INTERNATIONAL STANDARD

IEC 61499-2

First edition
2005-01

Function blocks –

Part 2:
Software tools requirements

Reference number
IEC 61499-2:2005(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/ justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

IEC 61499-2

First edition
2005-01

Function blocks –

Part 2:
Software tools requirements

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE X

For price, see current catalogue

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FUNCTION BLOCKS –

Part 2: Software tool requirements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61499-2 has been prepared by IEC technical committee 65: Industrial-process measurement and control.

This standard cancels and replaces IEC/PAS 61499-2 published in 2001. This first edition constitutes a technical revision.

The following major technical changes have occurred between the PAS edition and this edition:

a) Syntax for network segments, links and parameters has been added in Annex A for consistency with IEC 61499-1.
b) Syntax for parameters instead of constant data connections has been included for consistency with IEC 61499-1.

The text of this standard is based on the following documents:

CDV	Report on voting
65/339/CDV	65/347/RVC

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

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

IEC 61499 consists of the following parts, under the general title *Function blocks:*

Part 1: Architecture

Part 2: Software tool requirements

Part 3: Tutorial information

Part 4: Rules for compliance profiles [1]

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

A bilingual version of this standard may be issued at a later date.

[1] Under consideration.

INTRODUCTION

The IEC 61499 series consists of four Parts:

- Part 1 contains:
  - general requirements, including an introduction, scope, normative references, definitions, and reference models;
  - rules for the declaration of *function block types*, and rules for the behaviour of *instances* of the types so declared;
  - rules for the use of function blocks in the *configuration* of distributed Industrial-Process Measurement and Control *Systems* (IPMCSs);
  - rules for the use of function blocks in meeting the communication requirements of distributed IPMCSs;
  - rules for the use of function blocks in the management of *applications*, *resources* and *devices* in distributed IPMCSs.
- Part 2 (this part of IEC 61499) defines requirements for *software tools* to support the following systems engineering tasks enumerated in Clause 1 of IEC 61499-1:
  - the specification of *function block types*;
  - the functional specification of *resource types* and *device types*;
  - the specification, analysis, and validation of distributed IPMCSs;
  - the *configuration*, *implementation*, operation, and maintenance of distributed IPMCSs;
  - the exchange of *information* among *software tools*.

  It is assumed that such software tools may be used in the context of an Engineering Support System (ESS) as described in Clause C.1 of IEC 61499-1.
- Part 3 has the purpose of increasing the understanding, acceptance, and both generic and domain-specific applicability of IPMCS architectures and software tools meeting the requirements of the other Parts, by providing:
  - answers to Frequently Asked Questions (FAQs) regarding the IEC 61499 series;
  - examples of the use of IEC 61499 constructs to solve frequently encountered problems in control and automation engineering.
- Part 4 defines rules for the development of *compliance profiles* which specify the features of IEC 61499-1 and 61499-2 to be implemented in order to promote the following attributes of IEC 61499-based systems, devices and software tools:
  - interoperability of devices from multiple suppliers;
  - portability of software between software tools of multiple suppliers; and
  - configurability of devices from multiple vendors by software tools of multiple suppliers.

FUNCTION BLOCKS –

Part 2: Software tool requirements

1 Scope

This part of IEC 61499 defines requirements for *software tools* to support the following systems engineering tasks enumerated in Clause 1 of IEC 61499-1:

– the specification of *function block types*;

– the functional specification of *resource types* and *device types*;

– the specification, analysis, and validation of distributed IPMCSs;

– the *configuration*, *implementation*, operation, and maintenance of distributed IPMCSs;

– the exchange of *information* among *software tools*.

It is assumed that such software tools may be used in the context of an Engineering Support System (ESS) as described in Clause C.1 of IEC 61499-1.

It is beyond the scope of this part of IEC 61499 to specify the entire life cycle of industrial-process measurement and control systems (IPMCSs), or the entire set of tasks and activities required to support an IPCMS over its life cycle. However, other standards which do specify such tasks and activities may extend or modify the requirements specified in this Part.

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 61499-1, *Function blocks - Part 1: Architecture*

IEC 61499-4, *Function Blocks - Part 4: Rules for compliance profiles*[2]

The normative references given in IEC 61499-1 apply to this part of IEC 61499.

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61499-1 as well as the following apply.

3.1
library element
collection of declarations applying to a data type, function block type, adapter type, subapplication type, resource type, device type, or system configuration.

---

2 To be published.

4 Software tool requirements

4.1 Information to be provided by the software tool supplier

This Clause defines the functional requirements of *software tools* that support the performance of the systems engineering tasks enumerated in Clause 1.

The supplier of a *software tool* shall specify the following information in addition to other information required in this Clause:

a) The type or types of *library element* to which the software tool applies.

b) The engineering task or tasks supported by the software tool. Task descriptions may be taken from the enumeration of engineering tasks given in Clause 1, or may be defined by the supplier.

4.2 Exchange of library elements

A *software tool* shall be capable of exchanging its *library elements* with other software tools. This exchange shall take the form of *data* in the format defined in Annex A, written on physical media or exchanged over communication links or networks.

4.3 Information to be provided by the supplier of library elements

NOTE The provisions of this subclause are intended to provide the means by which the provider of a library element may achieve protection of intellectual property while still providing sufficient information to permit the effective use of the library element.

The provider of a *library element* may elect to provide an *implementation* of the library element.

EXAMPLE 1 The provider of a *function block type* library element may provide an implementation of the function block type as:

- one or more *instances* of the function block type in a *resource* contained in a *device* of Class 0 or higher as described in IEC 61499-4;
- an instantiable implementation of the function block type in a *resource* contained in a *device* of Class 1 or higher as described in IEC 61499-4;
- a file in an implementation-dependent format suitable for installation in a *resource* contained in a *device* of Class 2 as described in IEC 61499-4, for instance using an XML syntax which may be defined in a compliance profile developed according to the rules given in IEC 61499-4.

When an implementation of a library element is provided, the provider is not required to provide full details of the implementation. However, the provider shall provide sufficient information to enable the user to fully determine the functionality of the provided library element.

EXAMPLE 2 The requirement of the above paragraph would be met by the provider of an *instance* of a function block *type* in a *resource* through the provision, at a minimum, of the following information:

- a *function block type* library element specifying its *event* and *data interfaces* as defined in IEC 61499-1, 5.2.1, and its *services* as defined in IEC 61499-1, 6.1.3;
- *resource type* and *device type* library elements showing the occurrence and connections of the function block *instances*.

4.4 Display of declarations

A software tool shall be capable of displaying the *declarations* of its associated *library elements* in a form appropriate to the engineering task. This display may utilize the graphical or textual formats defined in IEC 61499-1, or a format defined by the supplier of the software tool.

NOTE The *declarations* of a library element may define its *interfaces* (event and data inputs and outputs) and internal *variables* as well as its *algorithms* and the control of their *execution*, for example via an *execution control chart* (ECC), etc.

4.5 Modification of declarations

A software tool shall enable its user to modify the declarations of its associated library elements as appropriate to the engineering task. Such modifications may include adding, deleting or changing the contents of declarations, and may be performed either graphically or textually or both.

4.6 Validation of declarations

If required by the associated engineering task, a software tool shall provide facilities for validation of the declarations of its associated library elements. Such facilities may include, but are not limited to:

a) Checking the correctness of the syntax of declarations.

b) Checking the semantic correctness of declarations, for instance, checking whether all *function block instances* in an *application* and its associated *subapplications* are properly allocated to *resources*, interconnected within resources, and intercommunicating among resources in a *system configuration*.

c) Simulation and testing of the operation of an *instance* of a library element *type*, either by itself or in association with other instances of the same or different types.

