name of the node.
 - The second IML system element "Activity" shall be responsible for enabling the IML system element "State Transition" to synchronously deactivate the IML system element "State" according to the end of the node. Hence, the IML system element "Activity" shall contain a definition for the duration in order to delay the start of the activity and to store it for synchronization reasons, see C.3.5. The naming convention is: "TA_" + name of the node.
 - The subsequent IML system element "State Transition" shall have the following condition: Name of the second IML system element "Activity" + "= true".

- For predecessor and successor relations in an activity-on-node network the following provisions apply:
 - If an activity-on-node network node has two or more successor nodes, it shall be connected to them by a simultaneous divergence, see C.3.6.
 - If an activity-on-node network node has two or more predecessor nodes, it shall be connected to them by a simultaneous convergence. Between this simultaneous convergence and the predecessor states there shall be synchronization states with no activities and a successor transition with a “true” condition, see C.3.7.
 - If there are no explicit relations between activity-on-node network nodes, i. e. they are concurrent, the associated IML states shall be parallel and they shall have no ordering relation based on state-state transition sequences, see C.3.6 and C.3.7.
 - If predecessor/successor relations between activity-on-node network nodes are defined, the resulting structure shall be a sequence of IML states and state transitions, see C.3.6 and C.3.7.
 - Within activity-on-node networks, various timing conditions are used. They shall be stored in AutomationML specific additional data related to the first activity (“DA_” + name of the activity-on-node network node) according to the additional data structure defined by PLCOpen XML 2.0.
 - activity-on-node networks use global time values. Hence, for timing values of IML activities these timing values shall be converted to local relative time related to the activation time of the current state.
- For the end of an activity-on-node network the following provision applies:
 - Each activity-on-node network shall have a unique termination state, see C.3.8.

C.3.2 Mapping of the start of an activity-on-node network

Table C.8 specifies the mapping of the start of an activity-on-node network to IML system elements.

IML element	Graphical IML representation as SFC
General: state s with s.ID = <some ID> s.Name = “InitialStep” s.Init = true state transition st with st.ID = <some ID> st.Name = “InitialTransition” st.Guard.Formula = true st.Pre = s.ID addData ad with ad.ID = <some ID> ad.Type = ChartType ad.Value = “ActivityOnNodeNetwork” ad.Pre = s.ID	 <pre> graph TD S[Initial Step] -- true --> T[] </pre>
Case A: <i>(one node with no predecessor nodes)</i> <i>Do nothing</i>	 <pre> graph TD S[Initial Step] -- true --> T[] </pre>

<p>Case B:</p> <p>(more than one node with no predecessor nodes in the diagram additionally)</p> <p>simultaneous divergence simDiv with</p> <p>simDiv.ID = <some ID></p> <p>simDiv.Pre = st.ID</p>	
--	--

Table C.8: Mapping of the start of an activity-on-node network to IML system elements

C.3.3 Mapping of nodes

Table C.9 specifies the mapping of activity-on-node network nodes to IML system elements.

IML element	Graphical IML representation as SFC
<p>General:</p> <p>state s with</p> <p>s.ID = <some ID></p> <p>s.Name = "S_" + node name</p> <p>s.Init = false</p> <p>activity a₁ with</p> <p>a₁.ID = <some ID></p> <p>a₁.Name = "DA_" + node name</p> <p>a₁.Pre = s.ID</p> <p>activity a₂ with</p> <p>a₂.ID = <some ID></p> <p>a₂.Name = "TA_" + node name</p> <p>a₂.Pre = s.ID</p> <p>state transition st with</p> <p>st.ID = <some ID></p> <p>st.Name = "T_" + node name</p> <p>st.Guard.Formula = "a₂.name"</p> <p>st.Pre = s.ID</p>	

Table C.9: Mapping of activity-on-node network nodes to IML system elements

C.3.4 Mapping of node start points

Table C.10 specifies the mapping of activity-on-node network node start points to IML system elements.

IML element	Graphical IML representation as SFC
General: activity a_1 with $a_1.\text{Time.Start.Earliest} = \text{Node.earlieststart}$ $a_1.\text{Time.start.latest} = \text{Node.lateststart}$	

Table C.10: Mapping of activity-on-node network node start points to IML system elements

C.3.5 Mapping of node end points

Table C.11 specifies the mapping of activity-on-node network node end points to IML system elements.

IML element	Graphical IML representation as SFC
General: activity a_2 with $a_2.\text{Time.Duration} = \text{Node.duration}$ $a_2.\text{Time.end.earliest} = \text{Node.earliestend}$ $a_2.\text{Time.end.latest} = \text{Node.latestend}$ $a_2.\text{Time.delay} = \text{Node.delay}$	

Table C.11: Mapping of activity-on-node network node end points to IML system elements

C.3.6 Mapping of successor nodes

Table C.12 specifies the mapping of activity-on-node network successor activities to IML system elements.

IML element	Graphical IML representation as SFC
Case A: (no or only one successor node) Do nothing	Case A:
Case B: (more than one successor node) simultaneous divergence simDiv with $\text{simDiv.ID} = \langle \text{some ID} \rangle$ $\text{simDiv.Pre} = \text{st.ID}$	Case B:

Table C.12: Mapping of activity-on-node network successor nodes to IML system elements

C.3.7 Mapping of predecessor nodes

Table C.13 specifies the mapping of activity-on-node network predecessor nodes to IML system elements.

IML element	Graphical IML representation as SFC
<p>Case A:</p> <p>(the only node in the diagram with no predecessor node)</p> <p>state s with</p> <p>s.Pre = st.ID of the initial state transition</p>	<p>Case A:</p>
<p>Case B:</p> <p>(no predecessor node and at least one other node in the activity-on-node network with no predecessor)</p> <p>state s with</p> <p>s.Pre = simDiv.ID of the initial simultaneous divergence</p>	<p>Case B:</p>
<p>Case C:</p> <p>(only one predecessor node and no other activity with the same predecessor node)</p> <p>state s with</p> <p>s.Pre = st.ID of the predecessor state transition generated for the predecessor node</p>	<p>Case C:</p>
<p>Case D:</p> <p>(only one predecessor node and this predecessor node has more than one successor nodes)</p> <p>state s with</p> <p>s.Pre = simDiv.ID of the predecessor simultaneous divergence generated for the predecessor node</p>	<p>Case D:</p>
<p>Case E:</p> <p>(more than one predecessor node, the predecessor node has only one successor node)</p> <p>state s_i with</p> <p>s_i.ID = <some ID></p>	<p>Case E:</p>

s_i .Init = false

s_i .Name = "SyS_" + node name + "_" + i

s_i .Pre = st.ID of the predecessor state transition generated for the predecessor mode

Note: If more than one predecessor state exists, one synchronization state for each predecessor state shall be created. They shall be numbered to ensure a unique naming.

simultaneous convergence **simCon** with

simCon.ID = <some ID>

simCon.Pre = { ..., s_i .ID }

state transition **st** with

st.ID = <some ID>

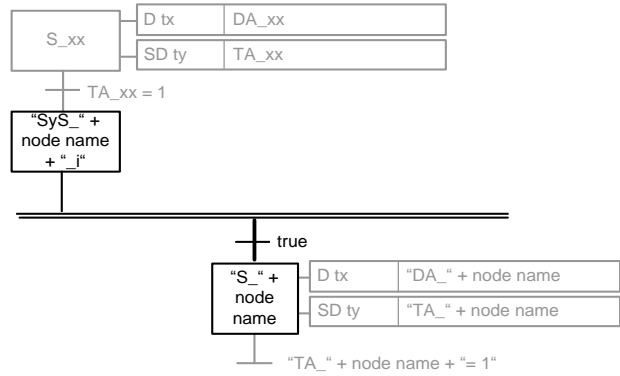
st.Name = "SyT_" + node name

st.Guard.Boolean = true

st.Pre = simCon.ID

state s_{i+1} with

s_{i+1} .Pre = st.ID



Case F:

(more than one predecessor node, the predecessor node has more than one succeeding nodes)

state s_i with

s_i .ID = <some ID>

s_i .Init = false

s_i .Name = "SyS_" + node name + "_" + i

s_i .Pre = simDiv.ID of the predecessor simultaneous divergence generated for the predecessor node

Note: If more than one predecessor state exists, one synchronization state for each predecessor state shall be created. They shall be numbered to ensure a unique naming.

simultaneous convergence **simCon** with

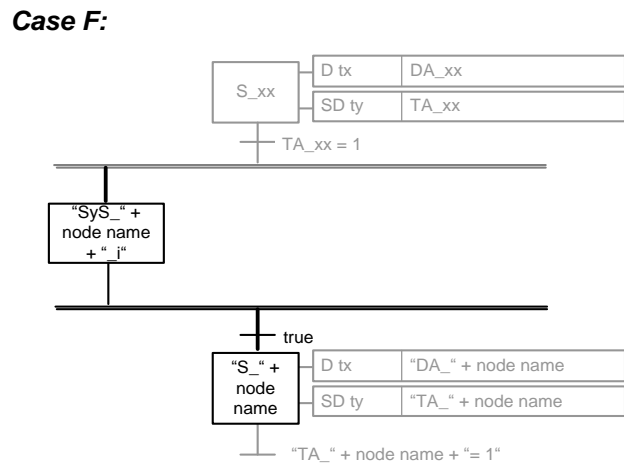
simCon.ID = <some ID>

simCon.Pre = { ..., s_i .ID }

state transition **st** with

st.ID = <some ID>

st.Name = "SyT_" + node name

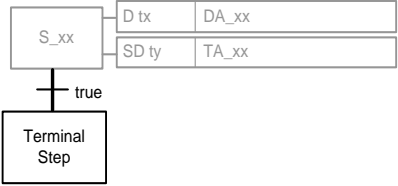
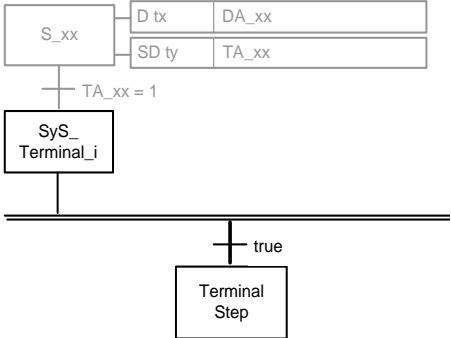


<pre> st.Guard.Boolean = true st.Pre = simCon.ID state s_{i+1} with s_{i+1}.Pre = st.ID </pre>	
--	--

Table C.13: Mapping of activity-on-node network predecessor nodes to IML system elements

C.3.8 Mapping of the end of an activity-on-node network

Table C.14 specifies the mapping of the end of an activity-on-node network to IML system elements.

IML element	Graphical IML representation as SFC
<p>Case A:</p> <p>(One predecessor state transition for the terminal state)</p> <p>state transition st with</p> <pre> st.ID = <some ID> st.Name = "TerminalTransition" st.Guard.Boolean = true </pre> <p>state s with</p> <pre> s.ID = <some ID> s.Init = false s.Name = "TerminalStep" s.Pre = st.ID </pre>	<p>Case A:</p> 
<p>Case B:</p> <p>(One simultaneous convergence for all synchronisation states)</p> <p>Synchronization state:</p> <p>state s_i with</p> <pre> s_i.ID = <some ID> s_i.Init = false s_i.Name = "SyS_Terminal" + " " + i s_i.Pre = st.ID of the predecessor state transition generated for the predecessor bar </pre> <p><i>Note:</i> For each state with no successor state (if there is more than one) one synchronization state is created. They shall be numbered to ensure unique naming. All synchronization states are predecessors of the final simultaneous convergence.</p> <p>simultaneous convergence simCon with</p> <pre> simCon.ID = <some ID> simCon.Pre = { ..., s_i.ID } </pre>	<p>Case B:</p> 

state transition st with st.ID = <some ID> st.Name = "TerminalTransition" st.Guard.Boolean = true st.Pre = simCon.ID <i>Terminal State:</i> state s_{i+1} with s _{i+1} .ID = <some ID> s _{i+1} .Init = false s _{i+1} .Name = "TerminalStep" s _{i+1} .Pre = st.ID	
--	--

Table C.14: Mapping of the end of an activity-on-node network to IML system elements

C.4 Mapping of timing diagrams to IML

C.4.1 Common rules

For mapping a timing diagram to an IML system, the following provisions apply:

- Each timing diagram shall be mapped to an IML system.
- For the startof a timing diagram the following provision applies:
 - For each IML representation of a timing diagram, an initial state (named "InitialStep") shall be created, followed by a state transition with condition "true" and an initial simultaneous divergence as link to all further elements, see C.4.2.
- For the timeline in a timing diagram the following provision applies:
 - The timeline within a timing diagram shall be represented by one parallel branch of a simultaneous divergence in the IML system, see C.4.3.
- For resources in a timing diagram the following provisions apply:
 - Each resource within a timing diagram shall be represented by a parallel branch of a simultaneous divergencein the IML system, describing the corresponding resource state flow, see C.4.4.
- For resource statesin a timing diagram the following provisions apply (see C.4.5):
- Each resource state flow shall be represented by a sequence of states and their state transitions in the associated branch of the IML system.
- Each resource state and each resource state change shall be represented by an activity and shall be associated to a SFC state within the corresponding branch.
- The current resource state or resource state change shall be represented by one activity attached to the active state in the corresponding IML system branch.
- Each resource state change shall be triggered by a signal.

- Each resource state change shall have a duration which is defined in the corresponding activity. The duration shall be equal to or greater than zero. The end of a resource state change leads to firing a signal which shall be used only as condition to enter the subsequent resource state or resource state change.
- Predecessor/successor relations between resource states and resource state changes of one resource shall be represented by sequences of states and state transitions.
- For signals in a timing diagram the following provisions apply (see C.4.6):
 - Signals shall be represented by the integration of Boolean values within state transition guards.
 - Firing of signals resulting from a resource state change shall be represented by an activity, associated to the corresponding state of the resource.
 - Each firing of a signal from the timeline (time signal) shall be described by a state and one attached activity within the timeline.

Note: Signals can be fired by the timeline at any time or by a resource after remaining within a resource state with a predefined duration.

- For the end of a timing diagram the following provision applies:
 - Each representation of a timing diagram shall contain a terminal simultaneous convergence, a terminal state transition, and a terminal step as end frame, see C.4.7. The condition for the terminal state transition is the firing of an external end signal.

C.4.2 Mapping of the start of a timing diagram

Table C.15 specifies the mapping of the start of a timing diagram to IML system elements.

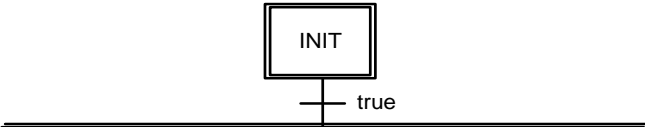
IML element	Graphical IML representation as SFC
<p>General:</p> <p>state s with</p> <p style="padding-left: 20px;">s.ID = <some ID></p> <p style="padding-left: 20px;">s.Name = "INIT"</p> <p style="padding-left: 20px;">s.Init = true</p> <p>state transition st with</p> <p style="padding-left: 20px;">st.ID = <some ID></p> <p style="padding-left: 20px;">st.Name = "InitialTransition"</p> <p style="padding-left: 20px;">st.Guard.Formula = true</p> <p style="padding-left: 20px;">st.Pre = s.ID</p> <p>addData ad with</p> <p style="padding-left: 20px;">ad.ID = <some ID></p> <p style="padding-left: 20px;">ad.Type = ChartType</p> <p style="padding-left: 20px;">ad.Value = "TimingDiagram"</p> <p style="padding-left: 20px;">ad.Pre = s.ID</p> <p>simultaneous divergence simDiv with</p> <p style="padding-left: 20px;">simDiv.ID = <some ID></p> <p style="padding-left: 20px;">simDiv.Pre = st.ID</p>	 <p>The diagram shows a state box labeled 'INIT' with a vertical line extending downwards to a horizontal line. A small vertical tick mark is on the horizontal line, and the word 'true' is written to the right of the tick mark, indicating a self-loop guard.</p>

Table C.15: Mapping of the start of a timing diagram to IML system elements

C.4.3 Mapping of the timeline

Table C.16 specifies the mapping of the timeline to IML system elements.

IML element	Graphical IML representation as SFC
<p>General:</p> <p>state s with</p> <ul style="list-style-type: none"> s.ID = <some ID> s.Name = "Time_Step_0" s.Init = false <p>s.Pre = simDiv.ID of the simultaneous divergence resulting from the start of the timing diagram</p> <p>activity a with</p> <ul style="list-style-type: none"> a.ID = <some ID> a.Name = "Signal_Time_0" a.Pre = s.ID a.Time.Delay = 0 <p>state transition st with</p> <ul style="list-style-type: none"> st.ID = <some ID> st.Name = "T_" + a.Name st.Guard.Boolean = a.Name st.Pre = s.ID <p>For each additional fired external signal:</p> <p>state s_x with</p> <ul style="list-style-type: none"> s_x.ID = <some ID> s_x.Name = "Time_Step_" + sequence number s_x.Init = false s_x.Pre = st_x.ID of predecessor state transition within timeline 	<pre> graph TD INIT[INIT] -- true --> TS0[Time_Step_0] TS0 -- "[Signal_Time_0 = true]" --> TSseq["Time_Step_ + sequence number"] TSseq -- "["Signal_Time_ + sequence number = true]" --> End[...] style TS0 fill:#fff,stroke:#000 style TSseq fill:#fff,stroke:#000 style End fill:none,stroke:none </pre>

Table C.16: Mapping of the timeline to IML system elements

C.4.4 Mapping of resources

Table C.17 specifies the mapping of timing diagram resources to IML system elements.

IML element	Graphical IML representation as SFC
<p>General:</p> <p>For each resource one parallel branch is created in the SFC. This branch contains:</p> <p>state s with</p> <ul style="list-style-type: none"> s.ID = <some ID> s.Name = resource name + "_0" s.Init = False 	<pre> graph TD INIT[INIT] -- true --> RS["resource name + \"_0\""] RS -- "resource name + \"_\" + initial resource state name" --> End[...] style RS fill:#fff,stroke:#000 style End fill:none,stroke:none </pre>

<p>s.Pre = simDiv.ID</p> <p>activity a with</p> <p>a.ID = <some ID></p> <p>a.Name = resource name + “_” initial resource state name</p> <p>a.Pre = s.ID</p> <p>a.Time.Delay = 0</p> <p><i>Note:</i>The sequence of values for the resource is represented by the resource state flow.</p>	
--	--

Table C.17: Mapping of timing diagram resources to IML system elements

C.4.5 Mapping of resource states

Table C.18 specifies the mapping of timing diagram resource states to IML system elements.

IML element	Graphical IML representation as SFC
<p>Case A:</p> <p>(remain within one resource state [active state])</p> <p>For each active resource state one state shall be created:</p> <p>state s with</p> <p>s.ID = <some ID></p> <p>s.Name = resource name + “_” + sequence number</p> <p>s.Init = false</p> <p>s.Pre = st.ID of the predecessor state transition within the resource state flow</p> <p>activity a with</p> <p>a.ID = <some ID></p> <p>a.Name = resource name + “_” + resource state name</p> <p>a.Time.Delay = 0</p> <p>a.Pre = s.ID of associated state within the resource state flow</p> <p><i>Note 1:</i>If the resource state is never entered within the resource state flow, the action is defined in the declarative part of the PLCopen XML document only.</p> <p><i>Note 2:</i>Multiple activations of resource states are possible and lead to a new state each time.</p> <p><i>Note 3:</i> To each state at least one activity is associated, setting the active resource state. The following state transition waits for a signal to trigger the succeeding</p>	

resource state change. The active resource state will automatically be set to "false" when leaving the step. When leaving the state after a predefined duration, a second activity is needed.

Case B:

(resource state change)

Each resource state change leads to one state:

state **s** with

s.ID = <some ID>

s.Name = resource name + "_" + sequence number

s.Init = false

s.Pre = st.ID of the predecessor state transition within the resource state flow

activity **a** with

a.ID = <some ID>

a.Name = resource name + "_" + origin resource state name + "_to_" + target resource state name

a.Time.Delay = 0

a.Pre = s.ID of associated state within the resource state flow

Note 1: If a predefined resource state change is never executed within the resource state flow the action is defined in the declarative part of the PLCOpen XML document only.

Note 2: Multiple activations, even with different durations, are possible and lead to a new state each time.

Note 3: To each state two activities are associated: one activity to set the active resource state change and one activity to fire a signal at the end of the resource state change. A state transition activates the succeeding state. The condition for this state transition shall only be the boolean signal of the resource state change.

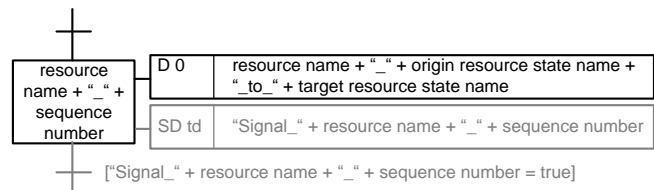


Table C.18: Mapping of timing diagram resource states to IML system elements

C.4.6 Mapping of signals

Table C.19 specifies the mapping of timing diagram signals to IML system elements.