4.7 Implementation of declarations

If required by the associated engineering task, a software tool shall provide facilities for the *implementation* of the *declarations* of its associated *library elements*. Such facilities may include, but are not limited to:

a) The production of executable code ("firmware") for embedding in *instances* of *resource types* and *device types*.

b) The creation and interconnection ("downloading") of *function block instances* in *resources* and *devices*, for instance by using the management facilities defined in subclause 3.3 and Annexes F and G of IEC 61499-1.

4.8 System operation, testing and maintenance

If required by the associated engineering task, a software tool shall provide facilities for the operation, testing and maintenance of an Industrial Process Measurement and Control System (IPMCS) specified by its associated library elements. Such facilities may include, but are not limited to:

a) The facilities described in preceding subclauses of this Clause.

b) The information exchange facilities defined in subclause 6.2 and Annex F of IEC 61499-1.

Annex A
(normative)

Document Type Definitions (DTDs)

A.1 General principles

This Annex presents Document Type Definitions (DTDs) for the exchange of IEC 61499 library elements between *software tools*. These DTDs are defined in the syntax defined in the eXtensible Markup Language (XML) specification (see www.w3.org/TR/2000/REC-xml-20001006).

The correspondences between the DTD elements given in this Annex, the library elements defined in IEC 61499-1, C.2.2, and the textual syntax given in IEC 61499-1, Annex B are given in Table A.1.

Table A.1 – Document Type Definition (DTD) elements

DTD element	LibraryElement	Textual syntax
DataType	DataTypeDeclaration	data_type_declaration (IEC 61131-3-B.1.3)
FBType	FBTypeDeclaration	fb_type_declaration
SubapplicationType	SubapplicationTypeDeclaration	subapplication_type_ declaration
AdapterType	AdapterTypeDeclaration	adapter_type_declaration
ResourceType	ResourceTypeDeclaration	resource_type_ specification
DeviceType	DeviceTypeDeclaration	device_type_specification
System	SystemConfiguration	system_configuration

The first table of each subclause of this Annex contains the DTD for the corresponding library element. The second table of each subclause provides a reference to the textual syntax (if any) plus an explanation for the major elements and attributes in the DTD. Following this, examples are given of the resulting XML files for typical library elements.

If there is a conflict between the provisions of this Annex and the provisions of Annex B of IEC 61499-1, the provisions of the latter shall prevail.

NOTE 1 The examples given in this Annex provide a representative but not exhaustive sample of the features of the associated DTDs. In particular, these examples are not intended to be used as a test suite for compliance to the provisions of this part of IEC 61499.

A.2 DataType DTD

An XML document complying with the DTD in Table A.2 represents a DataTypeDeclaration object as described in Clause C.1 of IEC 61499-1.

Table A.2 – DataType DTD

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT DataType (Identification?, VersionInfo+, CompilerInfo?,
ASN1Tag?, (DirectlyDerivedType | EnumeratedType | SubrangeType |
ArrayType | StructuredType))>

<!ATTLIST DataType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT Identification EMPTY>
<!ATTLIST Identification
 Standard CDATA #IMPLIED
 Classification CDATA #IMPLIED
 ApplicationDomain CDATA #IMPLIED
 Function CDATA #IMPLIED
 Type CDATA #IMPLIED
 Description CDATA #IMPLIED>
```

```
<!ELEMENT VersionInfo EMPTY>
<!ATTLIST VersionInfo
 Organization CDATA #REQUIRED
 Version CDATA #REQUIRED
 Author CDATA #REQUIRED
 Date CDATA #REQUIRED
 Remarks CDATA #IMPLIED>
```

```
<!ELEMENT ASN1Tag EMPTY>
<!ATTLIST ASN1Tag
 Class (UNIVERSAL | APPLICATION | CONTEXT | PRIVATE) #IMPLIED
 Number CDATA #REQUIRED>
```

```
<!ELEMENT CompilerInfo (Compiler*)>
<!ATTLIST CompilerInfo
 header CDATA #IMPLIED
 classdef CDATA #IMPLIED>
```

```
<!ELEMENT Compiler EMPTY>
<!ATTLIST Compiler
 Language (Java | Cpp | C | Other) #REQUIRED
 Vendor CDATA #REQUIRED
 Product CDATA #REQUIRED
 Version CDATA #REQUIRED>
```

```
<!ELEMENT DirectlyDerivedType EMPTY>
<!ATTLIST DirectlyDerivedType
 BaseType (BOOL | SINT | INT | DINT | LINT | USINT | UINT | UDINT
| ULINT | REAL | LREAL | TIME | DATE | TIME_OF_DAY | TOD
| DATE_AND_TIME | DT | STRING | BYTE |WORD | DWORD |LWORD | WSTRING)
  #REQUIRED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT EnumeratedType (EnumeratedValue+)>
<!ATTLIST EnumeratedType
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT EnumeratedValue EMPTY>
<!ATTLIST EnumeratedValue
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT SubrangeType (Subrange)>
<!ATTLIST SubrangeType
 BaseType (SINT|INT|DINT|LINT|USINT|UINT|UDINT|ULINT) #REQUIRED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

Table A.2 – DataType DTD

```
<!ELEMENT Subrange EMPTY>
<!ATTLIST Subrange
 LowerLimit CDATA #REQUIRED
 UpperLimit CDATA #REQUIRED>
```

```
<!ELEMENT ArrayType (Subrange+)>
<!ATTLIST ArrayType
 BaseType CDATA #REQUIRED
 InitialValues CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT StructuredType (VarDeclaration|SubrangeVarDeclaration)+>
<!ATTLIST StructuredType
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT VarDeclaration EMPTY >
<!ATTLIST VarDeclaration
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 ArraySize CDATA #IMPLIED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT SubrangeVarDeclaration (Subrange+) >
<!ATTLIST SubrangeVarDeclaration
 Name CDATA #REQUIRED
 Type (SINT|INT|DINT|LINT|USINT|UINT|UDINT|ULINT) #REQUIRED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

Explanations of the elements of the DTD given in Table A.2, and (where applicable) references to the formal syntax for their attributes, are given in Table A.3.

Table A.3 – DataType DTD elements

Element attributes	Textual syntax (IEC 61131-3, Annex B)	Explanation
DataType		See IEC 61131-3
Name	data_type_name	
Comment	--	A comment as per IEC 61131-3 without (* and *) delimiters
Identification	Information for data base retrieval	
Standard	Primary reference standard in number-part-subclause format	
Classification	Classification code as defined in reference standard	
ApplicationDomain	Application domain as defined in reference standard	
Function	Function of this element as defined in reference standard	
Type	Element type (e.g., device type) as defined in reference standard	
Description	Descriptive phrase as defined in reference standard	

Table A.3 – DataType DTD elements

Element attributes	Textual syntax (IEC 61131-3, Annex B)	Explanation
VersionInfo	--	Possibly one of several entries: First entry – Most recent version Second entry – Immediately preceding released version, etc. last entry – First released version
Organization	--	The organization supplying this library element
Version	digit [digit] '.' digit [digit] [letter]	The Version identification for this library element
Author	--	The author of this library element
Date	date_literal ['-' daytime]	The release date of this version
Remarks	--	Comments relating to this version
ASN1Tag		ASN.1 tag per ISO/IEC 8824-1
Class		ASN.1 tag class per ISO/IEC 8824-1
Number		ASN.1 tag number per ISO/IEC 8824-1
CompilerInfo	--	Information for and about compilers used with this class
header	--	Header information such as package, imports, etc.
classdef	--	The class definition information such as superclass and implemented interfaces. If none is given, a default abstract superclass is used.
Compiler	--	Possibly one of several compilers used with this version
Language	--	The source language of this compiler
Vendor	--	The vendor of this compiler
Product	--	The product name of this compiler
Version	--	The version of this compiler
DirectlyDerivedType		See IEC 61131-3, Tables 12 and 14, #1
BaseType	elementary_type_name	
InitialValue	constant	
Comment		A comment as per IEC 61131-3 without (* and *) delimiters
EnumeratedType		See IEC 61131-3, Tables 12 and 14, #2
InitialValue	enumerated_value	
Comment		A comment as per IEC 61131-3 without (* and *) delimiters
EnumeratedValue		See IEC 61131-3, Table 14, #2
Name	enumerated_value	
Comment		A comment as per IEC 61131-3 without (* and *) delimiters
SubrangeType	--	See IEC 61131-3, Tables 12 and 14, #3
BaseType	integer_type_name	
InitialValue	signed_integer	
Comment		A comment as per IEC 61131-3 without (* and *) delimiters

Table A.3 – DataType DTD elements

Element attributes	Textual syntax (IEC 61131-3, Annex B)	Explanation
Subrange	See IEC 61131-3, Tables 12 and 14, #3	
LowerLimit	signed_integer	
UpperLimit	signed_integer	
ArrayType	See IEC 61131-3, Tables 12 and 14, #4	
BaseType	non_generic_type_name	
InitialValues	array_initialization	
StructuredType	See IEC 61131-3, Tables 12, #5 and 14, #5 and #6	
VarDeclaration		
Name	structure_element_name	
Type	non_generic_type_name	
ArraySize	See NOTE 1	
InitialValue	See NOTE 2	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
SubrangeVarDeclaration	See IEC 61131-3, 2.3.3.	
Name	structure_element_name	
Type	integer_type_name	
InitialValue	signed_integer	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	

NOTE 1 The syntax of this attribute when present is equivalent to the syntactic expression
(subrange {',' subrange}) | integer {',' integer}
where the non-terminals subrange and integer are as defined in Annex B of IEC 61131-3. Each term of the second form is equivalent to the subrange 0..n-1, where n is the value of the corresponding integer syntactic element. If this attribute is missing, the structure component is not an anonymously defined array.

NOTE 2 The syntax of this attribute is the syntax for initialization of the corresponding variable type as defined in Annex B.1.4.3 of IEC 61131-3.

EXAMPLE The structured data type ANALOG_CHANNEL_CONFIGURATIONI example is expressed in IEC 61131-3, Table 14 as follows:

```
TYPE ANALOG_CHANNEL_CONFIGURATIONI :
    STRUCT
      RANGE : ANALOG_SIGNAL_RANGE ;
      MIN_SCALE : ANALOG_DATA := -4095 ;
      MAX_SCALE : ANALOG_DATA :=  4095 ;
    END_STRUCT ;
END_TYPE
```

A corresponding XML document could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataType SYSTEM "DataType.dtd"
>
<DataType
   Name="ANALOG_CHANNEL_CONFIGURATIONI"
   Comment="IEC 61131-3, Table 14#5">
<Identification
   Function="Configuration Data"
   Standard="61131-3-2.3.3.2"
   ApplicationDomain="Any"
   Classification="Data type"
   Type="Analog"
   Description="Table 14, #5"/>
<VersionInfo
   Organization="IEC SC65B/WG7/TF3"
   Version="2.0"
   Author="JHC"
   Date="2000-01-31"/>
<StructuredType>
  <VarDeclaration Name="SIGNAL_RANGE"
     Type="ANALOG_SIGNAL_RANGE"/>
  <VarDeclaration Name="MIN_SCALE"
     Type="ANALOG_DATA"
     InitialValue="-4095"/>
  <VarDeclaration Name="MAX_SCALE"
     Type="ANALOG_DATA"
     InitialValue="4095"/>
</StructuredType>
</DataType>
```

A.3 LibraryElement DTD

An XML document complying with the DTD in Table A.4 represents a LibraryElement object as described in Annex C of IEC 61499-1. Possible root elements of such a document are `FBType`, `AdapterType`, `ResourceType`, `DeviceType`, `System`, and `SubappType`, representing the concrete subclasses FBTypeDeclaration, AdapterTypeDeclaration, ResourceTypeDeclaration, DeviceTypeDeclaration, SystemConfiguration, and Subapplication-TypeDeclaration of the abstract superclass LibraryElement, respectively. The DataType-Declaration subclass is represented separately by the DTD given in Clause A.1 of this document.

Table A.4 – Library Element DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Common elements -->
<!ELEMENT Identification EMPTY>
<!ATTLIST Identification
 Standard CDATA #IMPLIED
 Classification CDATA #IMPLIED
 ApplicationDomain CDATA #IMPLIED
 Function CDATA #IMPLIED
 Type CDATA #IMPLIED
 Description CDATA #IMPLIED>
```

```
<!ELEMENT VersionInfo EMPTY>
<!ATTLIST VersionInfo
 Organization CDATA #REQUIRED
 Version CDATA #REQUIRED
 Author CDATA #REQUIRED
 Date CDATA #REQUIRED
 Remarks CDATA #IMPLIED>
```

Table A.4 – Library Element DTD

```
<!ELEMENT CompilerInfo (Compiler*)>
<!ATTLIST CompilerInfo
 header CDATA #IMPLIED
 classdef CDATA #IMPLIED>

<!ELEMENT Compiler EMPTY>
<!ATTLIST Compiler
 Language (Java | Cpp | C | Other) #REQUIRED
 Vendor CDATA #REQUIRED
 Product CDATA #REQUIRED
 Version CDATA #REQUIRED>
```

```
<!ELEMENT FBNetwork
(FB*,EventConnections?,DataConnections?,AdapterConnections?)>

<!ELEMENT FB (Parameter*)>
<!ATTLIST FB
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED>
```

```
<!ELEMENT EventConnections (Connection+)>

<!ELEMENT DataConnections (Connection+)>
<!ELEMENT AdapterConnections (Connection+)>
<!ELEMENT Connection EMPTY>
<!ATTLIST Connection
 Source CDATA #REQUIRED
 Destination CDATA #REQUIRED
 Comment CDATA #IMPLIED
 dx1 CDATA #IMPLIED
 dx2 CDATA #IMPLIED
 dy CDATA #IMPLIED
>
```

```
<!-- FBType elements -->
<!ELEMENT FBType
(Identification?,VersionInfo+,CompilerInfo?,InterfaceList,
 (BasicFB | FBNetwork)?, Service?) >
<!ATTLIST FBType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
>
```

```
<!ELEMENT InterfaceList
(EventInputs?,EventOutputs?,InputVars?,OutputVars?,
 Sockets?, Plugs?)>
<!ELEMENT EventInputs (Event+)>
<!ELEMENT EventOutputs (Event+)>
<!ELEMENT InputVars (VarDeclaration+)>
<!ELEMENT OutputVars (VarDeclaration+)>
<!ELEMENT Sockets (AdapterDeclaration+)>
<!ELEMENT Plugs (AdapterDeclaration+)>
```