IML element	Graphical IML representation as SFC
<p>Time signals:</p> <p>activity a with</p> <p>a.ID = <some ID></p> <p>a.Name = "Signal_Time" + sequence number</p> <p>a.Time.Duration = signal time delay</p> <p>a.Pre = s.ID of actual state within the timeline</p> <p><i>Note: Activity a belongs to a state in the timeline.</i></p> <p>state transition st₁ with</p> <p>st₁.ID = <some ID></p> <p>st₁.Name = "T_Signal_Time" + sequence number</p> <p>st₁.Guard.Boolean = a.Name</p> <p>st₁.Pre = s.ID of the state, which associates to a.ID</p> <p>state transition st₂ with</p> <p>st₂.ID = <some ID></p> <p>st₂.Name = "T_Signal_Time_" + sequence number</p> <p>st₂.Guard.Boolean = a.Name</p> <p>st₂.Pre = s.ID of active state within the resource state flow where a change is triggered by the signal</p>	
<p>Signals between two resource state flows:</p> <p>activity a with</p> <p>a.ID = <some ID></p> <p>a.Name = "Signal_" + resource name + "_" + sequence number</p> <p>a.Time.Duration = time delay</p> <p>a.Pre = s.ID of actual state within the resource state flow</p> <p>state transition st₁ with</p> <p>st₁.ID = <some ID></p> <p>st₁.Name = "T_" + resource name + "_" + sequence number</p> <p>st₁.Guard.Boolean = a.Name</p> <p>st₁.Pre = s.ID of the state, which</p>	

associates to a.ID

state transition **st₂** with

st₂.ID = <some ID>

st₂.Name = "T_" + resource name +
"_" + sequence number

st₂.Guard.Boolean = a.Name

st₂.Pre = s.ID of active state within the
resource state flow where a change is
triggered by the signal

*Note: If a change within a resource state
flow depends on more than one signal,
these signals can be combined with a
boolean expression.*

Signal within one resource state flow:

activity **a** with

a.ID = <some ID>

a.Name = "Signal_" + resource name + "_" +
sequence number

a.Time.Duration = time delay

a.Pre = s.ID of the actual state within the
resource state flow

state transition **st** with

st.ID = <some ID>

st.Name = "T_Signal_" + resource name +
"_" + sequence number

st.Guard.Boolean = a.Name

st.Pre = s.ID of the state associated to
a.ID

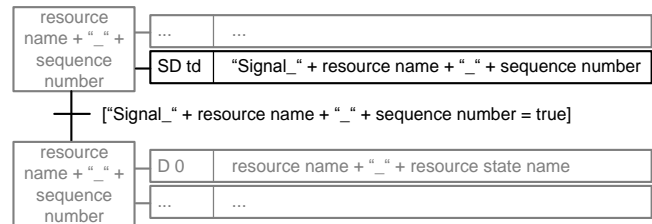


Table C.19: Mapping of timing diagram signals to IML system elements

C.4.7 Mapping of the end of a timing diagram

Table C.20 specifies the mapping of the end of a timing diagram to IML system elements.

IML element	Graphical IML representation as SFC
<p>General:</p> <p><i>For the termination of the complete timing diagram, all resource state flows and the timeline are synchronized by an end time signal and the following resulting elements:</i></p> <p><i>Timeline end state:</i></p> <p>state s₁ with</p> <ul style="list-style-type: none"> s₁.ID = <some ID> s₁.Init = false s₁.Name = "Time_Step_END" <p>s₁.Pre = st.ID of last state transition within timeline</p> <p>activity a with</p> <ul style="list-style-type: none"> a.ID = <some ID> a.Name = "Signal_Time_END" a.Time.Duration = signal time delay a.Pre = s₁.ID <p>simultaneous convergence simCon with</p> <ul style="list-style-type: none"> simCon.ID = <some ID> <p>simCon.Pre = [a.ID; ID of the last states in each resource state flow]</p> <p>state transition st with</p> <ul style="list-style-type: none"> st.ID = <some ID> st.Name = "TerminalTransition" <p>st.Guard.Boolean = [Signal_Time_END = true]</p> <p>st.Pre = simCon.ID</p> <p><i>Terminal state:</i></p> <p>state s₂ with</p> <ul style="list-style-type: none"> s₂.ID = <some ID> s₂.Init = false s₂.Terminal = true s₂.Name = "TerminalStep" s₂.Pre = st.ID 	<pre> graph TD TS_n[Time_Step_n] -- "[...]" --> TS_END[Time_Step_END] TS_END -- "SD std" --> SFE[Signal_Time_END] TS_END -- "[Signal_Time_END = true]" --> TS[Terminal Step] </pre>

Table C.20: Mapping of the end of a timing diagram to IML system elements

C.4.8 Mapping of timing diagram details

Table C.21 specifies the mapping of further information of a timing diagram to IML system elements.

IML element	Graphical IML representation as SFC
<p>Name of groups:</p> <p>Additional Data ad with</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramResourceGroup</p> <p>ad.value = name of the group the element belongs to</p> <p>ad.Pre = s.ID of the first step within the parallel branch belonging to the resource</p>	<p>Group = some_group/some_subgroup</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramResourceGroup</p> <p>ad.value = name of the group the element belongs to</p> <p>ad.Pre = s.ID of the first step within the parallel branch belonging to the resource</p>
<p>Names of resource state changes:</p> <p>Additional Data ad with</p> <p>ad.ID = <some ID></p> <p>ad.type = ResourceStateChangeDefinition.DefinitionName</p> <p>ad.value = name of the resource change</p> <p>ad.Pre = a.ID of associated action</p>	<p>Name = some name</p> <p>ad.ID = <some ID></p> <p>ad.type = ResourceStateChangeDefinition.DefinitionName</p> <p>ad.value = name of the resource change</p> <p>ad.Pre = a.ID of associated action</p>
<p>Durations of resource states and resource state changes:</p> <p>Additional Data ad with</p> <p>ad.ID = <some ID></p> <p>ad.type = ResourceStateChangeDefinition.Duration</p> <p>ad.value = name of the group the element belongs to</p> <p>ad.Pre = a.ID of associated action</p>	<p>Duration = some duration</p> <p>ad.ID = <some ID></p> <p>ad.type = ResourceStateChangeDefinition.Duration</p> <p>ad.value = name of the group the element belongs to</p> <p>ad.Pre = a.ID of associated action</p>
<p>Names of signal inputs associated to resource states:</p> <p>Additional Data ad with</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramPLCVariable.Name</p> <p>ad.value = name of the variable associated to the input</p> <p>ad.Pre = a.ID of associated action</p>	<p>PLCopenVariable = variable name</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramPLCVariable.Name</p> <p>ad.value = name of the variable associated to the input</p> <p>ad.Pre = a.ID of associated action</p>
<p>Names of actuator outputs associated to resource states:</p> <p>Additional Data ad with</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramPLCVariable.Name</p> <p>ad.value = name of the variable</p>	<p>PLCopenVariable = variable name</p> <p>ad.ID = <some ID></p> <p>ad.type = TimingDiagramPLCVariable.Name</p> <p>ad.value = name of the variable associated to the output</p> <p>ad.Pre = a.ID of associated action</p>

associated to the output	
ad.Pre = a.ID of associated action	

Table C.21: Mapping of timing diagram details to IML system elements

C.5 Mapping of state charts

C.5.1 Common rules

For mapping a state chart to an IML system, the following provisions apply:

- Each state chart shall be mapped to an IML system.
- Each region of a state chart shall be represented in an IML system by a header, see C.5.2.
- For states in a state chart the following provision applies:
 - Each state shall be represented in an IML system by a state, see C.5.3.
- For state transitions in a state chart the following provisions apply:
 - Each state transition shall be represented in an IML system by a state transition.
 - If an action is executed during a state transition, the state transition shall be represented in an IML system by a state transition, followed by a state with the action associated to it, followed by a state transition with condition “true” as link to all further elements, see C.5.9.
- For history connectors in a state chart the following provision applies:
 - Each history connector of a state chart shall be represented in an IML system by a state and an IML additional data element, see C.5.10.
- For actions in a state chart the following provision applies:
 - Each action shall be represented in an IML system by an action, which shall be assigned to a state, see C.5.5.
- For events in a state chart the following provision applies:
 - Each event shall be represented in an IML system by an event and additional IML attribute values, see C.5.7.
- For condition connectors the following provision applies:
 - Each condition connector of a state chart shall be represented in an IML system by a state and an IML additional data element, see Annex C.
- For signals in a state chart the following provision applies:
 - Each signal shall be represented in an IML system by a variable, see C.5.8.

C.5.2 Mapping of regions

Table C.22 specifies the definition of state chart regions.

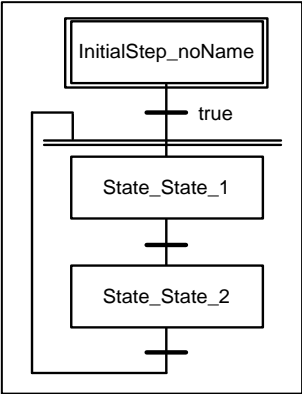
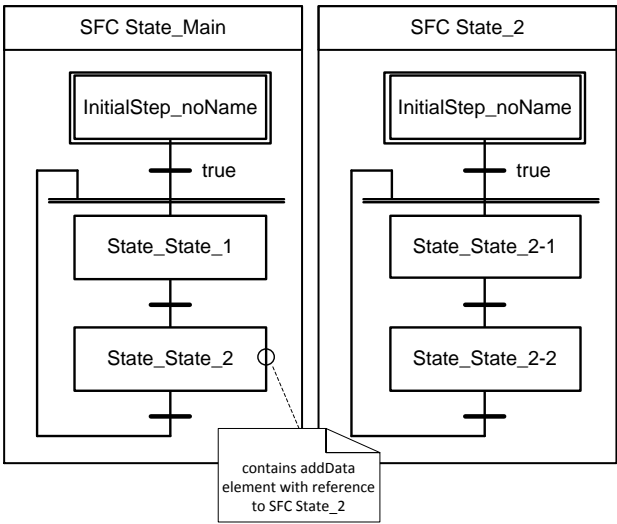
IML element	Example for a graphical IML representation as SFC
<p>Case A: (flat state chart)</p> <p>header h with</p> <ul style="list-style-type: none"> h.ID = <some ID> h.Name = name of the state chart h.Members = IDs of all entities resulting from the transformation of the state chart members 	<p>Case A:</p> 
<p>Case B: (state chart with hierarchies)</p> <p>A state chart with more than one hierarchy level shall result in one IML system with its IML header h for each sub state chart:</p> <p>header h with</p> <ul style="list-style-type: none"> h.ID = <some ID> h.Name = name of the state chart h.Members = IDs of all entities resulting from the transformation of the state chart members of the corresponding sub state chart <p>The relation between a state <i>s</i> and its internal sub state charts is represented by an additional data element attached to the state:</p> <p>addData ad with</p> <ul style="list-style-type: none"> ad.ID = <some ID> ad.Type = StateChartSubCharts/POURef ad.Value = reference to the IML system representing the sub state chart <p><u>Note:</u> In PLCopen XML the URI of the POU</p> <p>ad.Pre = s.ID</p>	<p>Case B:</p> 

Table C.22: Definition of state chart headers

C.5.3 Mapping of states

Table C.23 specifies the mapping of state chart states to IML system elements.

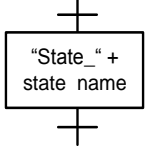
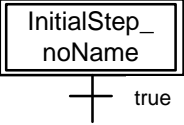
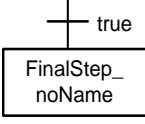
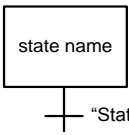
IML element	Graphical IML representation as SFC
General: state s with s.ID = <some ID> s.Name = "State_" + state name s.Init = false s.Terminal = false	General: 
Initial state: state s with s.ID = <some ID> s.Name = "InitialStep_noName" s.Init = true s.Terminal = false	Initial state: 
Terminal state: state transition st with st.ID = <some ID> st.Name = "StateTransition_" + state transition name st.Guard.Boolean = true state s with s.ID = <some ID> s.Init = false s.Terminal = true s.Name = "FinalStep_noName" s.Pre = st.ID	Terminal state: 

Table C.23: Mapping of state chart states to IML system elements

C.5.4 Mapping of successors of states

Table C.12 specifies the mapping of state chart successor activities to IML system elements.

IML element	Graphical IML representation as SFC
Case A: (no or only one successor) Do nothing	Case A: 
Case B: (more than one successor) simultaneous divergence simDiv with	Case B:

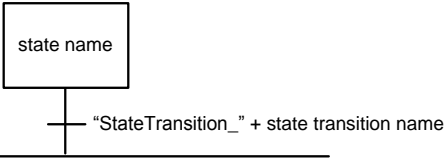
simDiv.ID = <some ID> simDiv.Pre = st.ID	
---	--

Table C.24: Mapping of state chart state successors to IML system elements

C.5.5 Mapping of predecessor states

Table C.13 specifies the mapping of state chart predecessor states to IML system elements.


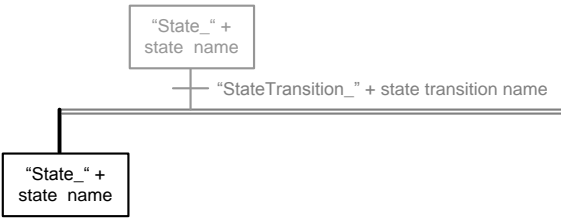
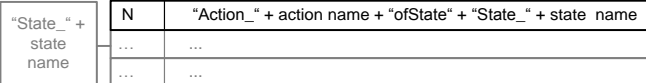
IML Element	Graphical IML representation as SFC
Case A: <i>(only one predecessor state and no other state with the same predecessor state)</i> state s with s.Pre = st.ID of the predecessor state transition, generated for the predecessor state	Case A: 
Case B: <i>(only one predecessor state and this predecessor state has more than one successor states)</i> state s with s.Pre = simDiv.ID of the predecessor simultaneous divergence generated for the predecessor state	Case B: 

Table C.25: Mapping of state chart predecessor states to IML system elements

C.5.6 Mapping of actions

Table C.26 specifies the mapping of state chart actions to IML system elements.

IML element	Graphical IML representation as SFC
Entry action: action a with a.ID = <some ID> a.Name = "Action_" + action name + "_ofState_" + s.Name for the state the entry action belongs to a.Formula = content of action a a.FiredEvents = set of events fired by a a.Pre = s.ID for the state the entry action belongs to addData ad with ad.ID = <some ID> ad.Type = StateChartActionType ad.Value = entryAction	Entry action: 

<div>ad.Pre = s.ID</div> <div>Action within a state; do action:</div> <div>action a with</div> <div> a.ID = <some ID></div> <div>a.Name = "Action_" + action name + "ofState_" + s.Name for the state s the do action belongs to</div> <div> a.Formula = content of action a</div> <div> a.FiredEvents = set of events fired by a</div> <div>a.Pre = s.ID for the state s the do action belongs to</div> <div>addData ad with</div> <div> ad.ID = <some ID></div> <div> ad.Type = StateChartActionType</div> <div> ad.Value = doAction</div> <div> ad.Pre = s.ID</div>	<div>Action within a state; do action:</div> <div><div><div>"State_" + state name</div><div><div>...</div><div>N</div><div>...</div></div><div><div>...</div><div>"Action_" + action name + "ofState" + "State_" + state name</div><div>...</div></div></div></div>
<div>Exit action:</div> <div>action a with</div> <div> a.ID = <some ID></div> <div>a.Name = "Action_" + action name + "ofState_" + s.Name for the state the exit action belongs to</div> <div> a.Formula = Content of action</div> <div> a.FiredEvents = set of events fired by a</div> <div>a.Pre = s.ID for the state the entry action belongs to</div> <div>addData ad with</div> <div> ad.ID = <some ID></div> <div> ad.Type = StateChartActionType</div> <div> ad.Value = exitAction</div> <div> ad.Pre = s.ID</div>	<div>Exit action:</div> <div><div><div>"State_" + state name</div><div><div>...</div><div>...</div><div>N</div></div><div><div>...</div><div>...</div><div>"Action_" + action name + "ofState" + "State_" + state name</div></div></div></div>

Table C.26: Mapping of state chart actions to IML system elements

C.5.7 Mapping of events

Table C.19 specifies the mapping of state chart events to IML system elements.

IML element	Graphical IML representation as SFC
General: event ev with ev.ID = <some ID> ev.Name = "Event_" + signal name	General: IML Event with: Name = Event_action_ev Variable with: Name = Action_action_ev Type= derived

Table C.27: Mapping of state chart events to IML system elements

C.5.8 Mapping of signals

Table C.28 specifies the mapping of state chart signals to IML system elements.

IML element	Graphical IML representation as SFC
General: variable var with var.ID = <some ID> var.Name = "Signal_" + signal name var.Type = Boolean var.SIUnit = empty var.InitialValue = empty var.Address = empty	General: Variable with: Name = Signal_signal1 Type= Boolean

Table C.28: Mapping of state chart signals to IML system elements

C.5.9 Mapping of state transitions

Table C.9 specifies the mapping of state chart state transitions to IML system elements.

IML element	Example for a graphical IML representation as SFC
Case A: <i>(Mapping of a state transition without an action to IML system elements)</i> state transition st with st.ID = <some ID> st.Name = "StateTransition_" + state transition name st.Guard.Formula = content of the state transition guard	Case A: <pre> graph TD S1[State_State_1] -- "[Guard]" --> S3[State_State_3] </pre>
Case B: <i>(Mapping of a state transition with an action to IML system elements)</i> state transition st₁ with st ₁ .ID = <some ID> st ₁ .Name = "StateTransition_" + state transition name	Case B:

st₁.Guard.Formular = content of the state transition guard

state **s** with

- s.ID = <some ID>
- s.Name = "StateForActivity_" + activity name
- s.Init = false
- s.Terminal = false
- s.Pre = st₁.ID

addData **ad** with

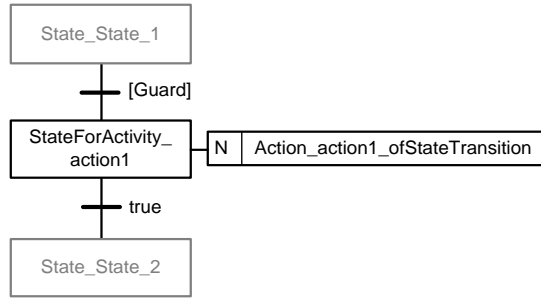
- ad.ID = <some ID>
- ad.Type = StateChartStateType
- ad.Value = stateForActivity
- ad.Pre = s.ID

action **a** with

- a.ID = <some ID>
- a.Name = "Action_" + action name + "_ofStateTransition"
- a.Formula = content of action a
- a.FiredEvents = set of events fired by a
- a.Pre = s.ID

state transition **st₂** with

- st₂.ID = <some ID>
- st₂.Name = "StateTransitionForActivity_" + activity name
- st₂.Guard.Formular = true
- st₂.Pre = s.ID



Case C:

(Mapping of a state transition from a higher state without an action to IML system elements)

Higher level state chart:

state transition **st** with

- st.ID = <some ID>
- st.Name = "StateTransition_" + state transition name
- st.Guard.Formular = content of the state transition guard
- st.Pre = s₁.ID (if st is the only successor state transition of the

Case C:

source state s_1) or selDiv.ID (if the source state s_1 has more than one successor state transition)

Sub state chart:

state s_2 with

$s_2.ID = \text{<some ID>}$

$s_2.Name = \text{"Proxy_"} + \text{state name of the originator of the state transition}$

$s_2.Init = \text{false}$

$s_2.Terminal = \text{false}$

$s_2.Pre = st.ID$

addData **ad** with

$ad.ID = \text{<some ID>}$

$ad.Type = \text{StateChartStateType}$

$ad.Value = \text{higherLevelState}$

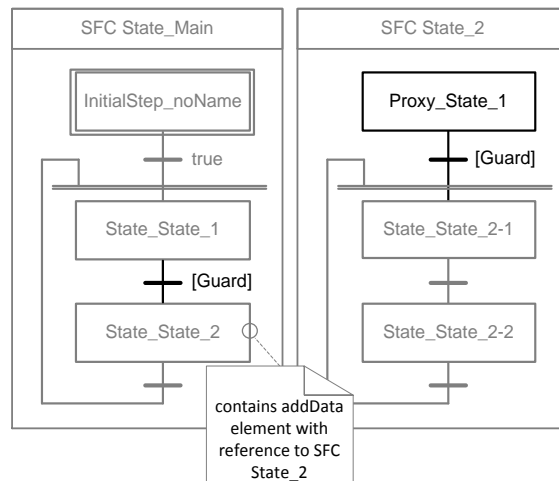
$ad.Pre = s_2.ID$

state transition **st** with

$st.Name = \text{"StateTransition_"} + \text{state transition name}$

$st.Guard.Formula = \text{content of the state transition guard}$

$st.Pre = s_2.ID$ (if st is the only successor state transition of the state s_2) or selDiv.ID (if the state s_2 has more than one successor state transition)



Case D:

(Mapping of a state transition from a higher state with an action to IML system elements)