```
<!ELEMENT Event (With*)>
<!ATTLIST Event
 Name CDATA #REQUIRED
 Type CDATA #IMPLIED
 Comment CDATA #IMPLIED>

<!ELEMENT With EMPTY>
<!ATTLIST With
 Var CDATA #REQUIRED>
```

Table A.4 – Library Element DTD

```
<!ELEMENT VarDeclaration EMPTY>
<!ATTLIST VarDeclaration
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 ArraySize CDATA #IMPLIED
 InitialValue CDATA #IMPLIED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT AdapterDeclaration (Parameter*)
<!ATTLIST AdapterDeclaration
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED>
```

```
<!ELEMENT BasicFB (InternalVars?,ECC?,Algorithm*)>
<!ELEMENT InternalVars (VarDeclaration+)>
<!ELEMENT ECC (ECState+,ECTransition+)>
```

```
<!ELEMENT ECState (ECAction*)>
<!ATTLIST ECState
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED>
```

```
<!ELEMENT ECTransition EMPTY>
<!ATTLIST ECTransition
 Source CDATA #REQUIRED
 Destination CDATA #REQUIRED
 Condition CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED>
```

```
<!ELEMENT ECAction EMPTY>
<!ATTLIST ECAction
 Algorithm CDATA #IMPLIED
 Output CDATA #IMPLIED>
```

```
<!ELEMENT Algorithm ((FBD | ST | LD | Other))>
<!ATTLIST Algorithm
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT FBD (FB+,DataConnections)>
```

```
<!ELEMENT ST PCDATA>
```

```
<!ELEMENT LD (Rung+)>
<!ELEMENT Rung EMPTY>
<!ATTLIST Rung
 Output CDATA #REQUIRED
 Expression CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT Other PCDATA>
<!ATTLIST Other
 Language CDATA #REQUIRED >
```

```
<!ELEMENT Service (ServiceSequence+)>
<!ATTLIST Service
 RightInterface CDATA #REQUIRED
 LeftInterface CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

Table A.4 – Library Element DTD

```
<!ELEMENT ServiceSequence (ServiceTransaction*)>
<!ATTLIST ServiceSequence
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!ELEMENT ServiceTransaction (InputPrimitive?, OutputPrimitive*)>

<!ELEMENT InputPrimitive EMPTY>
<!ATTLIST InputPrimitive
 Interface CDATA #REQUIRED
 Event CDATA #REQUIRED
 Parameters CDATA #IMPLIED>

<!ELEMENT OutputPrimitive EMPTY>
<!ATTLIST OutputPrimitive
 Interface CDATA #REQUIRED
 Event CDATA #REQUIRED
 Parameters CDATA #IMPLIED>
```

```
<!-- AdapterType elements -->
<!ELEMENT AdapterType
(Identification?,VersionInfo+,CompilerInfo?,InterfaceList,Service?)>
<!ATTLIST AdapterType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>
```

```
<!-- ResourceType elements -->
<!ELEMENT ResourceType (Identification?,VersionInfo+, CompilerInfo?,
FBTypeName*, VarDeclaration*, FBNetwork?)>
<!ATTLIST ResourceType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>

<!ELEMENT FBTypeName EMPTY>
<!ATTLIST FBTypeName
 Name CDATA #REQUIRED>
```

```
<!-- DeviceType elements -->
<!ELEMENT DeviceType (Identification?,VersionInfo+, CompilerInfo?,
VarDeclaration*, ResourceTypeName*, Resource*, FBNetwork?)>
<!ATTLIST DeviceType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>

<!ELEMENT ResourceTypeName EMPTY>
<!ATTLIST ResourceTypeName
 Name CDATA #REQUIRED >

<!ELEMENT Resource (Parameter*,FBNetwork?)>
<!ATTLIST Resource
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED>
```

Table A.4 – Library Element DTD

```
<!-- System elements -->
<!ELEMENT System (Identification?, VersionInfo+, Application*, Device+,
Mapping*, Segment*, Link*)>
<!ATTLIST System
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>

<!ELEMENT Application (SubAppNetwork)>
<!ATTLIST Application
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>

<!ELEMENT Mapping EMPTY>
<!ATTLIST Mapping
 From CDATA #REQUIRED
 To CDATA #REQUIRED>

<!ELEMENT Device (Parameter*,Resource*,FBNetwork?)>
<!ATTLIST Device
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
 x CDATA #IMPLIED
 y CDATA #IMPLIED>
```

```
<!-- SubAppType elements -->
<!ELEMENT SubAppType
(Identification?,VersionInfo+,CompilerInfo?,SubAppInterfaceList,
SubAppNetwork?)>
<!ATTLIST SubAppType
 Name CDATA #REQUIRED
 Comment CDATA #IMPLIED>

<!ELEMENT SubAppInterfaceList
(SubAppEventInputs?,SubAppEventOutputs?,InputVars?,OutputVars?)>
<!ELEMENT SubAppEventInputs (SubAppEvent+)>
<!ELEMENT SubAppEventOutputs (SubAppEvent+)>

<!ELEMENT SubAppEvent EMPTY>
<!ATTLIST SubAppEvent
 Name CDATA #REQUIRED
 Type CDATA #IMPLIED
 Comment CDATA #IMPLIED>

<!ELEMENT SubAppNetwork
(SubApp*,FB*,EventConnections?,DataConnections?, AdapterConnections?)>

<!ELEMENT SubApp EMPTY>
<!ATTLIST SubApp
 Name CDATA #REQUIRED
 Type CDATA #REQUIRED
 Comment CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED>
```

Table A.4 – Library Element DTD

```
<!-- Network elements -->
<!ELEMENT Segment (Parameter*)>
<!ATTLIST Segment
  Name CDATA #REQUIRED
  Type CDATA #REQUIRED
  Comment CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED
  dx1 CDATA #IMPLIED>

<!ELEMENT Parameter EMPTY>
<!ATTLIST Parameter
  Name CDATA #REQUIRED
  Value CDATA #REQUIRED
  Comment CDATA #IMPLIED>

<!ELEMENT Link (Parameter*)>
<!ATTLIST Link
  SegmentName CDATA #REQUIRED
  CommResource CDATA #REQUIRED
  Comment CDATA #IMPLIED >
```

Explanations of some of the elements of the above DTD, and (where applicable) references to the formal syntax for their attributes, are given in Table A.5.

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
Identification	See Table A.3	
VersionInfo		
CompilerInfo		
Compiler		
FBNetwork	A *function block network* as defined in IEC 61499-1.	
FB	A *function block instance* as defined in IEC 61499-1.	
Name	fb_instance_name	
Type	fb_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
x, y	See Annex B.	
Connection	An *event connection, data connection* or *adapter connection* as defined in IEC 61499-1.	
Source	See NOTE 3	
Destination	See NOTE 3	
dx1, dx2, dy	See Annex B	
FBType	A *function block type* as described in IEC 61499-1.	
Name	fb_type_name	
Comment	A comment as per IEC 61131 without (* and *) delimiters	

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
Event	A *declaration* of an *event interface.*	
Name	event_input_name | event_output_name	See NOTE 4
Type	event_type	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
With	A *declaration* of an association between an *event* and a *variable.*	
Var	input_variable_name | output_ variable _name	See NOTE 4
VarDeclaration	A declaration of a variable.	
Name	input_variable_name | output_variable_name | internal_variable_name	See NOTE 5
Type	identifier	
ArraySize	See NOTE 6	
InitialValue	See NOTE 7	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
AdapterDeclaration	A declaration of a plug or socket interface of a function block type.	
Name	plug_name | socket_name	See NOTE 8
Type	adapter_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
x, y	location (See Annex B) of plug or socket in the internal function block network of a *composite function block type*.	
ECState	An *EC state* as defined in IEC 61499-1.	
Name	ec_state_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
x, y	See Annex B	
ECTransition	An *EC transition* as defined in IEC 61499-1.	
Source	ec_state_name	
Destination	ec_state_name	
Condition	ec_transition_condition	
x, y	See Annex B	
ECAction	An *EC action* as defined in IEC 61499-1.	
Algorithm	algorithm_name	
Output	event_output_name	

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
Algorithm	An *algorithm* in a specified language (NOTE 1)	
Name	algorithm_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
ST	An *algorithm* in the IEC 61131-3 ST language, whose contents are in the syntax of a statement_list per IEC 61131-3, Annex B.	
Rung	A rung of an algorithm in the LD language	
Output	See NOTE 2	
Expression	See NOTE 2	
Other	An algorithm in a language other than FBD, ST or LD, whose contents are in the syntax defined for the particular language; see NOTE 1.	
Language	The name of the programming language; see NOTE 1	
Service	A *declaration* of a *service* as per IEC 61499-1	
RightInterface	service_interface_name	
LeftInterface	service_interface_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
ServiceSequence	A declaration of a service sequence as per IEC 61499-1	
Name	sequence_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
InputPrimitive	An "input" *service primitive* as per IEC 61499-1	
Interface	service_interface_name	
Event	(([plug_name '.'] event_input_name) | (socket_name '.' event_output_name)) ['+' | '-']	
Parameters	input_variable_name {',' input_variable_name}	
OutputPrimitive	An "output" *service primitive* as per IEC 61499-1	
Interface	service_interface_name	
Event	('NULL' | ([plug_name '.'] event_output_name) | socket_name '.' event_input_name)) ['+' | '-']	
Parameters	output_variable_name {',' output_variable_name}	
AdapterType	A declaration of an adapter interface type per IEC 61499-1	
Name	adapter_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
ResourceType	A *declaration* of a *resource type* as per IEC 61499-1	
Name	resource_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
FBTypeName	The name of a function block type supported by all instances of a resource type	
Name	fb_type_name	
DeviceType	A *declaration* of a *device type* per IEC 61499-1	
Name	device_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
ResourceTypeName	The name of a *resource type* supported by all *instances* of a *device type*	
Name	resource_type_name	
Resource	A resource instance present in all instances of a device type	
Name	resource_instance_name	
Type	resource _type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
System	A declaration of a system configuration per IEC 61499-1	
Name	system_name	
Comment	A comment as per IEC 61131-3-2.1.5 without (* and *) delimiters	
Application	A *declaration* of an *application* per IEC 61499-1	
Name	application_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
Mapping	Mapping of a *function block instance* from an *application* onto a function block instance in a *resource*.	
From	fb_instance_reference	Hierarchical *function block instance name* in its *application*, e.g., APP1.SUBAPP2.FB2
To	fb_resource_reference	Hierarchical *function block instance name* in the physical system (see NOTE 10).
Device	A declaration of a device configuration as per IEC 61499-1	
Name	device_instance_name	
Type	device_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
SubAppType	A declaration of a subapplication type per IEC 61499-1	
Name	subapp_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
SubAppEvent	A declaration of an event interface of a subapplication type.	
Name	event_input_name | event_output_name	See NOTE 9
Type	event_type	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
SubApp	A *subapplication instance* as defined in IEC 61499-1.	
Name	subapp_instance_name	
Type	subapp_type_name	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
x, y	See Annex B	
Segment	A segment of a communication *network*	
Name	identifier	
Type	identifier	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
x, y, dx1	See Annex B	
Link	A link between a Segment element and a Device element	
CommResource	resource_hierarchy	See NOTE 11
SegmentName	identifier	The segment to be linked
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	
Parameter	A *parameter* of an element, e.g. a Segment or Link element.	
Name	identifier	
Value	A character string in an appropriate format to express the value of the associated parameter.	
Comment	A comment as per IEC 61131-3 without (* and *) delimiters	

Table A.5 – LibraryElement DTD elements

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation

NOTE 1 The specification of algorithms in languages other than FBD, ST and LD is beyond the scope of this part of IEC 61499.

NOTE 2 Since the FBD and ST languages are available for the specification of complex algorithms, it is recommended that the usage of the LD language in the context of this part of IEC 61499 be limited to rungs performing the evaluation of assignment statements of the form `<output> := <expression>`. For portability between software tools, it is further recommended that the XML `Expression` element have the following simple postfix-operator textual syntax with whitespace_separated terms:

```
expression ::= and_expression
and_expression ::= (variable_name ['!']) | or_expression
      | and_expression and_expression '&'
or_expression ::= and_expression | or_expression or_expression '|'
```

See EXAMPLE 1 for an illustration of this recommended usage.

NOTE 3 Depending on the context, the syntax of a `Source` or `Destination` element should correspond to the syntax of the respective element in one of the productions `event_conn`, `data_conn`, `adapter_conn`, `subapp_event_conn`, `subapp_data_conn`, `config_event_conn`, `config_data_conn`, `config_adapter_conn`, `devtype_event_conn`, `devtype_data_conn`, or `devtype_adapter_conn` given in Annex B of IEC 61499-1.

NOTE 4 The productions `event_input_name` and `input_variable_name` apply when the `Event` element is part of an `EventInputs` element, and `event_output_name` and `output_variable_name` apply when it is part of an `EventOutputs` element.

NOTE 5 The productions `input_variable_name`, `output_variable_name` and `internal_variable_name` apply when the associated `VarDeclaration` element is part of an `InputVars`, `OutputVars` or `InternalVars` element, respectively.

NOTE 6 The syntax of this element when present shall be equivalent to the syntactic expression `(subrange {',' subrange}) | integer {',' integer}` where the non-terminals `subrange` and `integer` are as defined in Annex B of IEC 61131-3. Each term of the second form is equivalent to the subrange `0..n-1`, where *n* is the value of the corresponding `integer` syntactic element. If this element is missing, the variable is not an array of anonymous type, although it could still be an instance of a previously defined array type.

NOTE 7 The syntax of this element is the syntax for initialization of the corresponding variable type as defined in Annex B of IEC 61131-3.

NOTE 8 The productions `plug_name` and `socket_name` apply when the associated `AdapterDeclaration` element is part of a `Plugs` or `Sockets` element, respectively.

NOTE 9 The productions `event_input_name` and `event_output_name` apply when the `SubAppEvent` element is part of a `SubAppEventInputs` or `SubAppEventOutputs` element, respectively.

NOTE 10 This element may show a full device/resource/FB name hierarchy, e.g. `DEV1.RES2.FB2`; a device/resource hierarchy, e.g., `DEV1.RES2`; or (in the case where the device itself is a single resource) simply a device name, e.g., `DEV1`. In the latter two cases, the FB instance shall have the same name as in the application, e.g., if the source is `APP1.FB3` then the resulting mapping shall be to `DEV1.RES2.FB3` and `DEV1.FB3` respectively.

NOTE 11 This attribute references the communication resource linked to the network segment. It may show a full device/resource name sequence, e.g. `DEV1.RES2`; or (in the case where the device itself provides the communication interface) simply a device name, e.g., `DEV1`.