Higher level state chart:

state transition **st** with

$st.ID = \text{<some ID>}$

$st.Name = \text{"StateTransition_"} + \text{state transition name}$

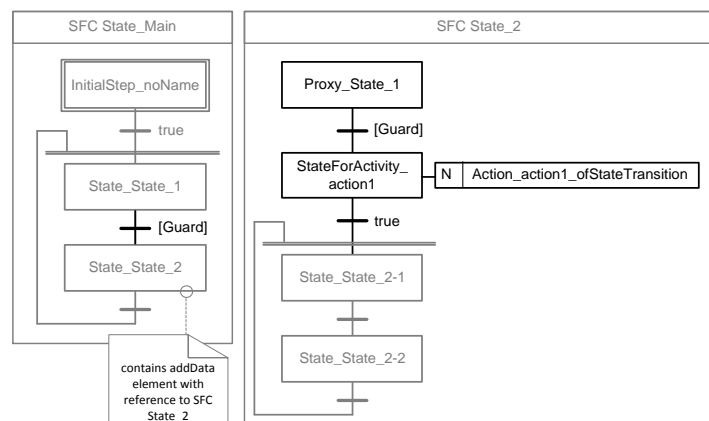
$st.Guard.Formula = \text{content of the state transition guard}$

$st.Pre = s_1.ID$ (if st is the only successor state transition of the source state s_1) or selDiv.ID (if the source state s_1 has more than one successor state transition)

Sub state chart:

state s_1 with

Case D:



```

s1.ID = <some ID>
s1.Name = "Proxy_" + state name of
the originator of the state transition
s1.Init = false
s1.Terminal = false
addData ad with
ad.ID = <some ID>
ad.Type = StateChartStateType
ad.Value = higherLevelState
ad.Pre = s1.ID
state transition st1 with
st1.ID = <some ID>
st1.Name = "StateTransition_" + state
transition name
st1.Guard.Formular = content of the
state transition guard
st1.Pre = s1.ID (if st1 is the only
successor state transition of the
source state s1) or selDiv.ID (if the
source state s1 has more than one
successor state transition)
state s2 with
s2.ID = <some ID>
s2.Name = "StateForActivity_" +
activity name
s2.Init = false
s2.Terminal = false
s2.Pre = st1.ID
addData ad with
ad.ID = <some ID>
ad.Type = StateChartStateType
ad.Value = stateForActivity
ad.Pre = s2.ID
state transition st2 with
st2.ID = <some ID>
st2.Name = "StateTransitionForActivity_" +
activity name
st2.Guard.Formular = true
st2.Pre = s2.ID
action a with
a.ID = <some ID>

```

a.Name = "Action_" + action name + "_ofStateTransition"

a.Formula = content of action a

a.FiredEvents = set of events fired by a

a.Pre = s₂.ID

Case E:

(Mapping of a state transition from a lower level state without an action to IML system elements)

Higher level state chart:

state transition **st** with

st.ID = <some ID>

st.Name = "StateTransition_" + state transition name

st.Guard.Formula = content of the state transition guard

st.Pre = s₁.ID (if st is the only successor state transition of the source state s₁) or selDiv.ID (if the source state s₁ has more than one successor state transition)

Sub state chart:

state **s₃** with

s₃.ID = <some ID>

s₃.Name = "Proxy_" + state name of the originator of the state transition

s₃.Init = false

s₃.Terminal = false

addData **ad** with

ad.ID = <some ID>

ad.Type = StateChartStateType

ad.Value = higherLevelState

ad.Pre = s₃.ID

state transition **st** with

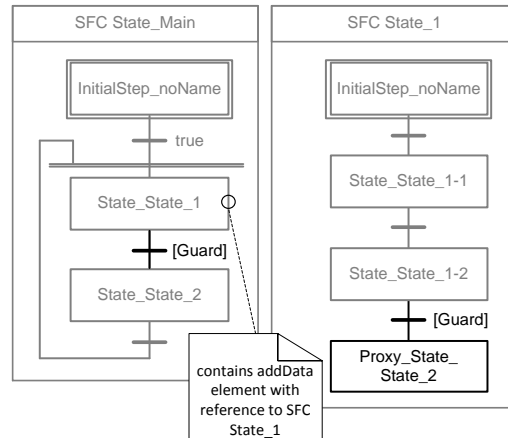
st.ID = <some ID>

st.Name = "StateTransition_" + state transition name

st.Guard.Formula = content of the state transition guard

st.Pre = s.ID (if st is the only successor state transition of the source state s₁) or selDiv.ID (if the source state s₁ has more than one

Case E:



successor state transition)

Case F:

(Mapping of a state transition from a lower level state with an action to IML system elements)

Higher level state chart:

state transition **st₁** with

st₁.ID = <some ID>

st₁.Name = "StateTransition_" + state transition name

st₁.Guard.Formular = content of the state transition guard

st₁.Pre = s₁.ID (if st₁ is the only successor state transition of the state s₁) or selDiv.ID (if the state s₁ has more than one successor state transition)

state **s₁** with

s₁.ID = <some ID>

s₁.Name = StateForActivity_" + activity name

s₁.Init = false

s₁.Terminal = false

s₁.Pre = st₁.ID

addData **ad** with

ad.ID = <some ID>

ad.Type = StateChartStateType

ad.Value = stateForActivity

ad.Pre = s₁.ID

state transition **st₂** with

st₂.ID = <some ID>

st₂.Name = "StateTransitionForActivity_" + activity name

st₂.Guard.Formular = true

st₂.Pre = s₁.ID

action **a** with

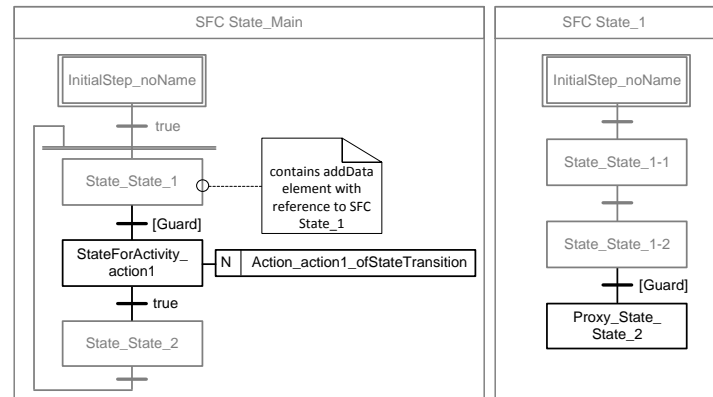
a.ID = <some ID>

a.Name = "Action_" + action name + "_ofStateTransition"

a.Formula = content of action a

a.FiredEvents = set of events fired by a

Case F:



<p>a.Pre = s₁.ID</p> <p><i>Sub state chart:</i></p> <p>state s₄ with</p> <p> s₄.ID = <some ID></p> <p>s₄.Name = "Proxy_" + state name of the originator of the state transition</p> <p> s₄.Init = false</p> <p> s₄.Terminal = false</p> <p>addData ad with</p> <p> ad.ID = <some ID></p> <p> ad.Type = StateChartStateType</p> <p> ad.Value = higherLevelState</p> <p> ad.Pre = s₄.ID</p> <p>state transition st with</p> <p> st.ID = <some ID></p> <p>st.Name = "StateTransition_" + state transition name</p> <p>st.Guard.Formula = content of the state transition guard</p> <p>st.Pre = s₄.ID (if st is the only successor state transition of the source state s₄) or selDiv.ID (if the source state s₄ has more than one successor state transition)</p>	
---	--

Table C.29: Mapping of state chart state transitions to IML system elements

C.5.10 Mapping of history connectors

Table C.30 specifies the mapping of history connectors of state charts to IML system elements.

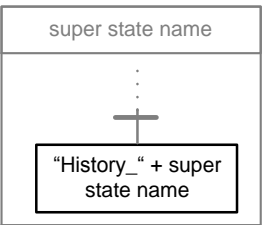
IML element	Graphical IML representation as SFC
<p>General:</p> <p>state s with</p> <p style="padding-left: 40px;">s.ID = <some ID></p> <p>s.Name = "History_" + state name of the super state of the history connector</p> <p style="padding-left: 40px;">s.Init = false</p> <p style="padding-left: 40px;">s.Terminal = false</p> <p>addData ad with</p> <p style="padding-left: 40px;">ad.ID = <some ID></p> <p style="padding-left: 40px;">ad.Type = StateChartStateType</p> <p style="padding-left: 40px;">ad.Value = historyConnector</p> <p style="padding-left: 40px;">ad.Pre = s.ID</p>	<p>General:</p> 

Table C.30: Mapping of history connectors of state charts to IML system elements

C.5.11 Mapping of condition connectors

Table C.31 specifies the mapping of condition connectors of a state chart to IML system elements.

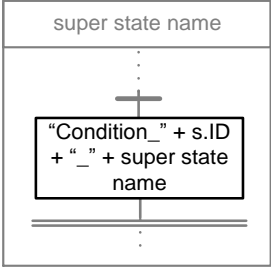
IML element	Graphical IML representation as SFC
<p>General:</p> <p>step s with</p> <p style="padding-left: 40px;">s.ID = <some ID></p> <p>s.Name = "Condition_" + s.ID + "_" + state name of the state the condition connector is directly integrated in</p> <p style="padding-left: 40px;">s.Init = false</p> <p style="padding-left: 40px;">s.Terminal = false</p> <p>addData ad with</p> <p style="padding-left: 40px;">ad.ID = <some ID></p> <p style="padding-left: 40px;">ad.Type = StateChartStateType</p> <p style="padding-left: 40px;">ad.Value = Condition Connector</p> <p style="padding-left: 40px;">ad.Pre = s.ID</p>	<p>General:</p> 

Table C.31: Mapping of state chart condition connectors to IML system elements

Annex D Referencing methods for logic information

D.1 General

This clause describes the referencing methods for logic information, which are stored in IEC 61131-XML documents. It comprises the referencing of logic information which is stored within one POU, which is distributed throughout several POU's, and the referencing of interlocking information which is stored throughout several IEC 61131-XML documents.

Logic information is expressed as logic models. To reference not only the logic model itself (as specified in D.2) but also certain parts of that logic model, e.g. a variable, additional referencing methods are specified in D.3.

Note: In the CAEX file, GUIDs as an implementation of UUIDs are applied for object identification (see IEC 62714-1), e.g. AC76BA86-7AD7-1033-7B44-A70000000000. GUIDs are presented in a short form such as "GUID1", "GUID100" etc. This serves the readability and acts as a real GUID.

D.2 Referencing logic information expressed as logic models

This subclause describes the referencing of logic information expressed as logic models of an AML object as specified in clause 6.

D.2.1 Referencing logic information stored in one POU

Sequencing, behaviour, or interlocking information, stored as a logic model within one POU, is referenced by modelling a CAEX ExternalInterface with an AML InterfaceClass "LogicInterface", "SequencingLogicInterface", "BehaviourLogicInterface", "SequencingBehaviourLogicInterface", "InterlockingLogicInterface", or a derivation of them, associated to it. This is depicted in Figure D.1 and Figure D.2, in which logic information expressed as SFC is referenced.

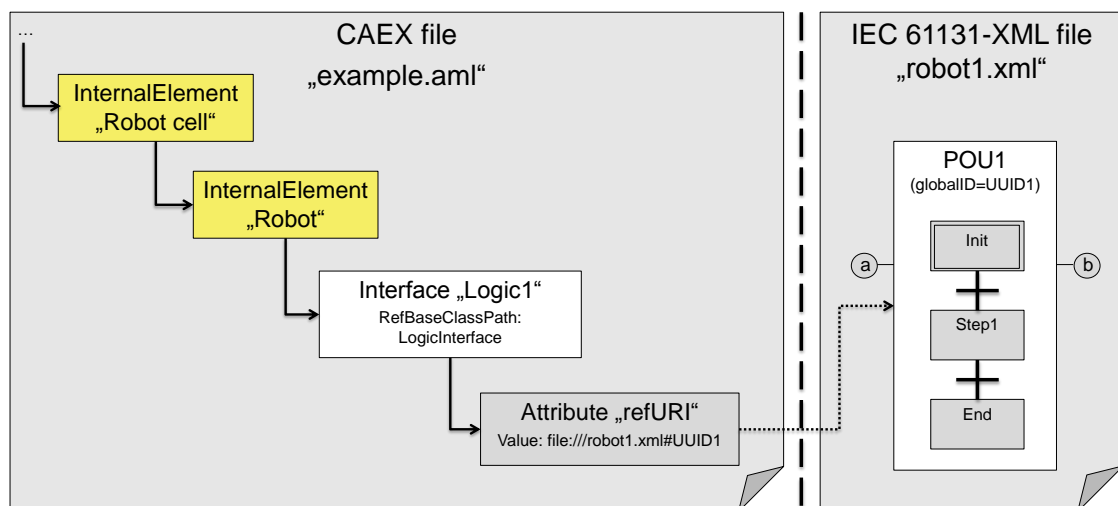


Figure D.1: Referencing logic information (as SFC) stored in one POU

```
<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>
```

Figure D.2: XML text of the CAEX file for referencing logic information stored in one POU

Logic information can also be expressed as FBD as defined in A.3. The referencing method remains unaffected.

D.2.2 Referencing logic information distributed throughout several POU

Sequencing, behaviour, or interlocking information, stored as a logic model distributed throughout several POU, is referenced by modelling a CAEX ExternalInterface with an AML InterfaceClass “LogicInterface”, “SequencingLogicInterface”, “BehaviourLogicInterface”, “SequencingBehaviourLogicInterface”, “InterlockingLogicInterface”, or a derivation of them, associated to it.

This is depicted in Figure D.3. Here, a POU, which contains a SFC, calls other POU containing SFC or FBD by IEC 61131-XML means. Also a POU, which contains a FBD, can call other POU. The referencing method remains unaffected. The XML text of the CAEX file for referencing logic information distributed throughout several POU is the same as depicted in Figure D.2.

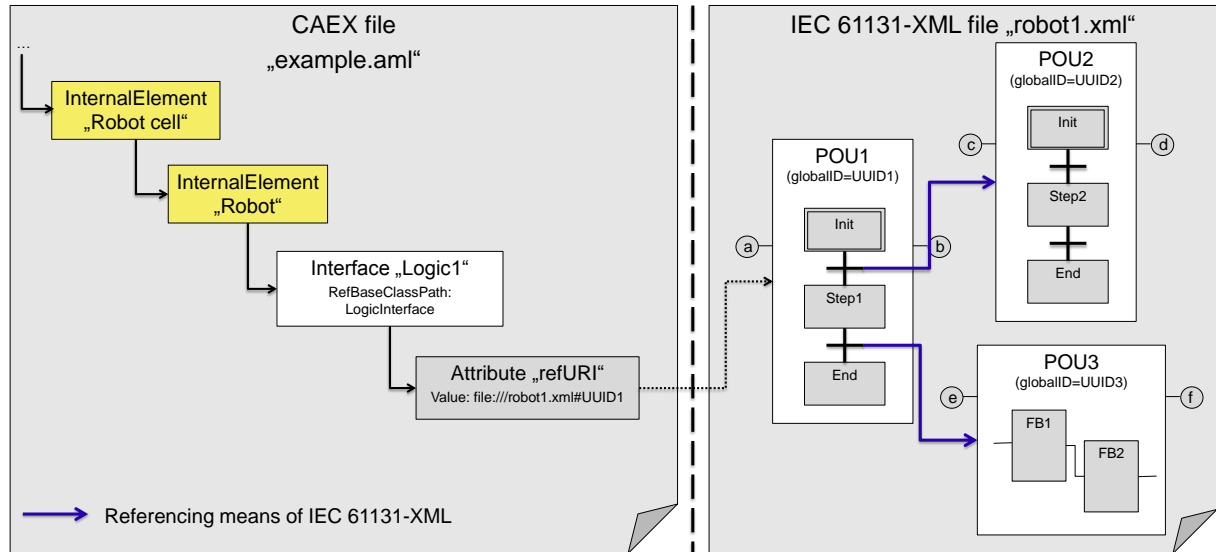


Figure D.3: Referencing logic information distributed throughout several POU

It is also possible that an IEC 61131-XML document contains several POU which are not interlinked with each other. In that case, the referencing method as specified in D.2.1 is applied.

D.2.3 Referencing interlocking information distributed throughout several IEC 61131-XML documents

This referencing method is only valid for interlocking information as specified in clause 10.

Interlocking information, stored as a logic model distributed throughout several IEC61131-XML documents, is modelled by a CAEX ExternalInterface for each IEC61131-XML document throughout the logic model containing the interlocking information is distributed. These CAEX ExternalInterfaces are associated with an AML InterfaceClass “InterlockingLogicInterface”, or a derivation of them. This is depicted in Figure D.4 and Figure D.5. Here, the interlocking information is distributed throughout two IEC61131-XML documents.

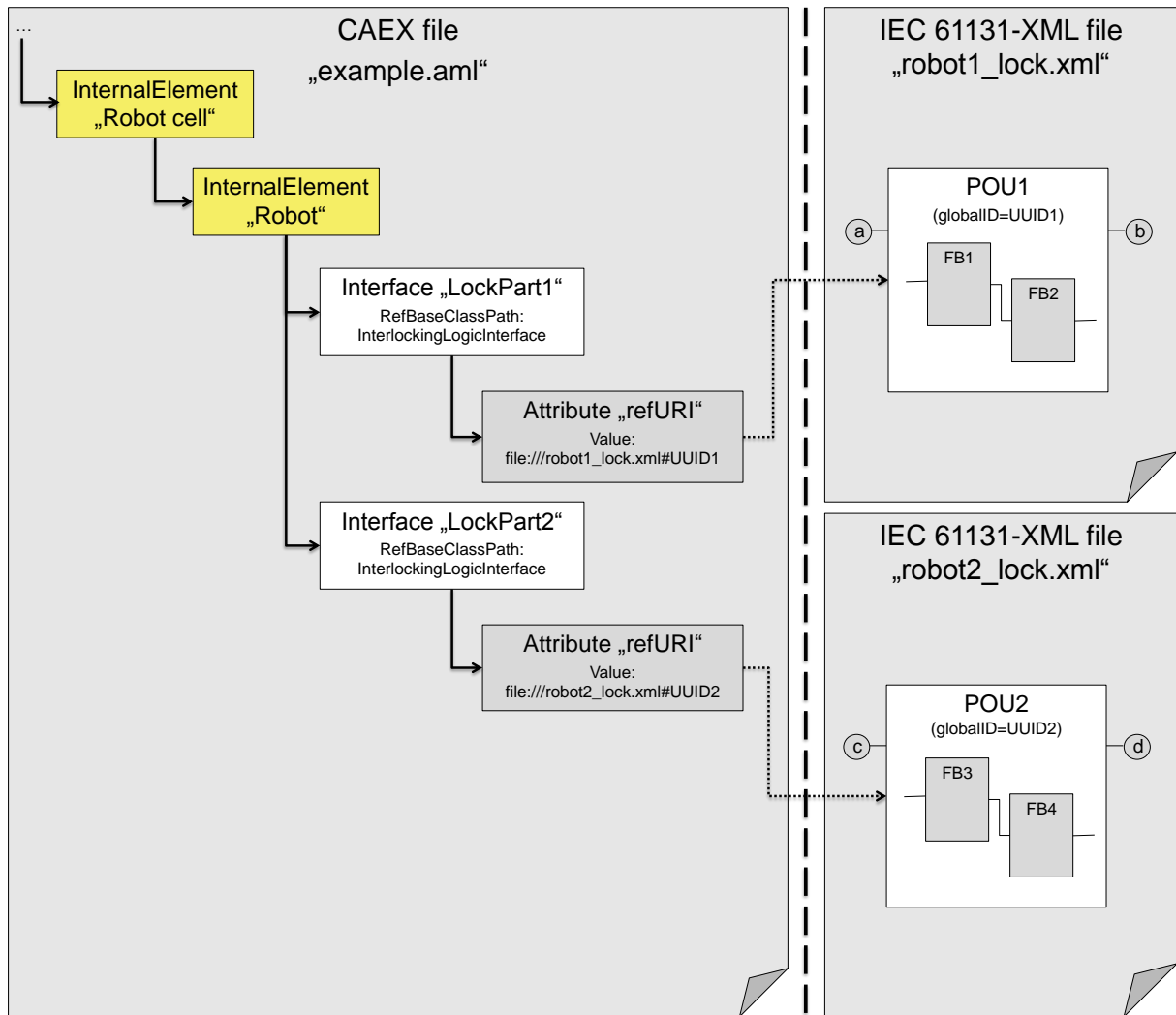


Figure D.4: Referencing interlocking information distributed throughout several IEC 61131-XML documents

```
<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="LockPart1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1_lock.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="LockPart2"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot2_lock.xml#UUID2</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>
```

Figure D.5: XML text of the CAEX file for referencing interlocking information distributed throughout several IEC 61131-XML documents

To connect both IEC61131-XML documents with each other, the relevant parts need to be referenced as well. They can be connected by CAEX means. In Figure D.6 and Figure D.6 the output variable of POU1 is connected with the input variable of POU2. The referencing method for variable is described in D.3.