Annex B
(informative)

Graphics model

B.1 Coordinate system

This Annex presents a simple graphic model which permits the approximate reconstruction of the graphical appearance of *communication networks, function block networks* and *execution control charts (ECCs)* between software tools, utilizing the data defined for the FBNetwork, ECC, ResourceType, DeviceType, SubAppType and System elements in the Library Element DTD in Annex A.

NOTE The graphical model in this Annex is intended to allow passing of information among software tools with a common semantic, but exact reproduction of graphics may not be achieved among tools with different layout and drawing algorithms.

Since the main direction of event and data flow in function block networks is from left to right, and secondarily from top to bottom, the coordinate system of the graphics model has its origin (x=y=0) at the upper left hand corner with x-coordinate values increasing in the left-to-right direction and y-coordinate values increasing in the top-to-bottom direction, as illustrated in Figure B.1.

To achieve independence from font sizes and display resolutions, both x and y coordinates are expressed as multiples of 1 % of the line height (shown as ***h*** in Figure B.1) used for the layout of function block inputs and outputs.

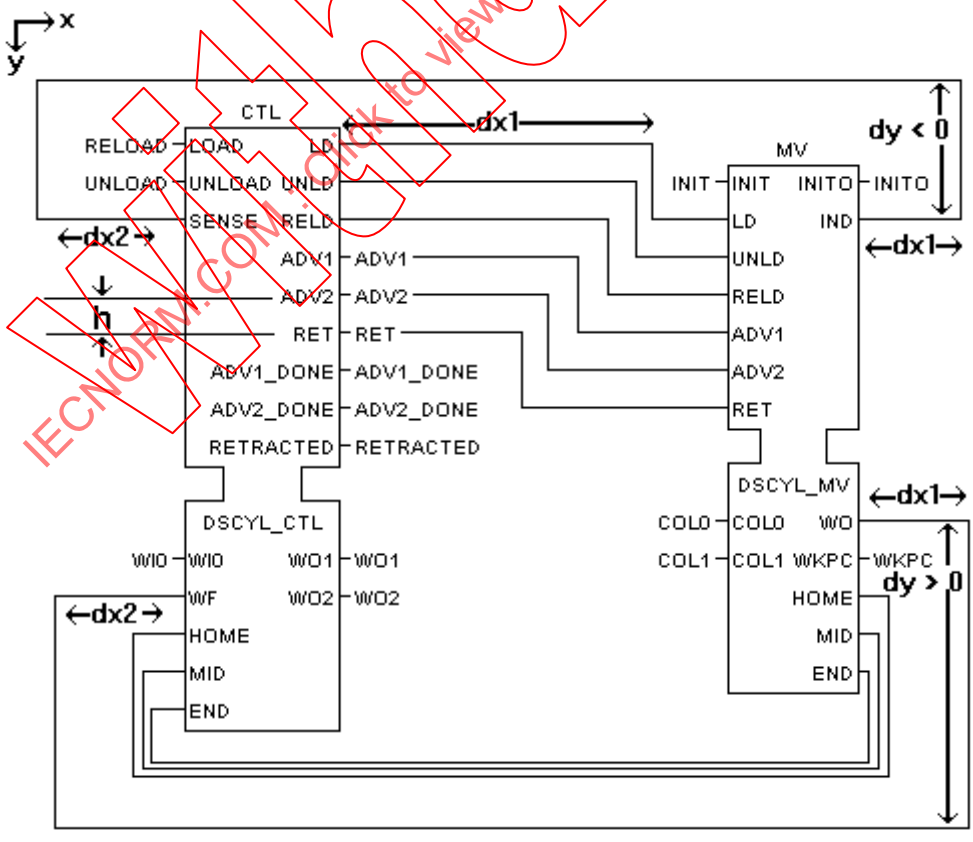


Figure B.1 – Graphics model

EXAMPLE The upper left-hand corner of the DSCYL_CTL instance named CTL in Figure B.1 is located approximately 10h units from the left-hand edge of the diagram and 5h units from the top of the diagram; hence the values of the x and y attributes of the corresponding FB sub-element in an FBNetwork element of an XML document defined according to one of the DTDs listed above would be 1000 and 500, respectively.

B.2 Location of graphical elements

The location of a *function block instance* is determined by the location of the upper left corner of its graphical outline.

The location of an *EC state* is determined by the center point of the bounding box containing the state name.

The location of an *EC transition condition* is determined by the center point of the invisible bounding box containing the transition condition.

NOTE (x,y) coordinates may be used in the transfer syntax of *device* and *resource instances*. However, software tools that use default graphical or tree notations for these elements are not required to use or produce these attributes.

B.3 Routing of connections

As illustrated in Figure B.1, *data connections*, *event connections* and *adapter connections* may be drawn as an odd number of line segments, for example:

a) When the source of the connection is to the left of its destination, the line may be drawn as a single straight line proceeding from the right edge of the function block which provides the source of the connection to the left edge of the function block which provides the destination of the connection.

b) When the source of the connection is to the left of its destination, the connection can be drawn as three contiguous line segments proceeding rightward a distance **dx1** from the right edge of the function block which provides the source of the connection; thence vertically an appropriate distance to proceed horizontally to the left edge of the function block which provides the destination of the connection.

c) When the source of the connection is to the left or right of its destination, the connection can be drawn as five contiguous line segments proceeding rightward a distance **dx1** from the right edge of the function block which provides the source of the connection; thence vertically a distance **dy**; thence horizontally to an x-coordinate a distance **dx2** left of the left edge of the function block which provides the destination of the connection; thence vertically an appropriate distance to proceed horizontally to the left edge of the function block which provides the destination of the connection.

d) *EC transitions* are drawn as two straight lines from the location of the source *EC action* to the location of the EC transition condition and thence to the location of the destination EC action, where these locations are as defined in B.2. The portions of these lines within the bounding boxes of EC state names and transition conditions are hidden, as illustrated in Figure B.2. Arrowheads or other graphic means may be used to indicate the direction of transitions.

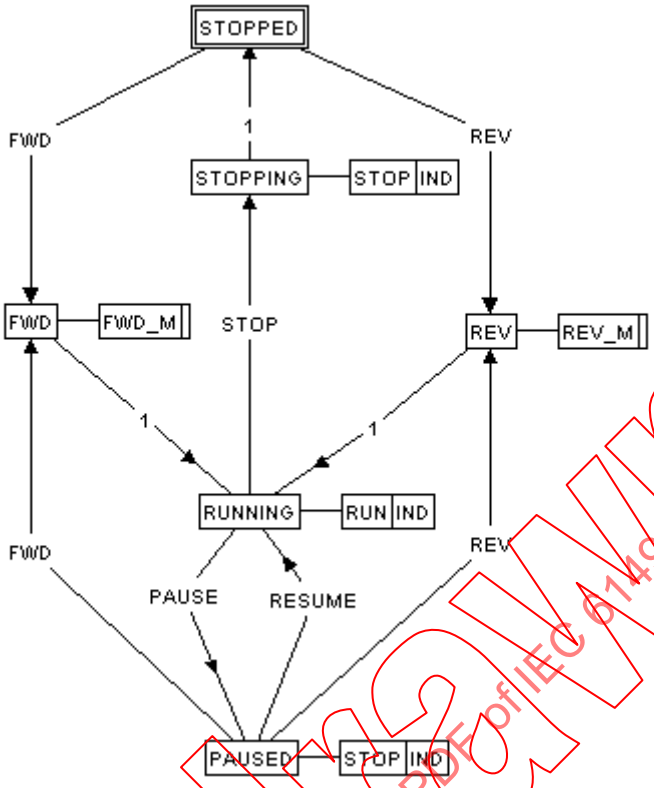


Figure B.2 – ECC drawing example

B.4 Default layouts

Suppliers of software tools shall specify the means employed to obtain default layouts of *ECCs* and *function block networks* when the necessary graphic information is not supplied in the XML transfer document of the associated *library element*, or when the information supplied in the document is inconsistent with the drawing algorithms employed by the particular tool.