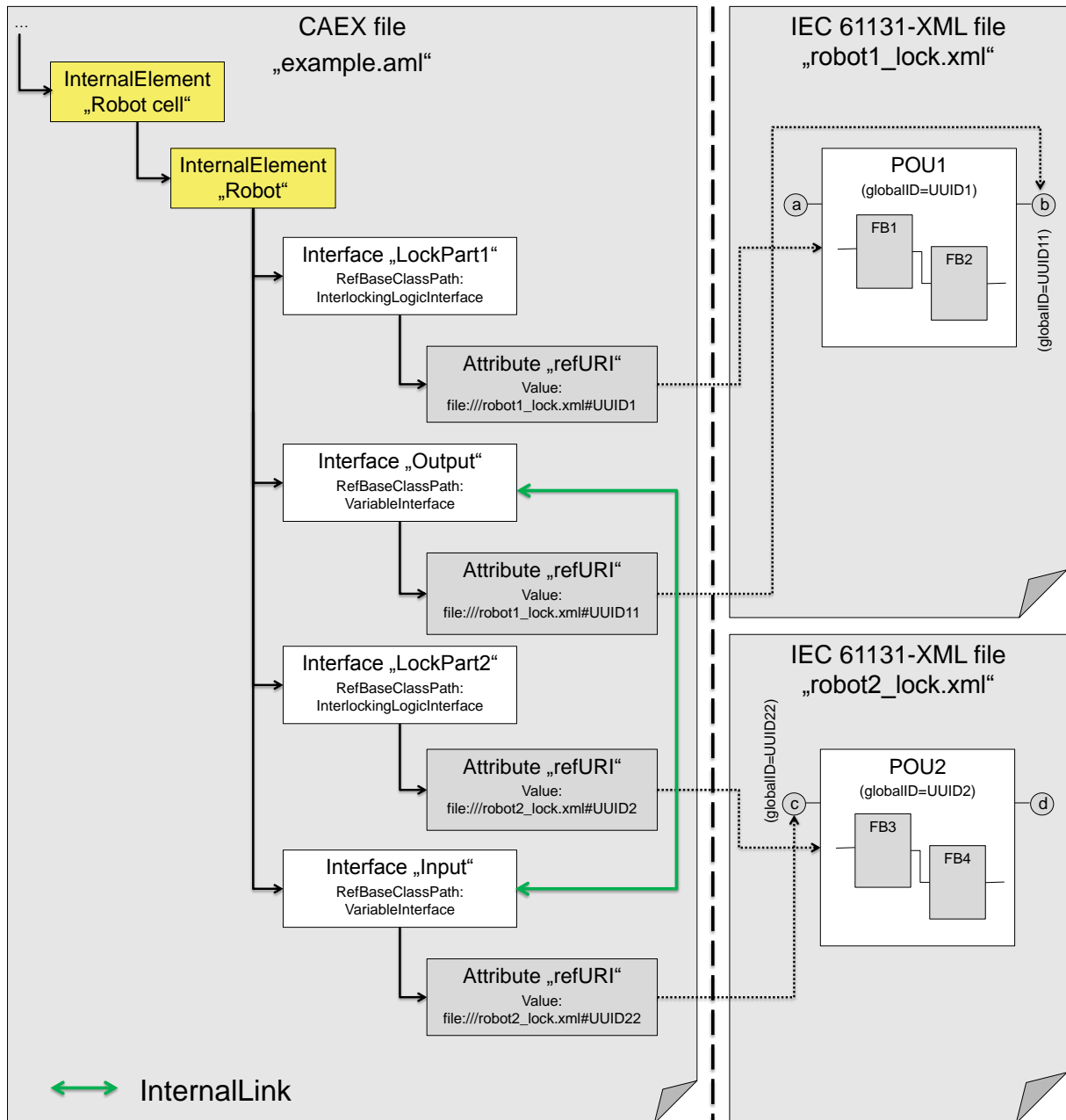


Figure D.6: Referencing interlocking information distributed throughout several IEC 61131-XML documents using CAEX means

```

<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="LockPart1"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1_lock.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Output"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1_lock.xml#UUID11</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="LockPart2"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot2_lock.xml#UUID2</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Input"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot2_lock.xml#UUID22</Value>
      </Attribute>
    </ExternalInterface>
    <InternalLink Name="Link_OutputInput" RefPartnerSideA="GUID101:Output"
RefPartnerSideB="GUID101:Input"/>
  </InternalElement>
</InternalElement>

```

Figure D.7: XML text of the CAEX file for referencing interlocking information distributed throughout several IEC 61131-XML documents using CAEX means

Normative provisions are given in 10. An informative overview is provided by Annex F.

D.3 Referencing logic information as a part of logic models

This subclause describes the referencing of logic information as a part of logic models of an AML object as specified in clause 6.

D.3.1 Referencing a variable

A variable in a logic model, containing sequencing, behaviour, or interlocking information, is referenced by modelling a CAEX ExternalInterface of the AML InterfaceClass “VariableInterface”. This is depicted in Figure D.8 and

```
<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Output"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID11</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>
```

Figure D.9, in which variable “b” is referenced. But the logic model (stored in “POU1”) of which the variable is a part of needs to be referenced before as described in D.2.

Note: Variables can be signals, parameters, internal variables.

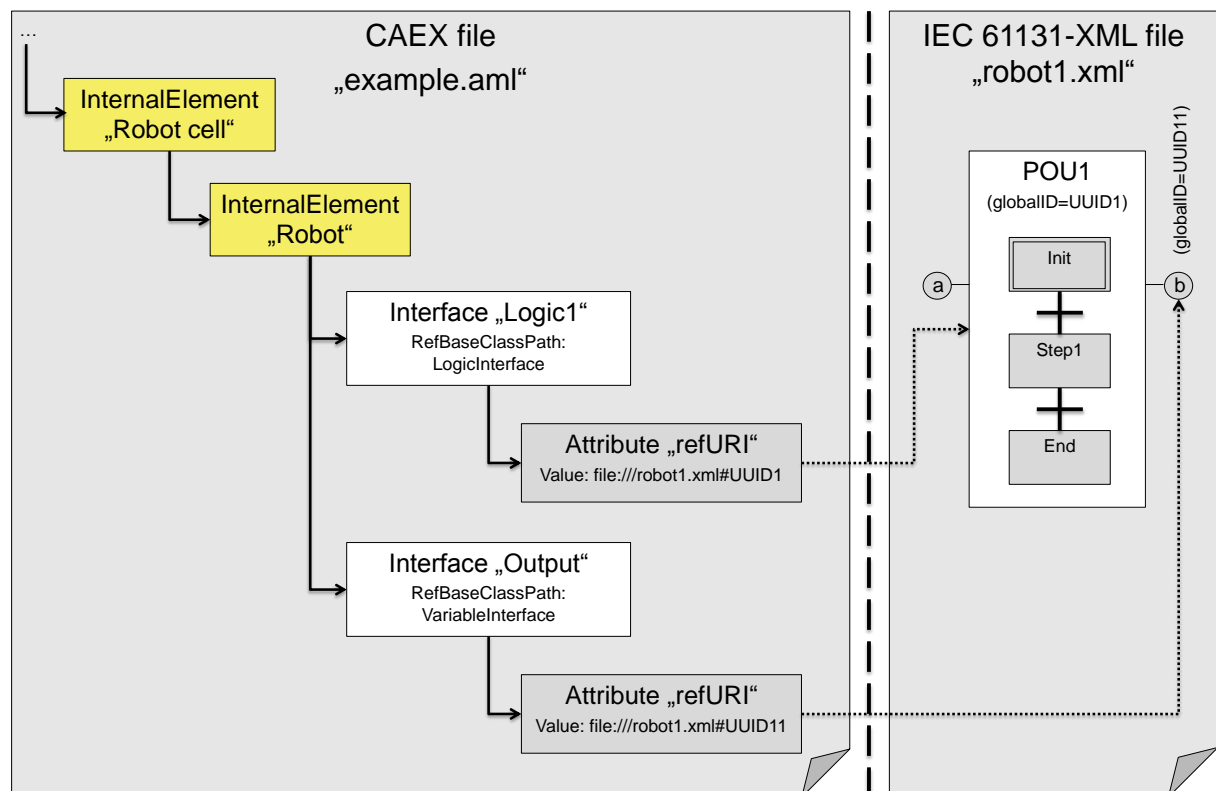


Figure D.8: Referencing a variable

```

<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
    RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
    ector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Output"
    RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
    ector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID11</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>

```

Figure D.9: XML text of the CAEX file for referencing a variable

D.3.2 Referencing an interlocking variable

An interlocking variable in a logic model, containing interlocking information, is referenced by modelling a CAEX ExternalInterface of the AML InterfaceClass “InterlockingVariableInterface”. This is depicted in Figure D.10 and Figure D.11, in which a variable “b” is referenced. But the logic model (stored in “POU1”) of which the interlocking variable is a part of needs to be referenced before as described in D.2.

Normative provisions are given in clause 10. An informative overview is provided by Annex F.

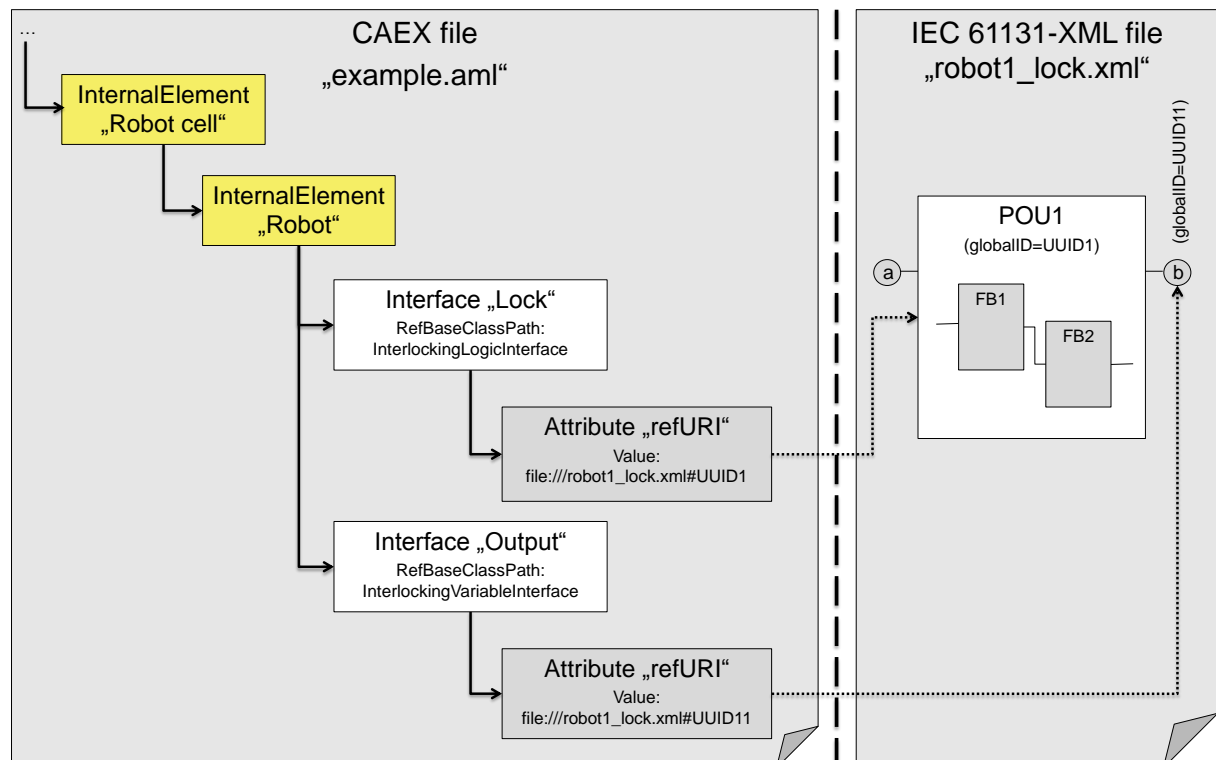


Figure D.10: Referencing an interlocking variable

```

<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Lock"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1_lock.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Output"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface/InterlockingVariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1_lock.xml#UUID11</Value>
      </Attribute>
      <Attribute Name="SafeConditionEquals" AttributeDataType="xs:boolean">
        <DefaultValue>true</DefaultValue>
        <Value>true</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>

```

Figure D.11: XML text of the CAEX file for referencing an interlocking variable

D.3.3 Referencing a logic object

A logic object in a logic model, containing sequencing, behaviour, or interlocking information, is referenced by modelling a CAEX ExternalInterfaces of the AML InterfaceClass “LogicObjectInterface”. This is depicted in Figure D.12 and Figure D.13

```

<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Process1"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicObjectInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID13</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>

```

Figure D.13, where the step “Step1” is referenced. But the logic model (stored in “POU1”) of which the logic object is a part of needs to be referenced before as described in D.2.

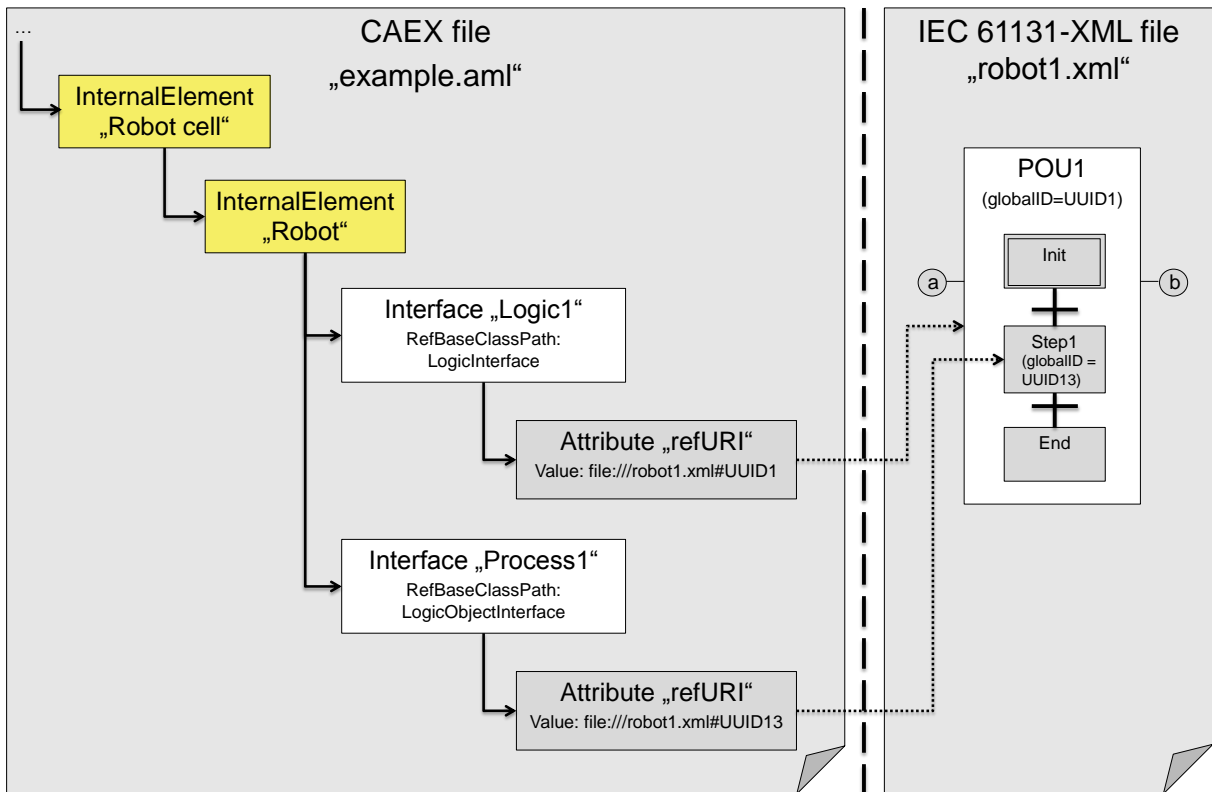


Figure D.12: Referencing a logic object

```
<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Process1"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicObjectInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID13</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
</InternalElement>
```

Figure D.13: XML text of the CAEX file for referencing a logic object

The following modelling elements of SFC are logic objects:

- <step>
- <transition> within <SFC> within <body>
- <action> within <SFC> within <body>
- <selectionDivergence>
- <simultaneousDivergence>
- <selectionConvergence>
- <simultaneousConvergence>

The following modelling elements of FBD are logic objects:

- <block> within <FBD> within <body>
- <action> within <FBD> within <body> within <actionBlock>
- All <inVariable>s, <outVariable>, <inOutvariable> in <FBD>

D.4 Referencing logic information as a part of already referenced logic models

Parts of a logic model, containing sequencing, behaviour, or interlocking information, are referenced from AML objects as described in D.3. But in case that this logic model is already referenced by one of its parent AML objects, it is not necessary to reference this logic model again. Instead a prompt referencing of the parts of this logic model is possible. This is depicted in Figure D.14 and Figure D.15, in which a variable “b” of an already referenced logic model is referenced.

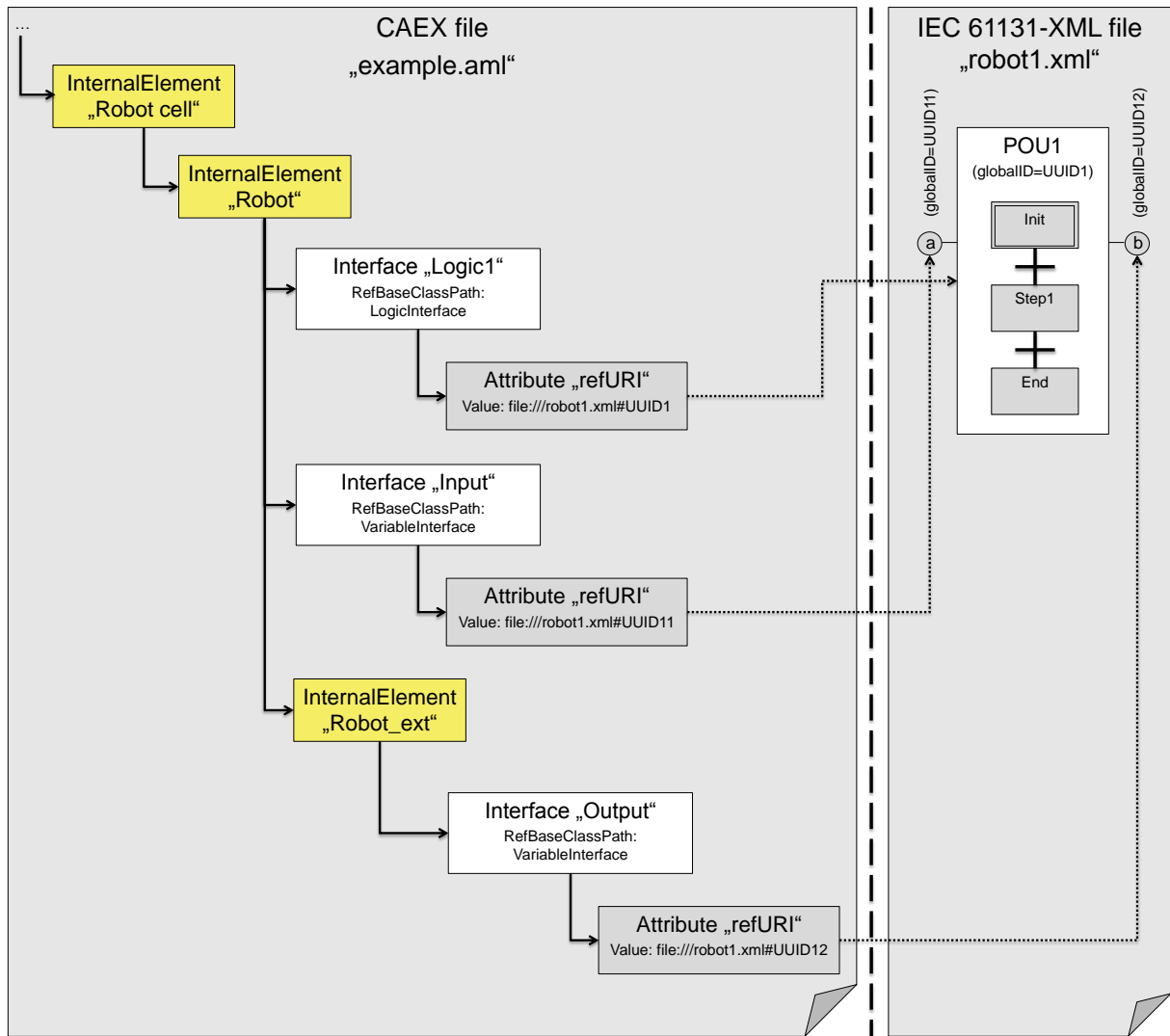


Figure D.14: Referencing a variable of an already referenced logic model

```

<InternalElement Name="Robot cell" ID="GUID100">
  <InternalElement Name="Robot" ID="GUID101">
    <ExternalInterface Name="Logic1"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Input"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///robot1.xml#UUID11</Value>
      </Attribute>
    </ExternalInterface>
    <InternalElement Name="Robot_ext" ID="GUID102">
      <ExternalInterface Name="Output"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface">
        <Attribute Name="refURI" AttributeDataType="xs:anyURI">
          <Value>file:///robot1.xml#UUID12</Value>
        </Attribute>
      </ExternalInterface>
    </InternalElement>
  </InternalElement>
</InternalElement>

```

Figure D.15: XML text of the CAEX file for referencing a variable of an already referenced logic model

Annex E Using mathematical expressions in logic information

E.1 General

This annex shows, how the provisions of clause 9 for the integration of MathML into IEC 61131-XML are applied to an example.

E.2 Example description

The example depicts an one-way flow control valve with exhaust-air flow control (see Figure E.1).

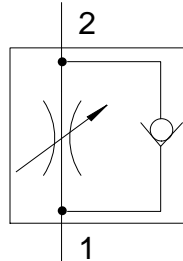


Figure E.1: Diagram of an one-way flow control valve with exhaust-air flow control

It is used to regulate exhaust air flow rates with double-acting cylinders. In the reverse direction, the air flows freely through the non-return valve with full cross-sectional flow.