NOTE An example of such inconsistency is when a software tool uses a different algorithm to determine the width and height of graphical elements from the algorithm employed by the tool producing the document, which may cause graphical interferences among elements such as overlapping of connecting lines with element outlines.

B.5 Graphical representation of system configurations

A *system configuration* as defined in IEC 61499-1, whose XML representation is given in the `System` element notation defined in Tables A.4 and A.5, can be laid out in the coordinate system defined in B.1, according to the following rules:

a) A *device* is represented as a generally rectangular block containing the device's type name, with its instance name at the top of the block. The (x,y) coordinates of the center of the block are given by the `x` and `y` attributes of the XML `Device` element, respectively.

b) A *network segment* is represented as a horizontal line segment, which may be thick enough vertically to contain textual information such as the segment instance name and type name and may have other implementation-dependent features such as arrowheads. The origin and length of the line segment, and the vertical location of the center of the line, are given by the `x`, `dx1` and `y` parameters of the XML `Segment` element, respectively.

c) A *network link* is represented as a vertical line segment from the center of a device (or other implementation-dependent position) to the horizontal center line of the corresponding network link. Portions of the link may be overlaid by the graphical representations of segments and devices.

Annex C
(informative)

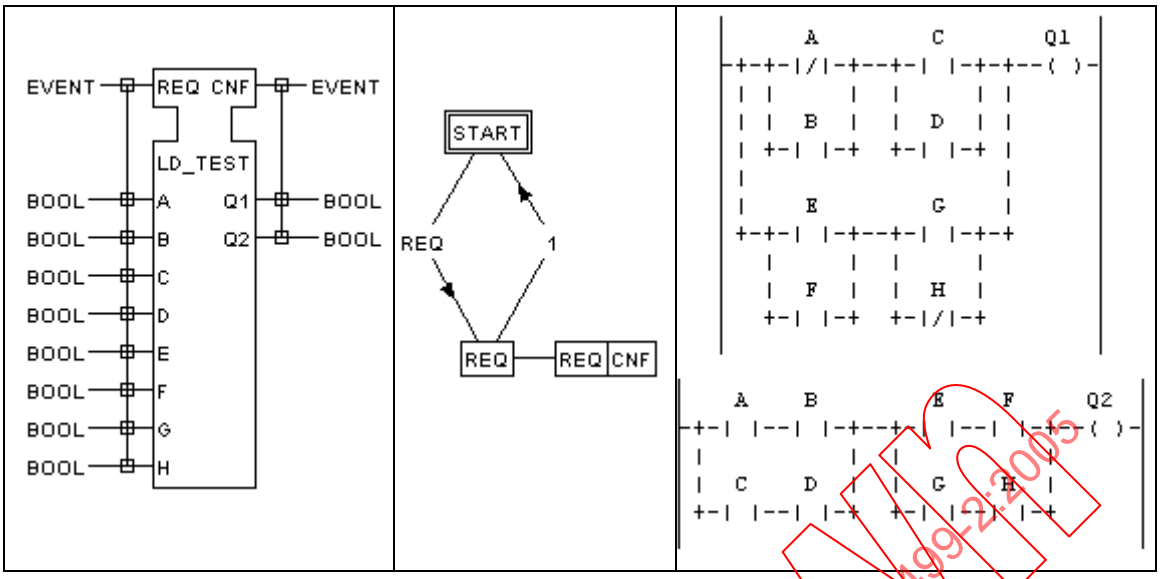
Examples

C.1 Basic function block types

EXAMPLE 1 A basic function block type containing a Ladder Diagram (LD) algorithm according to NOTE 2 of Table A.5 could be expressed textually as follows.

```
FUNCTION_BLOCK LD_TEST (* LD Algorithm Example *)
EVENT_INPUT
    REQ WITH A, B, C, D, E, F, G, H;
END_EVENT
EVENT_OUTPUT
    CNF WITH Q1, Q2; (* Execution Confirmation *)
END_EVENT
VAR_INPUT
    A : BOOL;
    B : BOOL;
    C : BOOL;
    D : BOOL;
    E : BOOL;
    F : BOOL;
    G : BOOL;
    H : BOOL;
END_VAR
VAR_OUTPUT
    Q1 : BOOL;
    Q2 : BOOL;
END_VAR
EC_STATES
    START : (* Initial State *)
    REQ : REQ -> CNF ; (* Normal execution *)
END_STATES
EC_TRANSITIONS
    START TO REQ := REQ;
    REQ TO START := 1;
END_TRANSITIONS
ALGORITHM REQ IN LD :
    Q1 := ((!A|B)&(C|D))|((E|F)&(G|!H));
    Q2 := ((A&B)|(C&D))&((E&F)|(G&H));
END_ALGORITHM
END_FUNCTION_BLOCK
```

The interface, ECC, and `REQ` algorithm could appear graphically as follows.



A corresponding XML document would be:

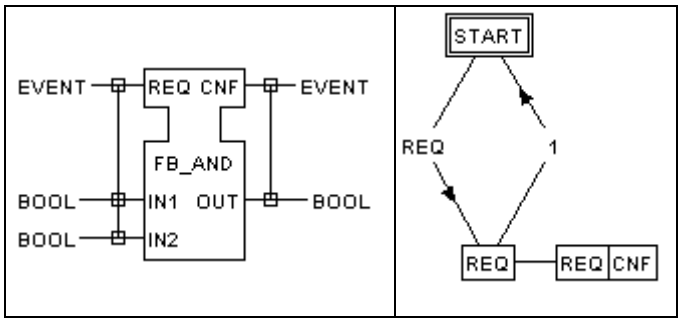
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="LD_TEST" Comment="LD Algorithm Example" >
  <Identification Standard="61499-2-C.1" Description="LD Algorithm Example" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.2" Author="JHC" Date="2000-11-
16" Remarks="Corrected Identification" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-06-
20" Remarks="Tested Sun compiler" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
01" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge" Version="3.0" />
    <Compiler Language="Java" Vendor="Sun" Product="JDK" Version="1.1.8" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="A" />
        <With Var="B" />
        <With Var="C" />
        <With Var="D" />
        <With Var="E" />
        <With Var="F" />
        <With Var="G" />
        <With Var="H" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" Comment="Execution Confirmation" >
        <With Var="Q1" />
        <With Var="Q2" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="A" Type="BOOL" />
      <VarDeclaration Name="B" Type="BOOL" />
      <VarDeclaration Name="C" Type="BOOL" />
      <VarDeclaration Name="D" Type="BOOL" />
      <VarDeclaration Name="E" Type="BOOL" />
      <VarDeclaration Name="F" Type="BOOL" />
      <VarDeclaration Name="G" Type="BOOL" />
      <VarDeclaration Name="H" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="Q1" Type="BOOL" />
      <VarDeclaration Name="Q2" Type="BOOL" />
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="341.1765" y="105.8824" >
      </ECState>
```

```
        <ECState Name="REQ" Comment="Normal execution" x="358.8235" y="858.8235" >
          <ECAction Algorithm="REQ" Output="CNF" />
        </ECState>
        <ECTransition Source="START" Destination="REQ" Condition="REQ" x="170.5882"
y="494.1176" />
        <ECTransition Source="REQ" Destination="START" Condition="1" x="564.7059"
y="500" />
      </ECC>
    <Algorithm Name="REQ" Comment="Normally executed algorithm" >
      <LD >
        <Rung Output="Q1" Expression="A ! B | C D | &#38; E F | G H ! | &#38; | " />
        <Rung Output="Q2" Expression="A B &#38; C D &#38; | E F &#38; G H &#38; | &#38;
" />
      </LD>
    </Algorithm>
    </BasicFB>
  </FBType>
```

EXAMPLE 2 A basic function block type containing a ST algorithm could be expressed textually as follows.