E.3 Mathematical functions expressed in MathML for logic information

For modelling behaviour information and sequencing information, Gantt charts, activity-on-node networks, timing diagrams, and state charts are not sufficient in some cases, especially with regard to the modelling of physical behaviour. For this purpose MathML can be used in the “addData” elements of IEC 61131-XML. This makes it possible to evaluate the corresponding MathML formulas and, thus, to influence the switching behaviour in event-discrete models.

For the example described in E.2, MathML could be used to describe the flow rate of the valve. The flow rate depends on the pressure at the input and output side and the adjustment of the adjustment screw. In the supercritical range the flow rate remains constant with a changing input/output pressure ratio. In the subcritical range the flow rate decreases if the input/output pressure decreases, see Figure E.2.

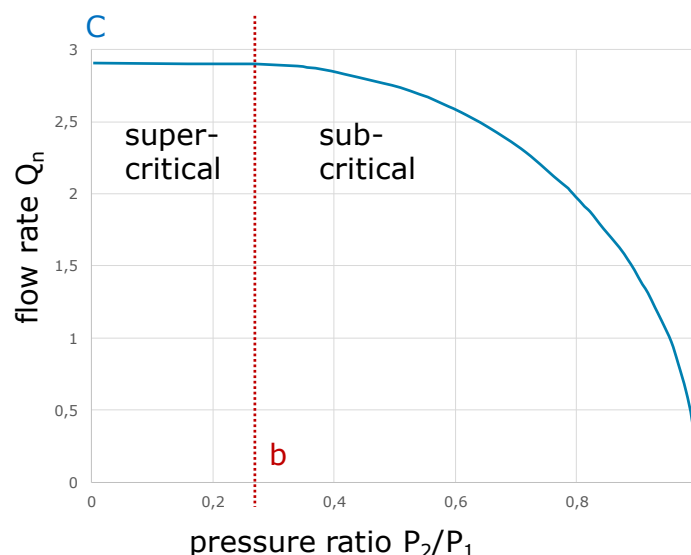


Figure E.2: Flow rate of valves

The flow rate can be calculated with the following formula:

$$Q = \begin{cases} P_1 * n * C * \sqrt{1 - \left(\frac{P_2 - b}{P_1}\right)^2}, & \frac{P_2}{P_1} \geq b \text{ (subcritical)} \\ P_1 * n * C, & \frac{P_2}{P_1} < b \text{ (supercritical)} \end{cases} \quad (1)$$

Q = flow rate

P1 = input absolute static pressure

P2 = output absolute static pressure

C = sonic conductance

b = critical pressure ratio

n = setting of adjustment screw (0..1)

According to the provisions of clause 9 formulas can be integrated in IEC 61131-XML by expressing them with the content part of MathML version 2.0.

Integration of MathML expressions in IEC 61131-XML “addData” elements

According to the provisions of clause 9 MathML expressions can be integrated in IEC 61131-XML in “addData” elements.

It is necessary to create an “addData” element, which includes two “data” elements:

- One “data” element for describing the mapping between IEC 61131-XML and MathML and
- One “data” element for describing the MathML expression itself.

The “data” element for describing the MathML expression should be structured as shown in Table 60. For the given example of the flow rate of the control valve the “data” element is shown in Figure E.3.

```
<data name="http://www.w3.org/1998/Math/" handleUnknown="preserve">
  <math xmlns="http://www.w3.org/1998/Math/MathML/" display="block">
    <apply>
      <eq/>
      <ci>Q</ci>
      <piecewise>
        <piece>
          <apply>
            <times/>
            <ci>P1</ci>
          </apply>
          <times/>
          <ci>n</ci>
        </piece>
        <piece>
          <times/>
          <ci>C</ci>
        </piece>
      </piecewise>
    </apply>
  </math>
</data>
```

```
<divide/>
<apply>
  <minus/>
    <apply>
      <divide/>
        <ci>P2</ci>
        <ci>P1</ci>
      </apply>
    <ci>b</ci>
  </apply>
<apply>
  <minus/>
    <cn>1</cn>
    <ci>b</ci>
  </apply>
</apply>
<cn>2</cn>
</apply>
</apply>
</apply>
</apply>
</apply>
</apply>
</apply>
</piece>
<piece>
  <apply>
    <times/>
      <ci>P1</ci>
    <apply>
      <times/>
        <ci>n</ci>
        <ci>C</ci>
      </apply>
    </apply>
  </apply>
</piece>
```

```

</t/>
<apply>
  <divide/>
  <ci>P2</ci>
  <ci>P1</ci>
</apply>
<ci>b</ci>
</apply>
</piece>
</piecewise>
</apply>
</math>
</data>

```

Figure E.3: “data” element for describing the MathML expression of the flow rate of a control valve

The content of the “data” element for describing the mapping between IEC 61131-XML and MathML depends on the formula and the amount of variables, which should be mapped.

For the given example, six variables are included in the formula: One output variable (Q) and five input variables (P1, P2, C, b and n). These variables can be mapped to the same amount of IEC 61131-XML variables, which are referenced by their globalIDs. Figure E.4 depicts the “data” element for the mapping.

```

<data name="http://www.automationml.org/IEC62714-4Ed1/MathMLinIEC61131-XML.xsd"
handleUnknown="preserve">
  <formula Name="flow rate of a control valve" ID="UUID34">
    <variable refMathMLVariable="Q" refGlobalID="UUID63" direction="Out"/>
    <variable refMathMLVariable="P1" refGlobalID="UUID64" direction="In"/>
    <variable refMathMLVariable="P2" refGlobalID="UUID65" direction="In"/>
    <variable refMathMLVariable="C" refGlobalID="UUID66" direction="In"/>
    <variable refMathMLVariable="b" refGlobalID="UUID67" direction="In"/>
    <variable refMathMLVariable="n" refGlobalID="UUID68" direction="In"/>
  </formula>
</data>

```

Figure E.4: “data” element for describing the mapping of MathML variables for calculating the flow rate of a control valve to IEC 61131-XML variables

Both “data” elements are a part of the same “addData” element. The structure of the “addData” element for the example is shown in Figure E.5.

```

<addData>
  <data name="http://www.automationml.org/IEC62714-4Ed1/MathMLinIEC61131-XML.xsd"
    handleUnknown="preserve">
    <formula Name="flow rate of a control valve" ID="UUID34">
      ...
    </formula>
  </data>
  <data name="http://www.w3.org/1998/Math/" handleUnknown="preserve">
    <math xmlns="http://www.w3.org/1998/Math/MathML/" display="block">
      ...
    </math>
  </data>
</addData>

```

Figure E.5. “addData” element including the MathML expression for the flow rate of a control valve and the mapping of MathML variables to IEC 61131-XML variables

E.4 Integration positions in IEC 61131-XML for “addData” elements including MathML expressions

MathML expressions can be only assigned to certain SFC and FBD elements in IEC 61131-XML, see 9.2. The position for the integration depends on the scope of the formula, which is expressed with MathML, e. g. if a formula describes the behaviour of a component while a certain step is active, the “addData” element of this step would be the appropriate position for the formula.

For the given example the flow rate could be relevant for a POU, which describes the behaviour of the one-way flow control valve. The integration of the “addData” element for such a POU is shown in Figure E.6.

```

<types>
  <datatypes/>
  <pous>
    <pou pouType="functionBlock" name="one-way flow control valve" globalId="UUID60">
      <interface/>
      <actions/>
      <transitions/>
      <body/>
      <addData>
        <data name="http://www.automationml.org/IEC62714-4Ed1/MathMLinIEC61131-XML.xsd"
          handleUnknown="preserve">
          <formula Name="flow rate of a control valve" ID="UUID34">
            ...
          </formula>
        </data>
        <data name="http://www.w3.org/1998/Math/" handleUnknown="preserve">
          <math xmlns="http://www.w3.org/1998/Math/MathML/">
            ...
          </math>
        </data>
      </addData>
    </pou>
  </pous>
</types>

```

Figure E.6: Integration of the “addData” element including a MathML expression about the flow rate of a control valve in a POU

Annex F Referencing interlocking information

F.1 General

This clause describes the two mechanisms for referencing interlocking information, which is stored in IEC 61131-XML documents. This comprises the modelling and storage of interlocking information without and with an explicitly modelled interlocking condition. Both levels are based on each other and are explained consecutively using one and the same example.

Normative provisions are given in clause 10.

Note: In the CAEX file, GUIDs as an implementation of UUIDs are applied for object identification (see IEC 62714-1), e.g. AC76BA86-7AD7-1033-7B44-A70000000000. GUIDs are presented in a short form such as “GUID1”, “GUID100” etc. This serves the readability and acts as a real GUID.

F.2 Example description

The example depicts a manufacturing system containing different manufacturing equipment and safety devices (see Figure F.1).

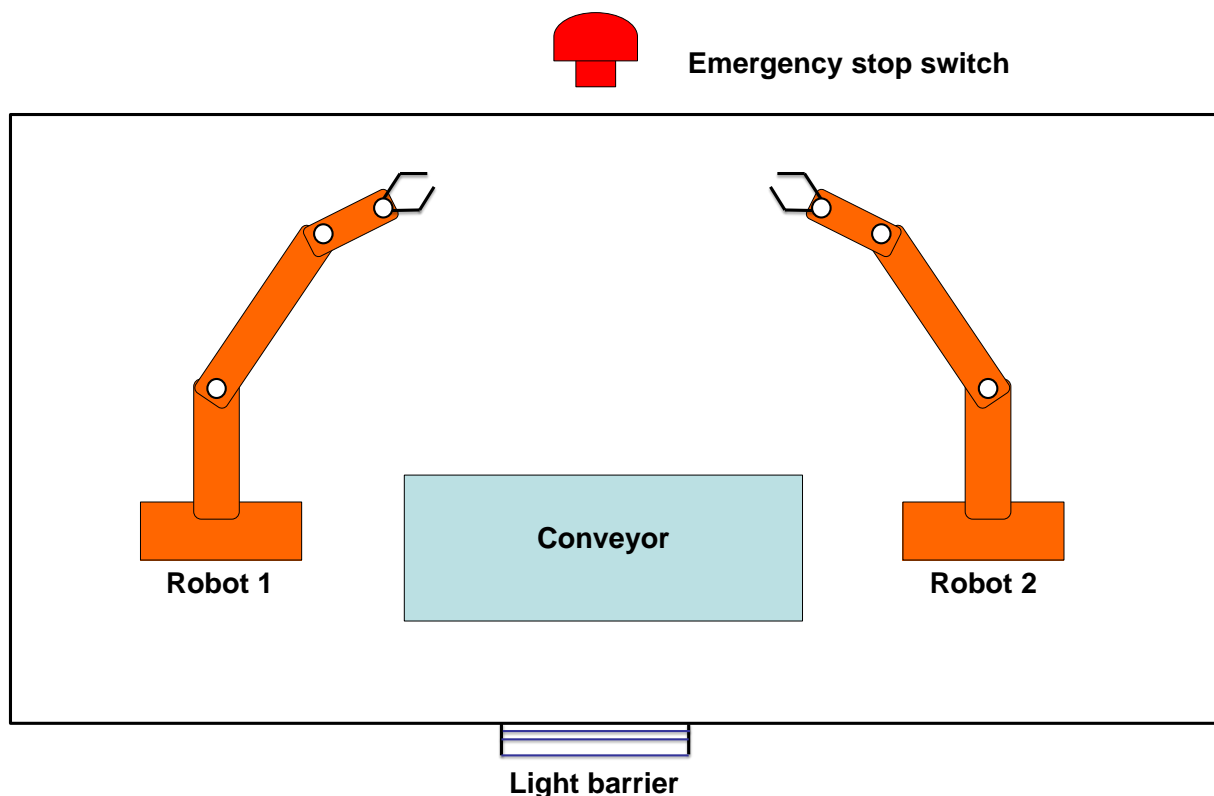


Figure F.1: Example manufacturing system

The robot cell contains the objects respectively manufacturing resources “Robot 1”, “Robot 2” and “Conveyor”. It also contains the safety devices “Light barrier” and “Emergency stop switch”. Figure F.2 shows the exemplary aggregation of the manufacturing resources into an interlocking source group and an interlocking target group. It is allowed to have more than one interlocking source group and interlocking target group. In this example, there is one of each.

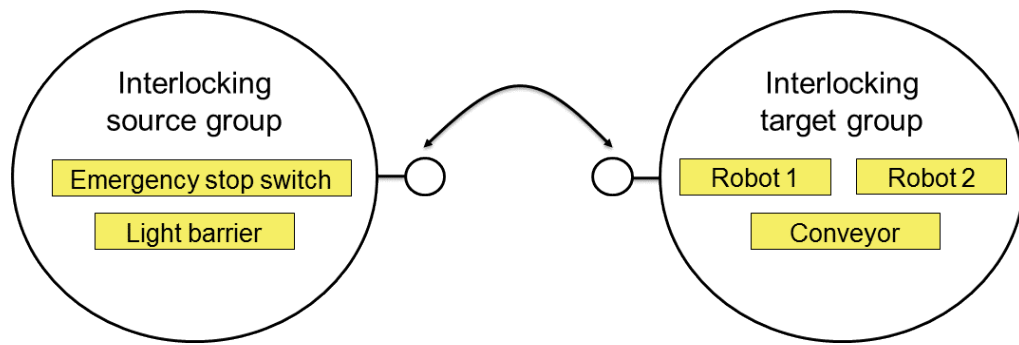


Figure F.2: Example interlocking source group and interlocking target group

F.3 Interlocking information

The third, complementary concept of logic information covered by AML is the interlocking information expressed on two different levels of detail belonging to an industrial production system or to single components. It reflects the two main concepts:

- The causes of an interlocking condition and,
- The resulting effects of an interlocking condition.

Hereby, the cause represents the situation resulting in the need to interlock something. This case is covered by the definition and description of interlocking source groups. The effect to other groups of objects is expressed by the definition, description of and assignment to interlocking target groups.

These relations between interlocking source group and interlocking target group express the relation between cause and effect within interlocking information. In AML different levels of detail are used to exchange these interlocking conditions. Therefore, AML provides two mechanisms to reference interlocking information.

To describe interlocking source groups and interlocking target groups the basic AML group concept is used. This concept exploits fundamental CAEX capabilities and enables an integration of the necessary information within the AML top-level documents (see F.4).

To express the relation between interlocking source group and interlocking target group and to express the internal logical relation within the groups, Function Block Diagrams (FBD) of the IEC 61131-3 are used (see F.5). They are stored in IEC 61131-XML documents similar to sequencing and behaviour information.

F.4 Referencing interlocking information without interlocking condition

On the first level of detail, functional dependencies among groups of objects are modelled: without explicitly modelling an interlocking condition which would further specify that functional dependency. Objects that indicate unsafe states within the manufacturing system are aggregated to the interlocking source group. These are the “Light barrier” and “Emergency stop switch”. Objects that are influenced in their behaviour by the state of the interlocking source group are aggregated to the interlocking target group. These are the “Robot 1”, “Robot 2”, and “Conveyor”. They need to execute, then, specific activities to re-establish safe system behaviour, e.g. stop of any movement of the objects of the interlocking target group.

For each group, an additional CAEX InternalElement, with either an AML RoleClass “InterlockingSourceGroup” or “InterlockingTargetGroup”, is modelled to separate structure information from instance information. Each group contains a CAEX ExternalInterface of the AML InterfaceClass “InterlockingConnector” to relate the interlocking source group and interlocking target group to each other. CAEX Objects which belong to one of those groups are modelled as mirror objects (see Figure F.3 and Figure F.4).

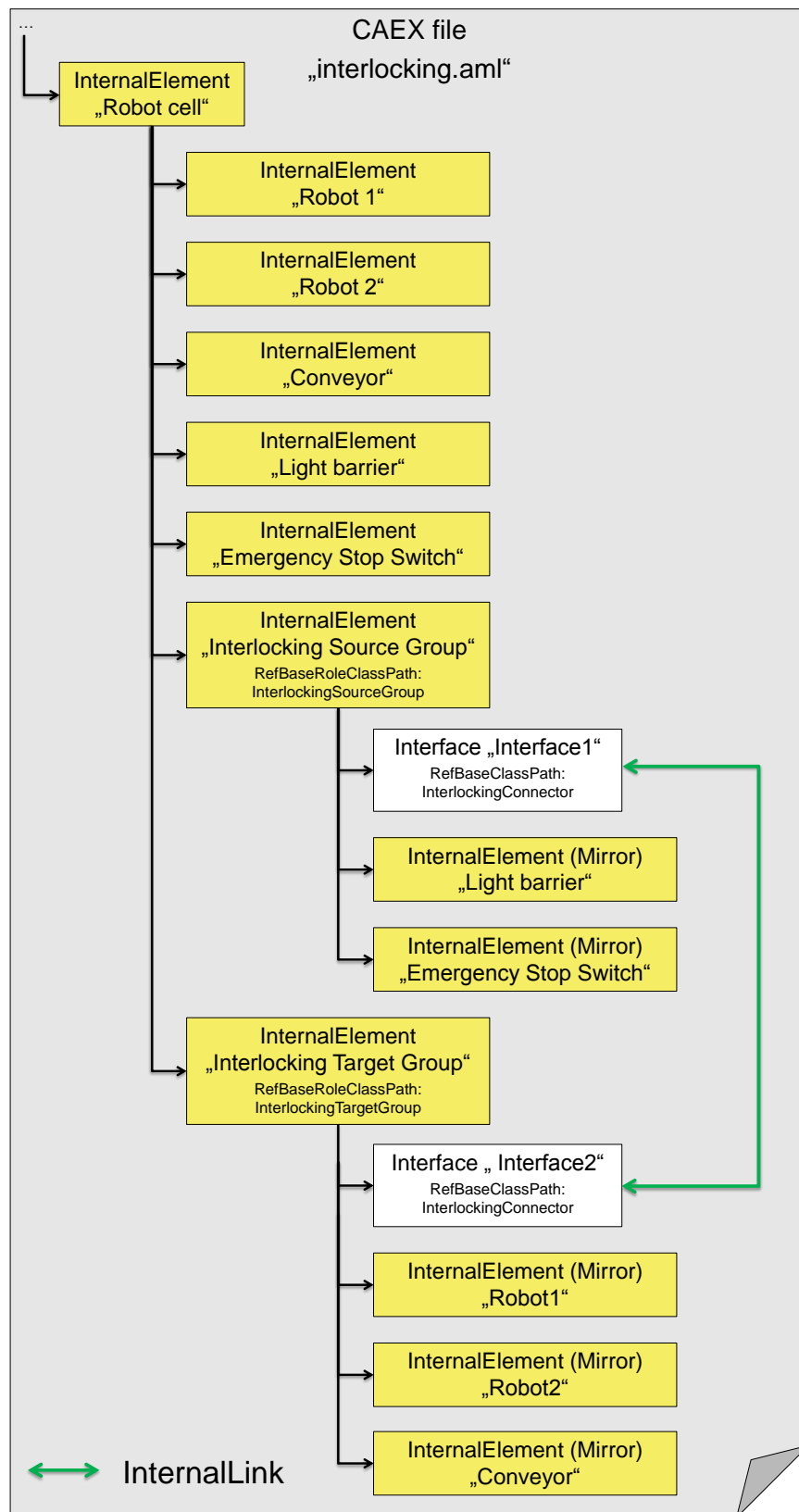


Figure F.3: Referencing interlocking information without interlocking condition

```

<InternalElement Name="Robot cell" ID="GUID200">
  <InternalElement Name="Robot 1" ID="GUID201"></InternalElement>
  <InternalElement Name="Robot 2" ID="GUID202"/>
  <InternalElement Name="Conveyor" ID="GUID203"/>
  <InternalElement Name="Light barrier" ID="GUID204"/>
  <InternalElement Name="Emergency Stop Switch" ID="GUID205"/>
  <InternalElement Name="Interlocking Source Group" ID="GUID206">
    <ExternalInterface Name="Interface1"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/InterlockingConnec
tor"/>
    <InternalElement Name="Light barrier" ID="GUID207"
RefBaseSystemUnitPath="GUID204"/>
    <InternalElement Name="Emergency Stop Switch" ID="GUID208"
RefBaseSystemUnitPath="GUID205"/>
    <RoleRequirements
RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/Interlockin
gSourceGroup"/>
  </InternalElement>
  <InternalElement Name="Interlocking Target Group" ID="GUID207">
    <ExternalInterface Name="Interface2"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/InterlockingConnec
tor"/>
    <InternalElement Name="Robot 1" ID="GUID208"
RefBaseSystemUnitPath="GUID201"></InternalElement>
    <InternalElement Name="Robot 2" ID="GUID209" RefBaseSystemUnitPath="GUID202"/>
    <InternalElement Name="Conveyor" ID="GUID210" RefBaseSystemUnitPath="GUID203"/>
    <RoleRequirements
RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/Interlockin
gTargetGroup"/>
  </InternalElement>
  <InternalLink Name="Link_InterlockingGroups" RefPartnerSideA="GUID206:Interface1"
RefPartnerSideB="GUID207:Interface2"/>
</InternalElement>

```

Figure F.4: XML text of the CAEX file for referencing interlocking information without interlocking condition

F.5 Referencing interlocking information with interlocking condition

On the second level of detail, a function describing the interlocking condition (modelled as FBD network) is associated to the interlocking source group. This level extends the first level of detail described in F.3.

Within the interlocking source group, two additional CAEX ExternalInterfaces are modelled: One with an AML InterfaceClass “InterlockingLogicInterface” following the rules of 6.2(if the POU is not referenced in the hierarchy before),and one CAEX ExternalInterface with an AML InterfaceClass “InterlockingVariableInterface”. This references the unique Boolean variable which describes the output respectively evaluation result of the interlocking information (see variable “z” in Figure F.5). And this result represents, whether the manufacturing system is in a safe state or not,indicated by the value of the attribute “SafeConditionEquals”.

The interlocking information also needs input variables of the objects of the interlocking source group respectively the safety devices (see variable “a” and “b” in Figure F.5 and Figure F.6). For this, “Light barrier” and “Emergency stop switch” also get two CAEX ExternalInterfaces of the AML InterfacesClasses “InterlockingLogicInterface” and “VariableInterface”.

To link the logical interface “VariableInterface” to the physical interface of the safety device a CAEX InternalLink is used. Figure F.7 and Figure F.8 depict this exemplarily for the light barrier.

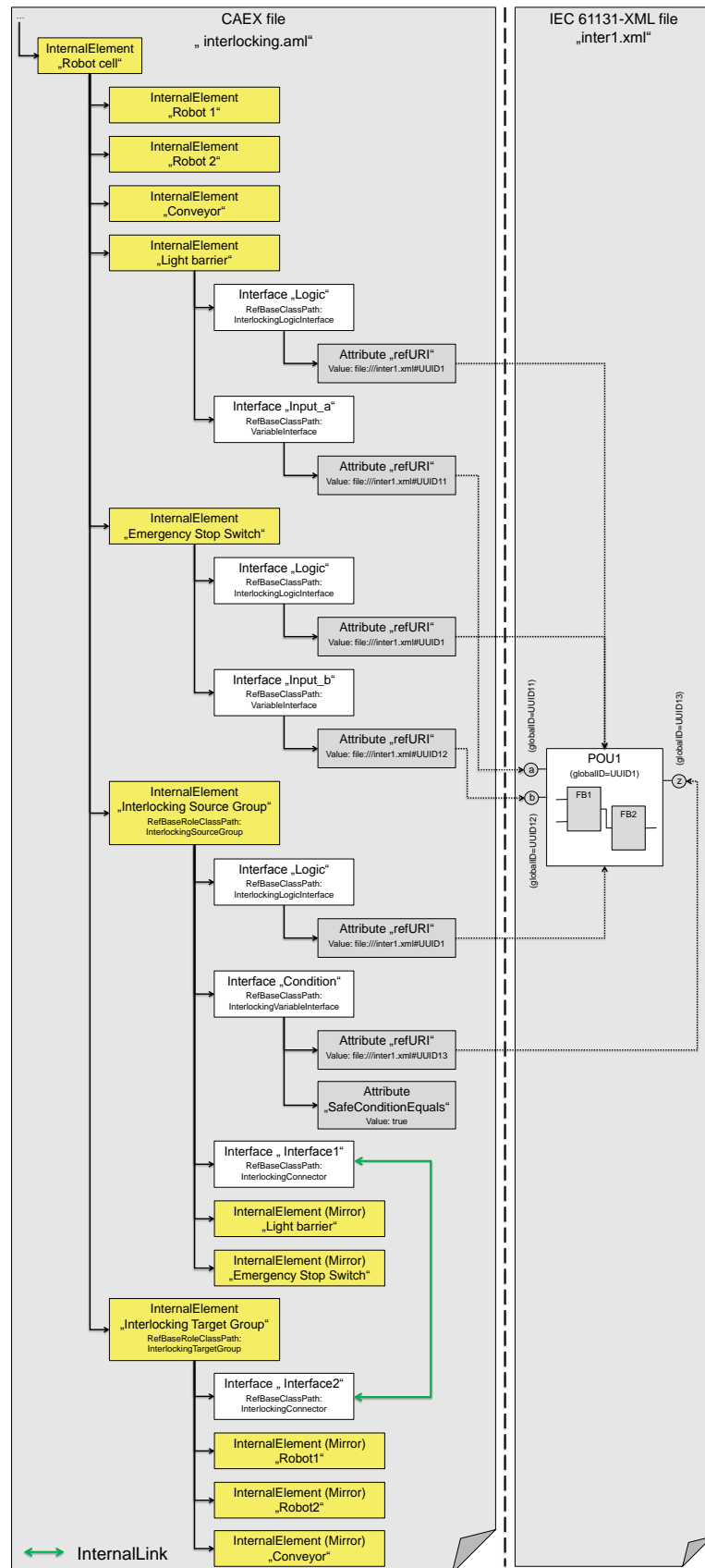


Figure F.5 – Referencing interlocking information with interlocking condition

```

<InternalElement Name="Robot cell" ID="GUID200">
  <InternalElement Name="Robot 1" ID="GUID201"></InternalElement>
  <InternalElement Name="Robot 2" ID="GUID202"/>
  <InternalElement Name="Conveyor" ID="GUID203"/>
  <InternalElement Name="Light barrier" ID="GUID204">
    <ExternalInterface
      Name="Logic"RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Input_a"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID11</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
  <InternalElement Name="Emergency Stop Switch" ID="GUID205">
    <ExternalInterface Name="Logic"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Input_b"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID12</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
  <InternalElement Name="Interlocking Source Group" ID="GUID206">
    <ExternalInterface Name="Logic"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID1</Value>
      </Attribute>
    </ExternalInterface>
    <ExternalInterface Name="Condition"
      RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface/InterlockingVariableInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI">
        <Value>file:///inter1.xml#UUID13</Value>
      </Attribute>
      <Attribute Name="SafeConditionEquals" AttributeDataType="xs:boolean">
        <DefaultValue>true</DefaultValue>
        <Value>true</Value>
      </Attribute>
    </ExternalInterface>
  </InternalElement>
  <ExternalInterface Name="Interface1"
    RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/InterlockingConnector"/>

```

```

<InternalElement Name="Light barrier" ID="GUID207"
RefBaseSystemUnitPath="GUID204"/>
  <InternalElement Name="Emergency Stop Switch" ID="GUID208"
RefBaseSystemUnitPath="GUID205"/>
    <RoleRequirements
RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/Interlockin
gSourceGroup"/>
    </InternalElement>
    <InternalElement Name="Interlocking Target Group" ID="GUID207">
      <ExternalInterface Name="Interface2"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/InterlockingConnec
tor"/>
      <InternalElement Name="Robot 1" ID="GUID208"
RefBaseSystemUnitPath="GUID201"/></InternalElement>
      <InternalElement Name="Robot 2" ID="GUID209" RefBaseSystemUnitPath="GUID202"/>
      <InternalElement Name="Conveyor" ID="GUID210" RefBaseSystemUnitPath="GUID203"/>
      <RoleRequirements
RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group/Interlockin
gTargetGroup"/>
      </InternalElement>
      <InternalLink Name="Link_InterlockingGroups" RefPartnerSideA="GUID206:Interface1"
RefPartnerSideB="GUID207:Interface2"/>
    </InternalElement>

```

Figure F.6 – XML text of the CAEX file for referencing interlocking information with interlocking condition

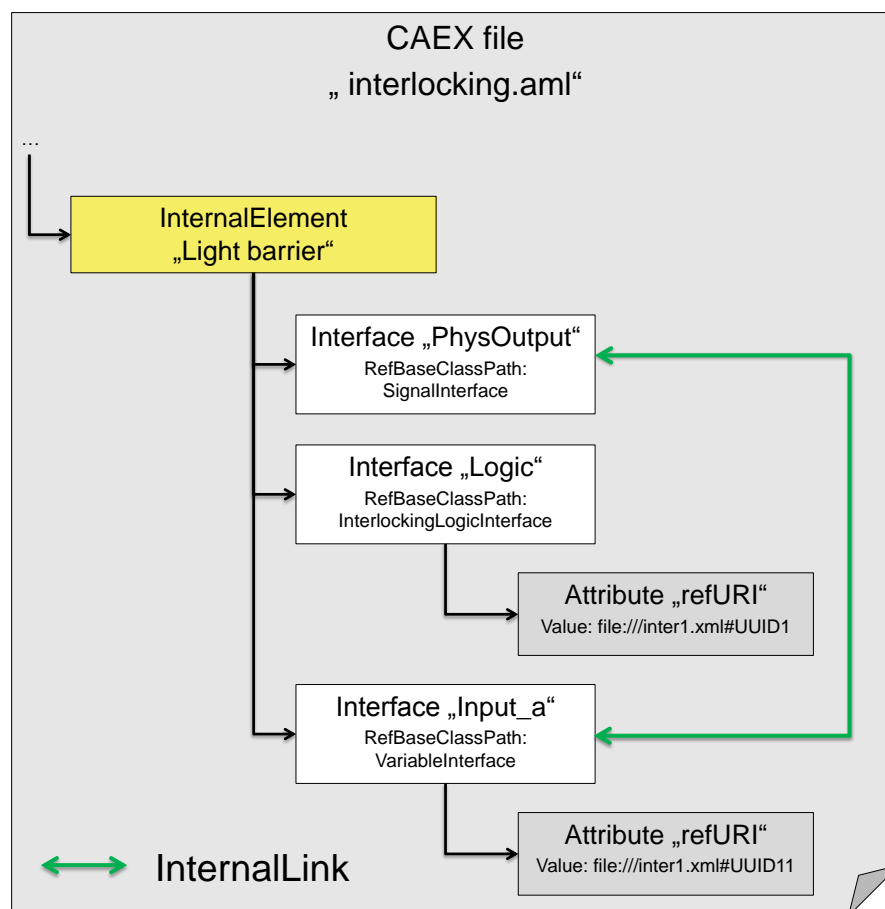


Figure F.7 – Linking logical interface with physical interface (extension to Figure F.5)

```

<InternalElement Name="Light barrier" ID="GUID204">
  <ExternalInterface Name="PhysOutput"
  RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface"/>
  <ExternalInterface Name="Logic"
  RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/LogicInterface/InterlockingLogicInterface">
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>file:///inter1.xml#UUID1</Value>
    </Attribute>
  </ExternalInterface>
  <ExternalInterface Name="Input_a"
  RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface">
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>file:///inter1.xml#UUID11</Value>
    </Attribute>
  </ExternalInterface>
  <InternalLink Name="Link1-LinkLogicPhysicalInterface" RefPartnerSideA="GUID204:PhysOutput"
  RefPartnerSideB="GUID204:Input_a"/>
</InternalElement>

```

Figure F.8 – XML text of the CAEX file for linking logical interface with physical interface (extension to Figure F.6)

In case the interlocking information is modelled as a POU network - which can either be stored in one IEC 61131-XML document or distributed throughout several IEC 61131-XML documents – the provisions of 6.2 apply.

Annex G Example for the mapping of logic models to IML

G.1 Mapping Examples of Gantt charts

The following examples show the usage of the mapping rules to transform Gantt charts to IML.

G.1.1 Mapping of activities without predecessor and successor relation

The following Gantt chart example has no predecessor and successor relations among the bars. It will be translated to a set of activities all being parallel within the resulting SFC. The temporal conditions will represent its sequence based on global timing.

Type	Graphical representation
Gantt Chart	
Graphical IML representation as SFC	

Table G.1: Mapping of the Gantt chart example “activities without predecessor and successor relations”

G.1.2 Mapping of an activity sequence

The following example depicts a Gantt chart with a predecessor/successor sequence among the bars. It will be translated to a sequence of states and activities within the resulting IML system. The temporal conditions represent its sequence based on local timing.

Type	Graphical representation
Gantt Chart	

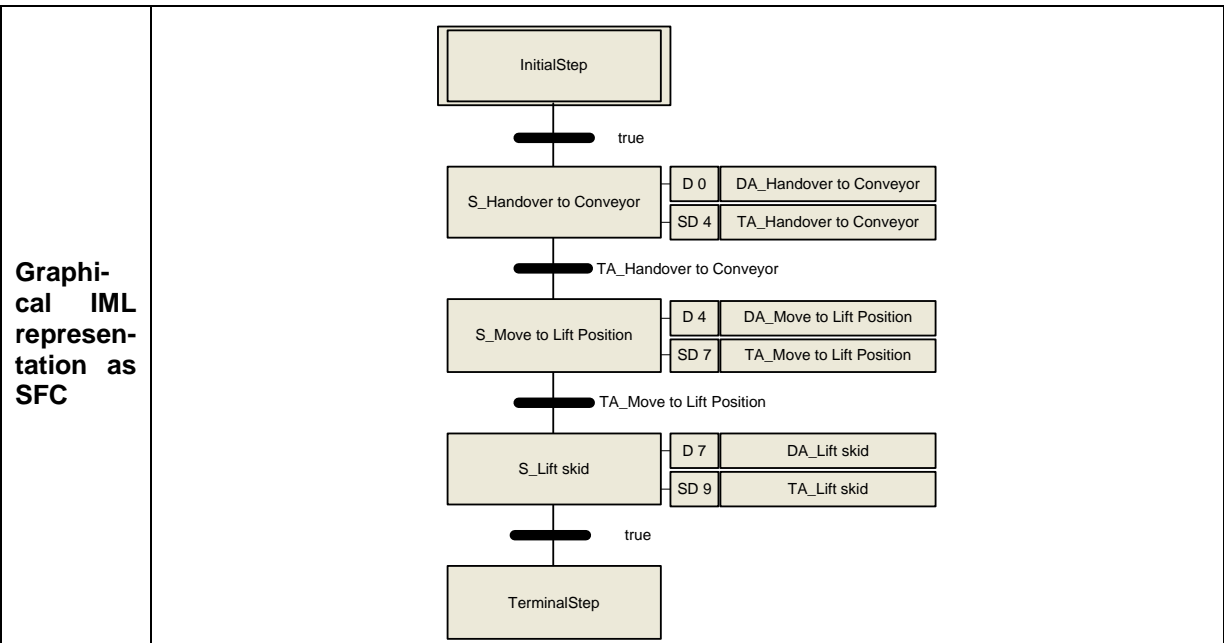


Table G.2: Mapping of the Gantt chart example “activity sequence”

G.1.3 Mapping of an activity sequence with divergences

In this example a simultaneous divergence is introduced to describe multiple successors of one bar in IML.

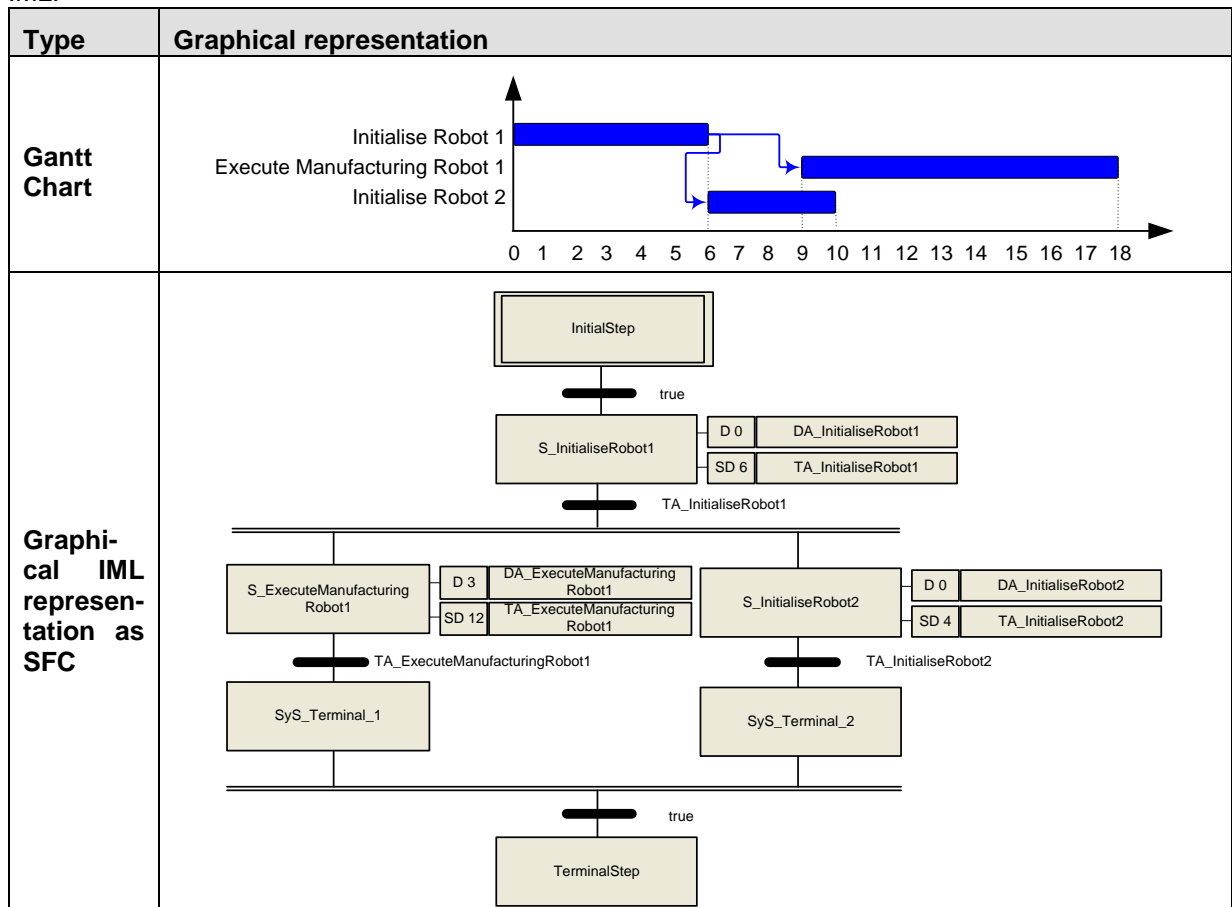


Table G.3: Mapping of the Gantt chart example “activity sequence with divergence”

G.1.4 Mapping of an activity sequence with convergences

The Gantt chart example includes a convergence with the predecessor sequence among the bars of the Gantt chart. This results in a simultaneous convergence in the IML.

Type	Graphical representation
Gantt Chart	
Graphical IML representation as SFC	

Table G.4: Mapping of the Gantt chart example “activity sequence with convergences”

G.2 Mapping examples of activity-on-node networks

The following examples show the usage of the mapping rules to transform activity-on-node networks to IML.

G.2.1 Mapping of an activity sequence

The following example depicts an activity-on-node network with a predecessor/successor sequence among the nodes. It will be translated to a sequence of states and activities within the resulting IML system. The temporal conditions represent its sequence based on local timing.

Type	Graphical representation
activity-on-node network	

Graphical IML representation as SFC

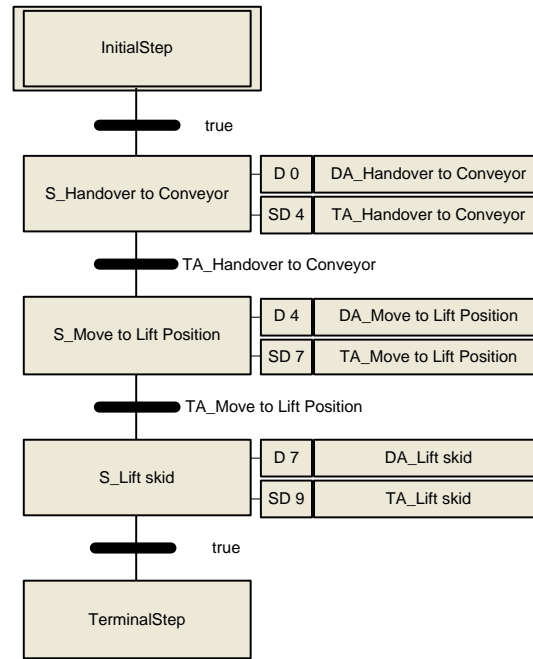


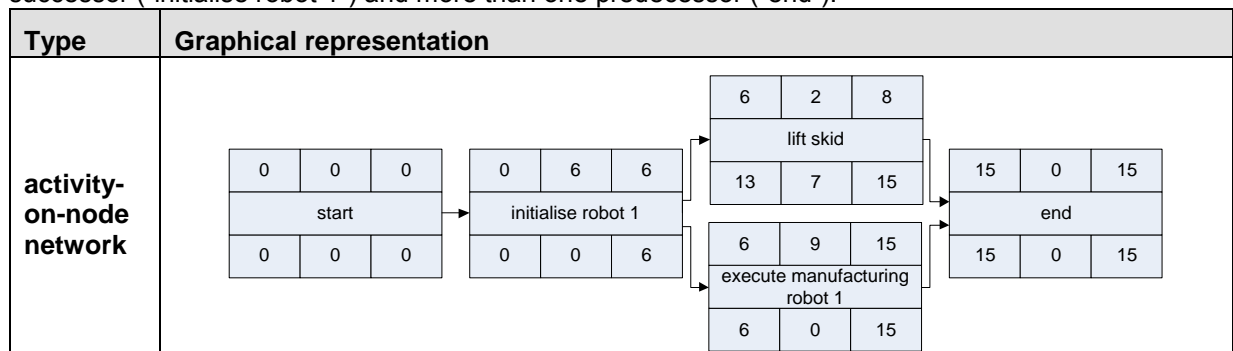
Table G.5: Mapping of the activity-on-node network example “activity sequence”

To achieve the transformation, the following elements are necessary:

- One state (named “InitialStep”) representing the initial node of the activity-on-node network followed by an initial state transition with condition “true”.
- One state transition for each arrow of the activity-on-node network.
- One state for each node of the activity-on-node network (“S_Handover to Conveyor”, “S_Move to Lift Position”, “S_Lift skid”) and two actions associated to it, including information about start and end points of the node.
- One state (named “TerminalStep”) representing the terminal node of the activity-on-node network.

G.2.2 Mapping of an activity sequence with divergences and convergences

The following example depicts an activity-on-node network with activities with more than one successor (“initialise robot 1”) and more than one predecessor (“end”).



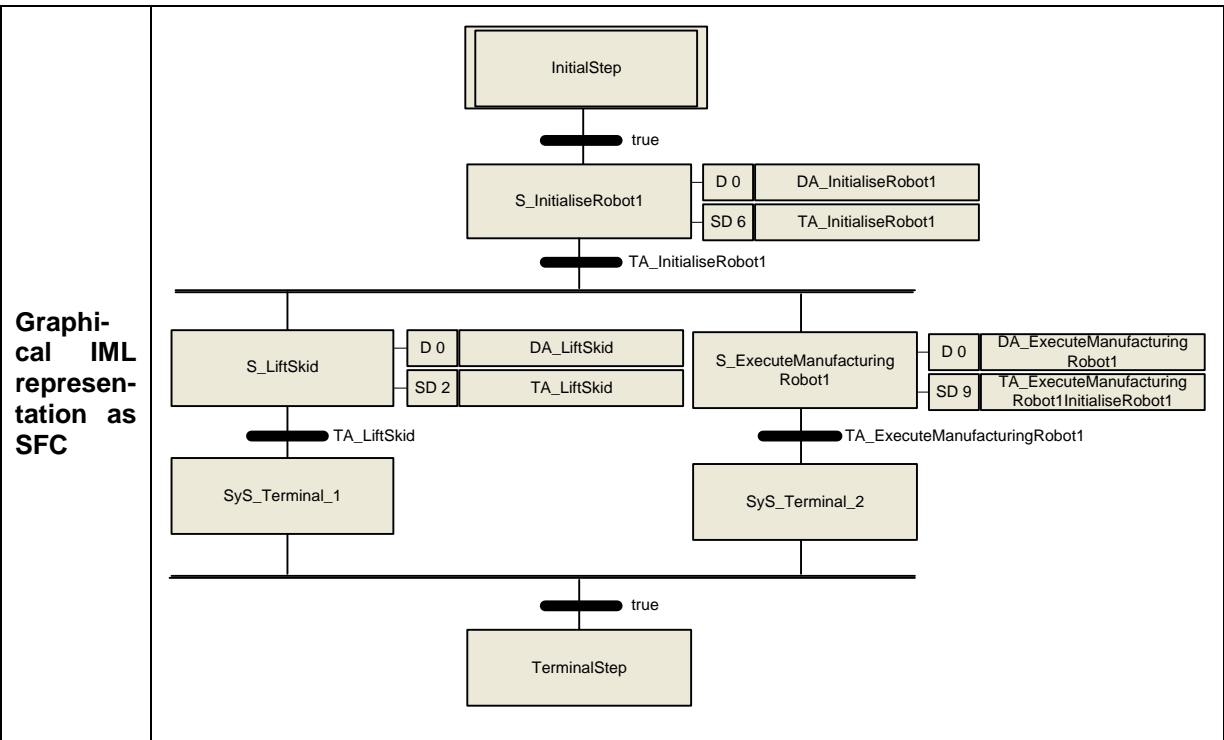


Table G.6: Mapping of the activity-on-node network example “activity sequence with divergences and convergences”

To achieve the transformation, the following elements are necessary:

- One state (named “InitialStep”) representing the initial node of the activity-on-node network followed by an initial state transition with condition “true”.
- One state transition for each arrow of the activity-on-node network.
- One simultaneous divergence for all synchronisation states.
- One state for each node of the activity-on-node network (“S_InitialiseRobot1”, “S_LiftSkid”, “S_ExecuteManufacturingRobot1”) and two actions associated to it, including information about start and end points of the node.
- One synchronization state for each predecessor state (named “SyS_Terminal_1” and “SyS_Terminal_2”).
- One simultaneous convergence for all synchronisation states.
- One state (named “TerminalStep”) representing the terminal node of the activity-on-node network.

G.2.3 Mapping of a complex activity sequence with divergences and convergences

Figure G.9 shows an example of an activity-on-node network. Predecessor and successor relations are depicted for example in “execute manufacturing robot 1”, which is a successor activity of “lift skid” as well as of “initialise robot 1” and a predecessor of the activities “postprocess robot 1” and “execute manufacturing robot 2”.

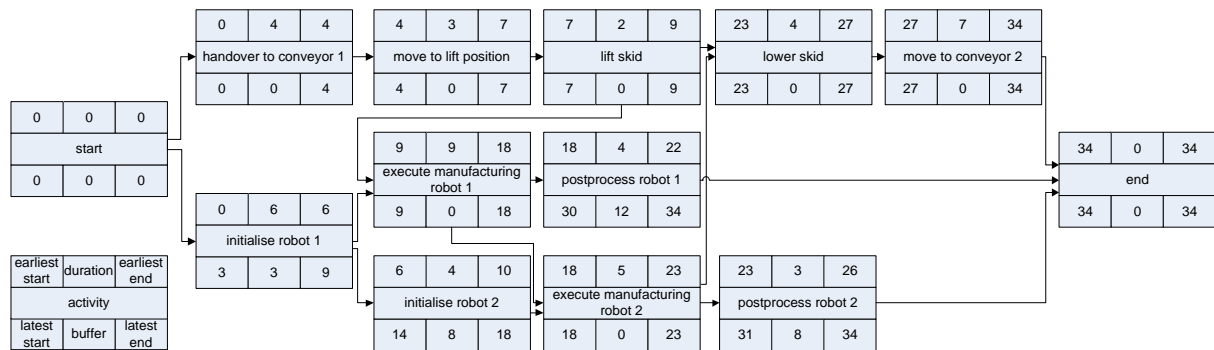


Figure G.9: Example of an activity-on-node network

The graphical representation of the resulting IML system of the transformation is depicted Figure G.10. The predecessor/successor relation between “execute manufacturing robot 1” and “lift skid” results in a predecessor/successor relation between “S_ExecuteManufacturingRobot1” and “S_Lift Skid” with an intermediate synchronization step to ensure the synchronization with the other predecessor of “execute manufacturing robot 1”. This additional step has no effect on the timing behaviour of the IML, besides that of storing the successor information of “execute manufacturing robot 1”. The additional steps “SyS_Terminal_1”, “SyS_Terminal_2”, and “SyS_Terminal_3” result from the definition of the terminal step.

AddData elements are not displayed in Figure G.10.

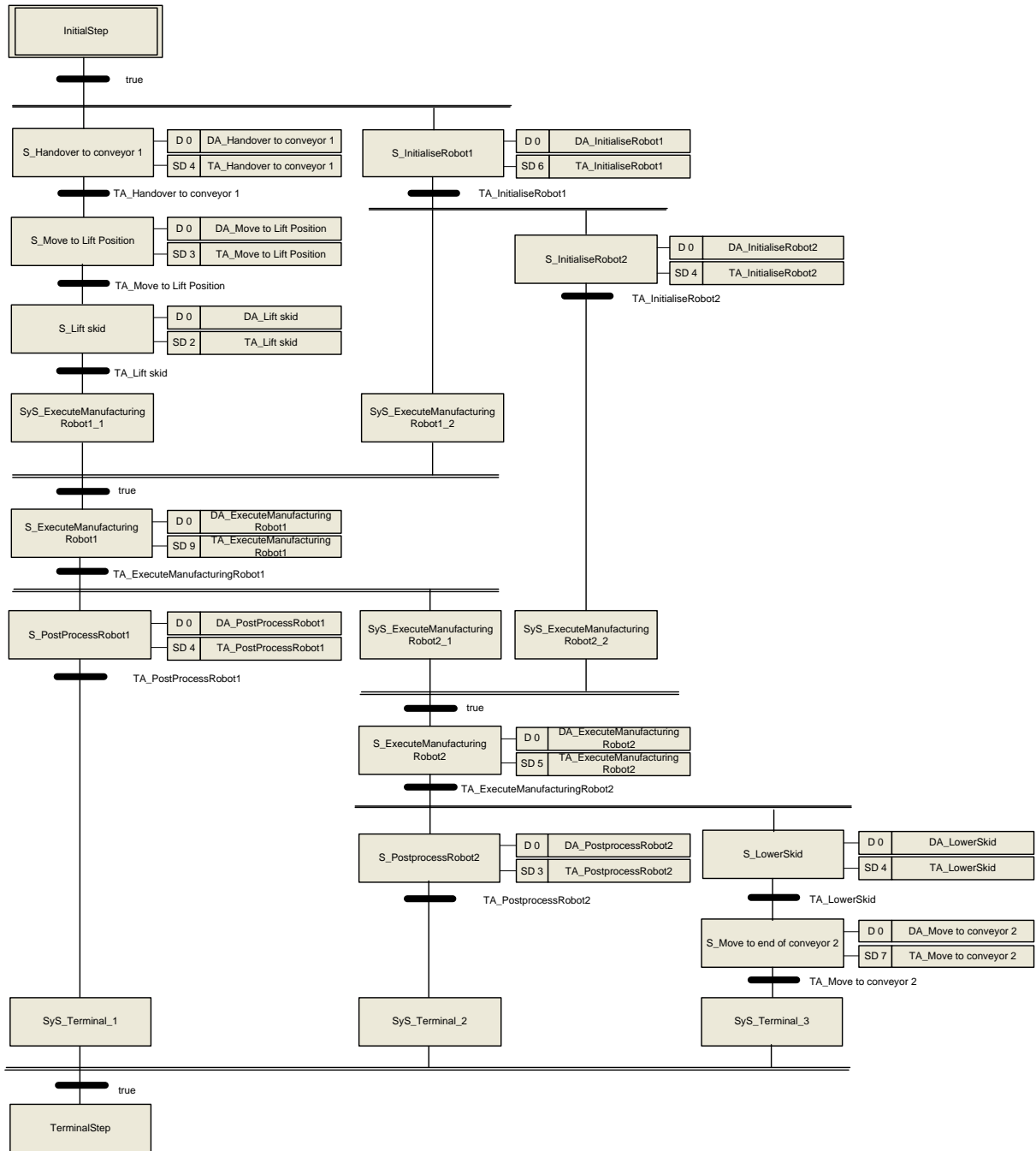


Figure G.10. Timing behaviour of the SFC derived from an activity-on-node network

G.3 Mapping examples of timing diagrams

The following examples show the use of the mapping rules to transform timing diagrams and parts of timing diagrams to SFCs.

G.3.1 Example transformation internal signal

The first example handles the transition from a state change to the subsequent state. To accomplish an executable SFC the additional signal “Signal_Motor1_1” has to be introduced, which is normally not displayed in the diagram, see Table G.7.

Type	Graphical representation
Timing diagram	<p>The timing diagram illustrates the transition from a state change to the subsequent state. It shows a Resource (Motor1) and a State (Fast_run, Slow_run) over time. A transition from Slow_run to Fast_run is marked with a delay of 1. A signal Signal_Motor1_1 is shown as a blue line that becomes true at the transition point. Motor1_0 and Motor1_1 are shown as dashed lines, and Motor1_2 is shown as a solid line.</p>
Graphical IML representation as SFC	<p>The graphical IML representation as SFC shows a sequence of states: Init, Time_Step_0, Time_Step_1, Motor1_0, Motor1_1, and Motor1_2. Transitions are labeled with conditions like [Signal_Time_0 = true] and [Signal_Motor1_1 = true]. Actions are labeled with D0 and SD1.</p>

Table G.7: Mapping of the timing diagram example “transition from a state change to the subsequent state”

To achieve the transformation, the following elements are necessary:

- One activity representing the additional signal with the name “Signal_Motor1_1”. This action has got a delay of “1” for the duration of the state change from “Slow_run” to “Fast_run”.
- One transition that is activated by the signal as successor for the state representing the resource state change from “Slow_run” to “Fast_run”, i. e. when “Signal_Motor1_1” becomes “true”.
- One new SFC state with an associated action, indicating the new status of the resource. In this case the resource has entered its resource state “Fast_run”.

G.3.2 Example External Signals

The next example handles two external signals, which are fired with a delay of three seconds.

Type	Graphical representation
Timing diagram	<p>The timing diagram shows a resource 'Motor1' with two states: 'Fast_run' and 'Slow_run'. The timeline is marked with time steps 0, 1, 2, and 3. At step 0, 'Signal_Time_0' occurs, triggering a transition from 'Slow_run' to 'Fast_run'. At step 3, 'Signal_Time_1' occurs, triggering a transition from 'Fast_run' back to 'Slow_run'. A box labeled 'Uses Signal_Motor1_1' points to the state at step 3. Below the timeline, four boxes labeled 'Motor1_0', 'Motor1_1', 'Motor1_2', and 'Motor1_3' are shown, corresponding to the time steps.</p>
Graphical IML representation as SFC	<p>The SFC diagram illustrates the logic for the Motor1 resource. It begins with an 'INIT' state leading to 'Time_Step_0'. 'Time_Step_0' has a transition to 'Time_Step_1' triggered by the event '[Signal_Time_0 = true]'. 'Time_Step_1' has a transition to 'Motor1_0' triggered by the event '[Signal_Time_1 = true]'. 'Motor1_0' has a transition to 'Motor1_1' triggered by the event '[Signal_Time_0 = true]'. 'Motor1_1' has a transition to 'Motor1_2' triggered by the event '[Signal_Motor1_1 = true]'. 'Motor1_2' has a transition to 'Motor1_3' triggered by the event '[Signal_Time_1 = true]'. Each state has associated actions: 'Motor1_0' (D 0 Motor1_Slow_run), 'Motor1_1' (D 0 Motor1_Slow_run_to_Fast_run, SD 1 Signal_Motor1_1), 'Motor1_2' (D 0 Motor1_Fast_run), and 'Motor1_3' (D 0 Motor1_Fast_run_to_Slow_run).</p>

Table G.8: Mapping of the timing diagram example “two external signals fired with delay of three seconds”

For this example the following SFC elements are needed:

- One activity for each external signal within the timeline representing the signal with the names “Signal_Time_0” and “Signal_Time_1”. The first action has got a delay of “0” and the second of “3” representing the delay between both signals.
- A transition after each of the two states associated to the signals within the timeline. These transitions are activated by the corresponding time signal.
- One transition as predecessor for each state representing the state change within a resource state flow that is triggered by the time signals. These transitions are activated by the corresponding time signal.
- One new SFC state for each signal within the resource state flow with the names “Motor1_1” and “Motor1_3”.
- The associated actions that represent the new status of the resource, in this example the state change from “Slow_run” to “Fast_run” for the first signal and the state change form “Fast_run” to “Slow_run” for the second signal with the names “Motor1_Fast_run_to_Slow_run” and “Motor1_Slow_run_to_Fast_run”.

G.3.3 Example Signal Between two Resource States Flows

The last example handles the transformation of a signal fired by one resource state and consumed by another.

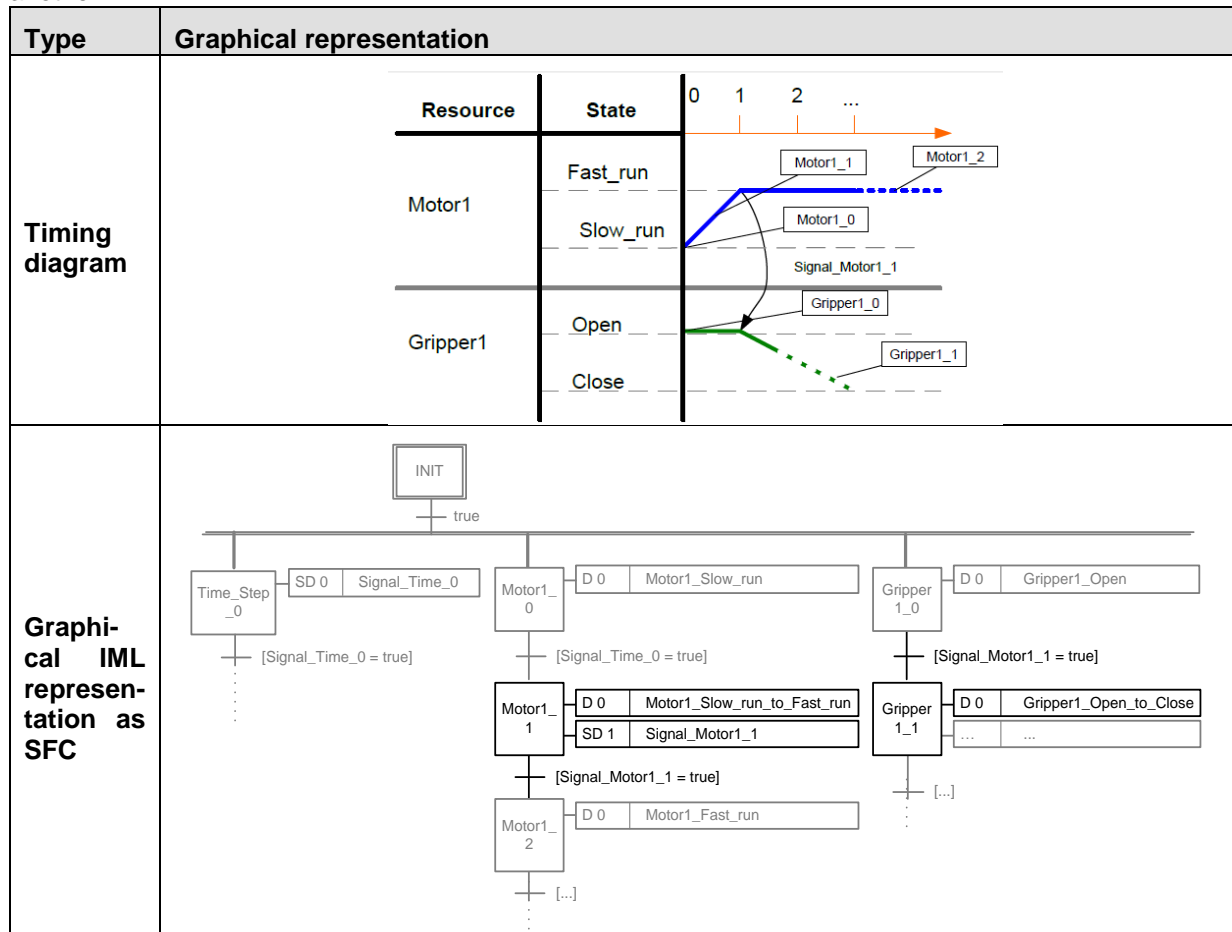


Table G.9: Mapping of the timing diagram example “signal fired by one resource state and consumed by another”

For this example the following SFC elements are needed:

- One activity for the signal within the firing resource state flow for “Motor1” with the name “Signal_Motor1_1”.
- A transition after the state associated to the signal within the resource state flow for “Motor1”. This transition is activated by the signal “Signal_Motor1_1”.
- One transition as predecessor for the state representing the state change from “Open” to “Close” within a resource state flow for “Gripper1”. The transition is activated by the signal “Signal_Motor1_1”.
- One new SFC state within the resource state flow of “Gripper1” with the name “Gripper1_1”.
- The associated action that represents the new status of the resource, in this example the state change from “Open” to “Close” with the name “Gripper1_Open_to_Close”.

G.4 Mapping examples of state charts

The following examples show the usage of the mapping rules to transform state charts to SFCs.

G.4.1 Mapping of a simple cyclic state chart

The first example represents the cyclic execution of a PLC program, see Table G.10.

Type	Graphical representation
state chart	<pre> graph TD Start(()) --> ReadInput([ReadInput]) ReadInput -- "[Inputs read]" --> ExecuteProgram([ExecuteProgram]) ExecuteProgram -- "[Program executed]" --> SetOutputs([SetOutputs]) SetOutputs -- "[Outputs set]" --> ReadInput </pre>
Graphical IML representation as SFC	<pre> graph TD InitialStep_noName[InitialStep_noName] -- "[TRUE]" --> State_ReadInput[State_ReadInput] State_ReadInput -- "[Input read]" --> State_ExecuteProgram[State_ExecuteProgram] State_ExecuteProgram -- "[Program executed]" --> State_SetOutputs[State_SetOutputs] State_SetOutputs -- "[Outputs set]" --> State_ReadInput </pre>

Table G.10: Mapping of the state chart example “simple cyclic state chart”

To achieve the transformation, the following elements are necessary:

- One state (named “InitialStep_noName”) representing the initial state of the state chart followed by an initial state transition with condition “true”.
- One state transition for each state transition of the state chart (“Inputs read”, “Program executed” and “Outputs set”).
- One state for each state of the state chart (“State_ReadInput”, “State_ExecuteProgram”, “State_SetOutputs”).
- One convergence between the initial state transition, the state transition for “Outputs set” and the state “State_ReadInput”, that represents the closing of the cycle.

G.4.2 Mapping of a state chart with states with different predecessors and successors

The second example enriches the first example by additional elements including interrupt handling, see Table G.11.

Type	Graphical representation
state chart	<pre> stateDiagram-v2 [*] --> ReadInput ReadInput --> ExecuteProgram : [Inputs read] ExecuteProgram --> SetOutputs : [Program executed] SetOutputs --> ReadInput : [Outputs set] ExecuteProgram --> InterruptHandling : [Interrupt] InterruptHandling --> ExecuteProgram : [Interrupt handled] </pre>
Graphical IML representation as SFC	<pre> stateDiagram-v2 state InitialStep_noName as InitialStep_noName state State_ReadInput as State_ReadInput state State_ExecuteProgram as State_ExecuteProgram state State_SetOutputs as State_SetOutputs state State_InterruptHandling as State_InterruptHandling InitialStep_noName --> State_ReadInput : [TRUE] State_ReadInput --> State_ExecuteProgram : [Input read] State_ExecuteProgram --> State_SetOutputs : [Program executed] State_ExecuteProgram --> State_InterruptHandling : [Interrupt] State_SetOutputs --> State_ReadInput : [Outputs set] State_InterruptHandling --> State_ExecuteProgram : [Interrupt handled] </pre>

Table G.11: Mapping of the state chart example “states with different predecessors and successors”

To achieve the transformation, the following elements are necessary in addition to G.4.1:

- One state (named “State_InterruptHandling”) representing the state chart state “InterruptHandling”.
- Two state transitions representing the state chart state transitions “Interrupt handled” and “Interrupt”.
- One convergence between the state transitions for “Input read”, “Interrupt handled” and the state “State_ExecuteProgram”, which represents the predecessor and successor relations of the state and the state transitions.
- One divergence between the state “State_ExecuteProgram” and the state transitions for “Program executed” and “Interrupt”, which represents the predecessor and successor relations of the state and the state transitions.

G.4.3 Mapping of a state chart with actions

The third example enriches the first example by an action associated to the state transition “Inputs read” and an action within the state “ExecuteProgram”, see Table G.12.

Type	Graphical representation
state chart	<pre> stateDiagram-v2 [*] --> ReadInput ReadInput --> SetOutputs : [Outputs set] SetOutputs --> ExecuteProgram : [Program executed] state ExecuteProgram { DoAction IncrementLineCount } ExecuteProgram --> ReadInput : [Inputs read] Note over ExecuteProgram, ReadInput: Action SetLineCountzero </pre>
Graphical IML representation as SFC	<pre> sequenceDiagram participant Start as InitialStep_noName participant Read as State_ReadInput participant Set as StateForActivity_SetLineCountzero participant Exec as State_ExecuteProgram participant Out as State_SetOutputs Start --> Read : [true] Read --> Set : [Input read] Set --> Exec : [true] Exec --> Out : [Program executed] Out --> Read : [Outputs set] </pre>

Table G.12: Mapping of the state chart example “state chart with actions”

To achieve the transformation, the following elements are necessary in addition to G.4.1:

- One state (named “StateForActivity_SetLineCountzero”) with an associated action (named “Action_SetLineCountzero_ofStateTransition”), representing the state chart action “SetLineCountzero” at the state transition “Input read”.
- One state (named “State_ExecuteProgram”) with an associated action (named “Action_IncrementLineCount_ofState_ExecuteProgram”), representing the action “IncrementLineCount” within the state chart state “ExecuteProgram”.

G.4.4 Mapping of a state chart with a condition connector

The fourth example enriches the first example by a condition connector and an interrupt handling, which is executed, when a certain condition is reached, see Table G.11.

Type	Graphical representation
state chart	<pre> graph TD Start(()) --> ReadInput([ReadInput]) ReadInput -- "[Inputs read]" --> C((C)) C -- "[NoInterrupt]" --> ExecuteProgram([ExecuteProgram]) ExecuteProgram -- "[Program executed]" --> SetOutputs([SetOutputs]) SetOutputs -- "[Outputs set]" --> ReadInput C -- "[Interrupt]" --> InterruptHandling([InterruptHandling]) InterruptHandling -- "[Interrupt handled]" --> ExecuteProgram </pre>
Graphical IML representation as SFC	<pre> graph TD Init[InitialStep_noName] -- "[true]" --> Read[State_ReadInput] Read -- "[Input read]" --> Cond[Condition_C] Cond -- "[NoInterrupt]" --> Exec[State_ExecuteProgram] Cond -- "[Interrupt]" --> Int[State_InterruptHandling] Int -- "[InterruptHandled]" --> Exec Exec -- "[Program executed]" --> Set[State_SetOutputs] Set -- "[Outputs set]" --> Read </pre>

Table G.13: Mapping of the state chart example “simple cyclic state chart”

To achieve the transformation, the following elements are necessary in addition to G.4.1:

- One state (named “Condition_C”) followed by a divergence, representing the condition connector and its successor relations.
- One state transition (named “NoInterrupt”) representing the state transition for the branch with no interrupt after reading the inputs.
- One state transition for “Interrupt” followed by a state (named “State_InterruptHandling”) and a state transition for “InterruptHandled”, representing the state transition for the branch with interrupts after reading the inputs, the state chart state “InterruptHandling” and the state transition “Interrupt handled”.
- One convergence after the state transitions for “NoInterrupt” and “InterruptHandled”, representing the convergence of the two interrupt branches.

G.4.5 Mapping of a state chart with simple hierarchy

In the fifth example the state “CyclicBehaviour” is refined hierarchically by composite states, the states of the first example, see Table G.14.

Type	Graphical representation
state chart	
Graphical IML representation as SFC	

Table G.14: Mapping of the state chart example “simple hierarchy”

To achieve the transformation, the following elements are necessary in addition to G.4.1:

- One initial state (named “InitialStep_noName”) followed by an initial state condition with condition “true”, representing the initial state and its state condition of the higher level state chart.
- One state (named “State_Programming”), representing the state “Programming” of the higher level state chart.
- Two state transitions for “Start” and “Stop”, representing the deep transitions from the higher level state chart to the sub state chart.
- One state (named “State_CyclicBehaviour”), which contains an addData element with a reference to the sub state chart, representing the state “CyclicBehaviour”.

G.4.6 Mapping of a state chart with a complex hierarchy and connectors

The sixth example enriches the fifth example by a sub state chart for the state “Programming”, which refines the state hierarchically by composite states and includes a condition connector and a historical connector, see Table G.15.

Type	Graphical representation
state chart	<pre> stateDiagram-v2 [*] --> Programming state Programming { [*] --> EditProgram : [Program change necessary] EditProgram --> UploadProgram : [Editing finished] UploadProgram --> EditProgram : [Program incorrect] UploadProgram --> CompileProgram : [Program correct] CompileProgram --> H : [Program compiled] state H } state CyclicBehaviour { [*] --> ReadInput : [Outputs set] ReadInput --> ExecuteProgram : [Inputs read] ExecuteProgram --> SetOutputs : [Program executed] SetOutputs --> ReadInput } Programming --> CyclicBehaviour : [Start] CyclicBehaviour --> Programming : [Stop] </pre>

Graphical IML representation as SFC

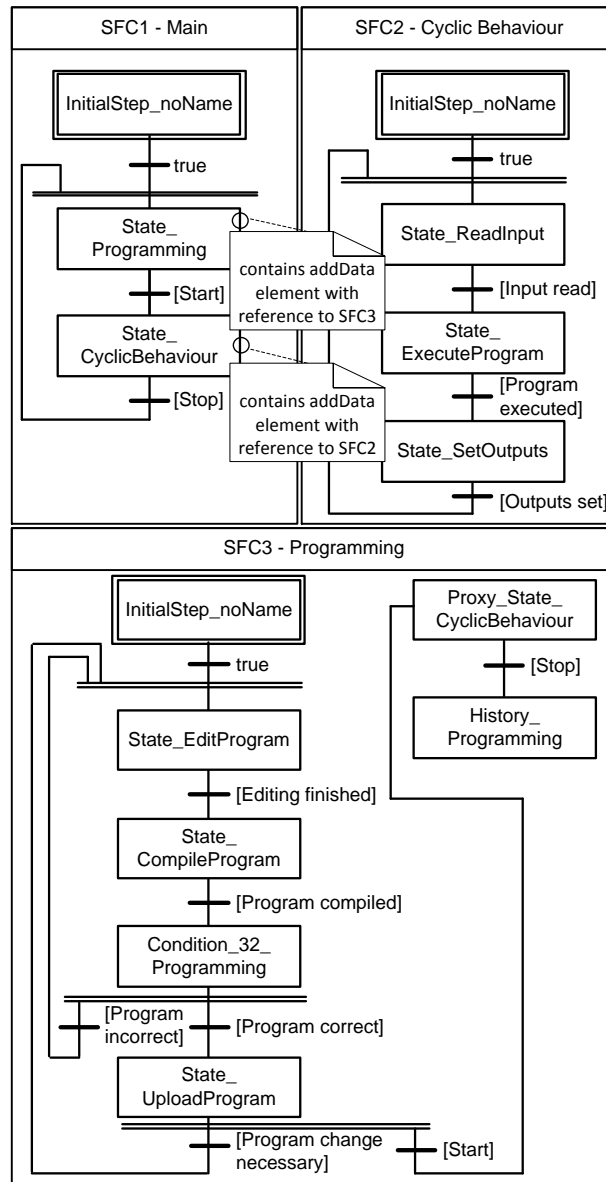


Table G.15: Mapping of the state chart example “complex hierarchy and connectors”

To achieve the transformation, the following elements are necessary in addition to G.4.5:

- One additional sub state chart for the state “Programming” with the following elements:
 - One initial state (named “InitialStep_noName”) followed by an initial state condition with condition “true”, representing the initial state and its state condition.
 - One state for each state of the sub state chart (named “State_EditProgram”, “State_CompileProgram”, “State_UploadProgram”).
 - One state transition for each state transition of the sub state chart for “Editing finished”, “Program compiled”, “Program incorrect”, “Program correct”, “Program change necessary”, “Start” and “Stop”.
 - One state (named “Condition_32_Programming”) followed by a divergence, representing the condition connector and its successor relations.
 - One state (named “History_Programming”), representing the history connector.

- One divergence between the state “State_UploadProgram” and the state transitions for “Program change necessary” and “Start”, representing the predecessor and successor relations of the state and the state transitions.
- One state (named “Proxy_State_CyclicBehaviour”) representing the inter level transition from the sub state chart “Programming” to the state “State_CyclicBehaviour” of the higher level state chart.
- One state (named “State_Programming”), which contains an addData element with a reference to the sub state chart, representing the state “Programming”.

Annex H XML representation of AML libraries

H.1 AutomationMLBaseRoleClassLib

```

<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
FileName="AutomationMLBaseRoleClassLib.aml" SchemaVersion="2.15">
  <AdditionalInformation AutomationMLVersion="2.0"/>
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>AutomationML e.V.</WriterName>
      <WriterID>AutomationML e.V.</WriterID>
      <WriterVendor>AutomationML e.V.</WriterVendor>
      <WriterVendorURL>www.automationml.org</WriterVendorURL>
      <WriterVersion>1.0</WriterVersion>
      <WriterRelease>1.0.0</WriterRelease>
      <LastWritingDateTime>2013-03-01</LastWritingDateTime>
      <WriterProjectTitle>Automation Markup Language Standard
Libraries</WriterProjectTitle>
      <WriterProjectID>Automation Markup Language Standard Libraries</WriterProjectID>
    </WriterHeader>
  </AdditionalInformation>
  <ExternalReference Path="AutomationMLInterfaceClassLib.aml"
Alias="AutomationMLInterfaceClassLib"/>
  <RoleClassLib Name="AutomationMLBaseRoleClassLib">
    <Description>Automation Markup Language Base Role Class Library - Part 1 Content
extended with Part 3 and Part 4 Content</Description>
    <Version>2.2.2</Version>
    <RoleClass Name="AutomationMLBaseRole">
      <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
        <Attribute Name="AssociatedFacet" AttributeDataType="xs:string"/>
        <RoleClass Name="InterlockingSourceGroup" RefBaseClassPath="Group"/>
        <RoleClass Name="InterlockingTargetGroup" RefBaseClassPath="Group"/>
      </RoleClass>
      <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="Port" RefBaseClassPath="AutomationMLBaseRole">
        <Attribute Name="Direction" AttributeDataType="xs:string"/>
        <Attribute Name="Cardinality">
          <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
          <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
        </Attribute>
        <Attribute Name="Category" AttributeDataType="xs:string"/>
        <ExternalInterface Name="ConnectionPoint" ID="9942bd9c-c19d-44e4-a197-
11b9edf264e7"
RefBaseClassPath="AutomationMLInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationM
LBaseInterface/PortConnector"/>
      </RoleClass>
      <RoleClass Name="Resource" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="Product" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="Process" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="Structure" RefBaseClassPath="AutomationMLBaseRole">
        <RoleClass Name="ProductStructure" RefBaseClassPath="Structure"/>
        <RoleClass Name="ProcessStructure" RefBaseClassPath="Structure"/>
        <RoleClass Name="ResourceStructure" RefBaseClassPath="Structure"/>
      </RoleClass>
      <RoleClass Name="PropertySet" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="Frame" RefBaseClassPath="AutomationMLBaseRole"/>
      <RoleClass Name="LogicObject" RefBaseClassPath="AutomationMLBaseRole"/>
    </RoleClass>
  </RoleClassLib>

```

```

</RoleClass>
</RoleClassLib>
</CAEXFile>

```

H.2 AutomationMLInterfaceClassLib

```

<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
FileName="AutomationMLInterfaceClassLib.aml" SchemaVersion="2.15">
  <AdditionalInformation AutomationMLVersion="2.0"/>
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>AutomationML e.V.</WriterName>
      <WriterID>AutomationML e.V.</WriterID>
      <WriterVendor>AutomationML e.V.</WriterVendor>
      <WriterVendorURL>www.automationml.org</WriterVendorURL>
      <WriterVersion>1.0</WriterVersion>
      <WriterRelease>1.0.0</WriterRelease>
      <LastWritingDateTime>2013-03-01</LastWritingDateTime>
      <WriterProjectTitle>Automation Markup Language Standard
Libraries</WriterProjectTitle>
      <WriterProjectID>Automation Markup Language Standard Libraries</WriterProjectID>
    </WriterHeader>
  </AdditionalInformation>
  <InterfaceClassLib Name="AutomationMLInterfaceClassLib">
    <Description>Standard Automation Markup Language Interface Class Library - Part 1
Content extended with Part 3 and Part 4 Content</Description>
    <Version>2.2.2</Version>
    <InterfaceClass Name="AutomationMLBaseInterface">
      <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
        <Attribute Name="Direction" AttributeDataType="xs:string"/>
      </InterfaceClass>
      <InterfaceClass Name="PortConnector"
RefBaseClassPath="AutomationMLBaseInterface"/>
      <InterfaceClass Name="InterlockingConnector"
RefBaseClassPath="AutomationMLBaseInterface"/>
      <InterfaceClass Name="PPRConnector"
RefBaseClassPath="AutomationMLBaseInterface"/>
      <InterfaceClass Name="ExternalDataConnector"
RefBaseClassPath="AutomationMLBaseInterface">
        <Attribute Name="refURI" AttributeDataType="xs:anyURI"/>
      <InterfaceClass Name="COLLADAInterface"
RefBaseClassPath="ExternalDataConnector">
        <Attribute Name="refType" AttributeDataType="xs:string"/>
        <Attribute Name="target" AttributeDataType="xs:token"/>
      </InterfaceClass>
      <InterfaceClass Name="PLCopenXMLInterface"
RefBaseClassPath="ExternalDataConnector">
        <InterfaceClass Name="LogicInterface"
RefBaseClassPath="PLCopenXMLInterface">
          <InterfaceClass Name="SequencingLogicInterface"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface"/>
          <InterfaceClass Name="BehaviourLogicInterface"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface"/>
          <InterfaceClass Name="SequencingBehaviourLogicInterface"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn

```

```

ector/PLCopenXMLInterface/LogicInterface"/>
    <InterfaceClass Name="InterlockingLogicInterface"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/LogicInterface"/>
    </InterfaceClass>
    <InterfaceClass Name="LogicObjectInterface"
RefBaseClassPath="PLCopenXMLInterface"/>
    <InterfaceClass Name="VariableInterface"
RefBaseClassPath="PLCopenXMLInterface"/>
    <InterfaceClass Name="InterlockingVariableInterface"
RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConn
ector/PLCopenXMLInterface/VariableInterface"/>
    <Attribute Name="SafeConditionEquals"
AttributeDataType="xs:boolean">
        <DefaultValue>true</DefaultValue>
    </Attribute>
</InterfaceClass>
</InterfaceClass>
</InterfaceClass>
</InterfaceClass>
<InterfaceClass Name="Communication"
RefBaseClassPath="AutomationMLBaseInterface">
    <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
</InterfaceClass>
    <InterfaceClass Name="AttachmentInterface"
RefBaseClassPath="AutomationMLBaseInterface"/>
</InterfaceClass>
</InterfaceClassLib>
</CAEXFile>

```

Annex I XML representation of schemata

I.1 AML_addData

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:aml="http://www.AutomationML.org/AutomationML_PLCOpen"
  targetNamespace="http://www.AutomationML.org/AutomationML_PLCOpen"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:complexType name="addDataBaseObject">
    <xs:attribute name="ID" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:element name="AML">
    <xs:annotation>
      <xs:documentation>It comprises all of the additional, AML specific logic
information.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Time" minOccurs="0">
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="aml:addDataBaseObject">
                <xs:sequence>
                  <xs:element name="Duration" minOccurs="0">
                    <xs:complexType>
                      <xs:complexContent>
                        <xs:extension base="aml:addDataBaseObject">
                          <xs:attribute name="Value"
type="xs:decimal" use="required"/>
                          <xs:attribute name="Unit" type="xs:string"
use="optional"/>
                        </xs:extension>
                      </xs:complexContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="EarliestStart" minOccurs="0">
                    <xs:complexType>
                      <xs:complexContent>
                        <xs:extension base="aml:addDataBaseObject">
                          <xs:attribute name="Value"
type="xs:decimal" use="required"/>
                          <xs:attribute name="Unit" type="xs:string"
use="optional"/>
                        </xs:extension>
                      </xs:complexContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="LatestStart" minOccurs="0">
                    <xs:complexType>
                      <xs:complexContent>
                        <xs:extension base="aml:addDataBaseObject">
                          <xs:attribute name="Value"
type="xs:decimal" use="required"/>
                          <xs:attribute name="Unit" type="xs:string"
use="optional"/>
                        </xs:extension>
                      </xs:complexContent>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

</xs:element>
<xs:element name="EarliestEnd" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="aml:addDataBaseObject">
        <xs:attribute name="Value"
type="xs:decimal" use="required"/>
        <xs:attribute name="Unit" type="xs:string"
use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="LatestEnd" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="aml:addDataBaseObject">
        <xs:attribute name="Value"
type="xs:decimal" use="required"/>
        <xs:attribute name="Unit" type="xs:string"
use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="Delay" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="aml:addDataBaseObject">
        <xs:attribute name="Value"
type="xs:decimal" use="required"/>
        <xs:attribute name="Unit" type="xs:string"
use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="ChartType" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="aml:addDataBaseObject">
        <xs:attribute name="ChartType" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="StateChart"/>
              <xs:enumeration value="TimingDiagram"/>
              <xs:enumeration value="GanttChart"/>
              <xs:enumeration value="ActivityOnNodeNetwork"/>
              <xs:enumeration value="SequentialFunctionChart"/>
              <xs:enumeration value="FunctionBlockDiagram"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```



```

        </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="ResourceStateChangeDefinition" minOccurs="0"
maxOccurs="unbounded">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="DefinitionName" type="xs:string"
use="required"/>
                <xs:attribute name="Duration" type="xs:decimal" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="InterruptibleAction" minOccurs="0">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Value" type="xs:boolean" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="StateChartSubCharts" minOccurs="0"
maxOccurs="unbounded">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="POURef" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="StateChartStateType" minOccurs="0">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="StateChartStateType" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="HigherLevelState"/>
                            <xs:enumeration value="HistoryConnector"/>
                            <xs:enumeration value="ConditionConnector"/>
                            <xs:enumeration value="StateForActivity"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="StateChartActionType" minOccurs="0">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="StateChartActionType" use="required">

```



```

        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="DoAction"/>
                <xs:enumeration value="ExitAction"/>
                <xs:enumeration value="EntryAction"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="TimingDiagramResourceGroup" minOccurs="0">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="TimingDiagramPLCVariable" minOccurs="0">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Name" type="xs:string" use="required"/>
                <xs:attribute name="DataType" type="xs:string" use="optional"/>
                <xs:attribute name="Address" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="StateStatus" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Current" type="xs:boolean" use="required"/>
                <xs:attribute name="Terminal" type="xs:boolean" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="ActionStatus" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Initial" type="xs:boolean" use="required"/>
                <xs:attribute name="Current" type="xs:boolean" use="required"/>
                <xs:attribute name="Terminal" type="xs:boolean" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="Unit" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="aml:addDataBaseObject">
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>

```

```

        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

I.2 MathMLinIEC61131-XML

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:formula="http://www.AutomationML.org/iec62714-4Ed1/AutomationML_PLCOpen"
  targetNamespace="http://www.AutomationML.org/iec62714-4Ed1/AutomationML_PLCOpen"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="formula">
    <xs:annotation>
      <xs:documentation>Root element for the variable mappings for MathML usage within
        IEC 61131-XML.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="variable" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="refMathMLVariable" type="xs:string" use="optional"/>
            <xs:attribute name="refGlobalID" type="xs:ID" use="required"/>
            <xs:attribute name="direction" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="In"/>
                  <xs:enumeration value="Out"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
      <xs:attribute name="ID" type="xs:ID" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Bibliography