```
FUNCTION_BLOCK FB_AND (* Boolean AND *)
EVENT_INPUT
     REQ WITH IN1, IN2;
END_EVENT
EVENT_OUTPUT
     CNF WITH OUT;
END_EVENT
VAR_INPUT
     IN1 : BOOL;
     IN2 : BOOL;
END_VAR
VAR_OUTPUT
     OUT : BOOL; (* IN1&IN2 *)
END_VAR
EC_STATES
     START ; (* Initial State *)
     REQ : REQ -> CNF ; (* Normal execution
*)
END_STATES
EC_TRANSITIONS
     START TO REQ := REQ;
     REQ TO START := 1;
END_TRANSITIONS
ALGORITHM REQ IN ST :
  OUT := (IN1 & IN2);
END_ALGORITHM
END_FUNCTION_BLOCK
```

The interface and ECC would appear graphically as follows.



A corresponding XML document would be:

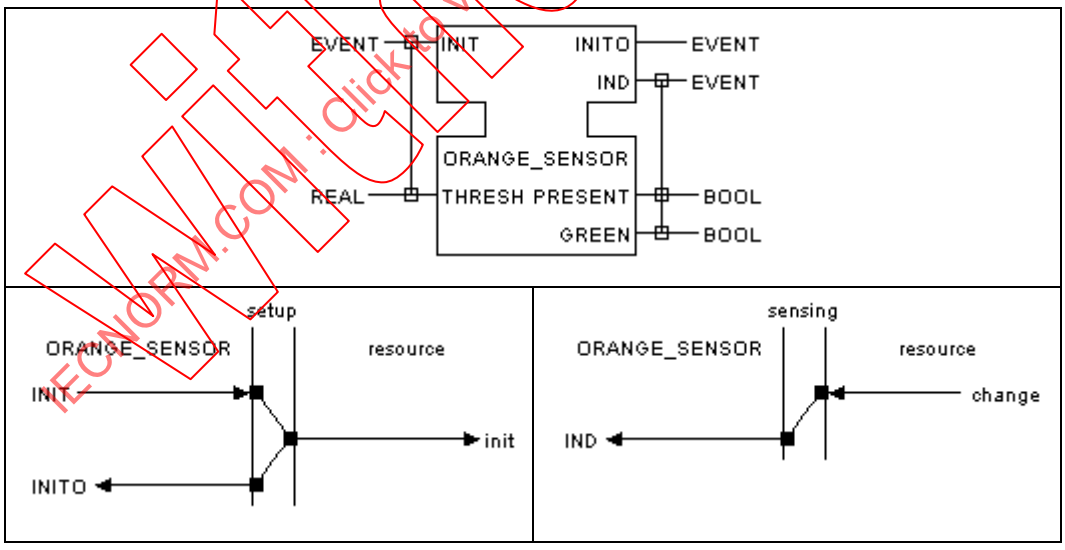
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="FB_AND" Comment="Boolean AND" >
  <Identification Standard="61499-1-D.1" Classification="Math"
ApplicationDomain="Any" Function="AND" Type="Boolean" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-10" Remarks="Tested Sun compiler." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-01-29" Remarks="Simple Boolean AND" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="Sun" Product="JDK" Version="1.1.8"
/>
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge"
Version="3.0" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="IN1" />
        <With Var="IN2" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" >
        <With Var="OUT" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="IN1" Type="BOOL" />
      <VarDeclaration Name="IN2" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="OUT" Type="BOOL" Comment="IN1&#38;IN2" />
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="200" y="105.8824" >
      </ECState>
      <ECState Name="REQ" Comment="Normal execution" x="205.8824"
y="676.4706" >
        <ECAction Algorithm="REQ" Output="CNF" />
      </ECState>
      <ECTransition Source="START" Destination="REQ" Condition="REQ"
x="370.5882" y="405.8824" />
      <ECTransition Source="REQ" Destination="START" Condition="1"
x="52.9412" y="429.4117" />
    </ECC>
  <Algorithm Name="REQ" >
    <ST Text="  OUT := (IN1 &#38; IN2);&#10;" />
  </Algorithm>
  </BasicFB>
</FBType>
```

C.2 Service interface function block types

EXAMPLE 1 A service interface function block type for sensing the presence and condition of an orange on a conveyor could be expressed textually as follows.

```
FUNCTION_BLOCK ORANGE_SENSOR (* Sense Presence & Color of Orange *)
EVENT_INPUT
    INIT WITH THRESH; (* Set Threshold *)
END_EVENT
EVENT_OUTPUT
    INITO; (* Threshold Set *)
    IND WITH PRESENT, GREEN; (* Change in Presence or Color *)
END_EVENT
VAR_INPUT
    THRESH : REAL; (* Adjustable Color Threshold *)
END_VAR
VAR_OUTPUT
    PRESENT : BOOL; (* Orange is Present *)
    GREEN : BOOL; (* Green is Above Threshold *)
END_VAR
SERVICE ORANGE_SENSOR/resource
SEQUENCE setup
    ORANGE_SENSOR.INIT(THRESH) -> resource.init() ->
ORANGE_SENSOR.INITO();
END_SEQUENCE
SEQUENCE sensing
    resource.change() -> ORANGE_SENSOR.IND(PRESENT,GREEN);
END_SEQUENCE
END_SERVICE
END_FUNCTION_BLOCK
```

The interface and service sequences would appear graphically as follows:



A corresponding XML document would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="ORANGE_SENSOR" Comment="Sense Presence &#38; Color of Orange" >
  <Identification Classification="C0202" ApplicationDomain="Food Processing"
Function="Detection" Type="Photoelectric Sensors" Description="Orange Presence and
Quality" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-05-
14" Remarks="Modified to use LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-01-
26" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="INIT" Comment="Set Threshold" >
        <With Var="THRESH" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="INITO" Comment="Threshold Set" >
      </Event>
      <Event Name="IND" Comment="Change in Presence or Color" >
        <With Var="PRESENT" />
        <With Var="GREEN" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="THRESH" Type="REAL" Comment="Adjustable Color Threshold"
/>
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="PRESENT" Type="BOOL" Comment="Orange is Present" />
      <VarDeclaration Name="GREEN" Type="BOOL" Comment="Green is Above Threshold" />
    </OutputVars>
  </InterfaceList>
  <Service RightInterface="resource" LeftInterface="ORANGE_SENSOR" >
    <ServiceSequence Name="setup" >
      <ServiceTransaction >
        <InputPrimitive Interface="ORANGE_SENSOR" Event="INIT" Parameters="THRESH"
/>
        <OutputPrimitive Interface="resource" Event="init" />
        <OutputPrimitive Interface="ORANGE_SENSOR" Event="INITO" />
      </ServiceTransaction>
    </ServiceSequence>
    <ServiceSequence Name="sensing" >
      <ServiceTransaction >
        <InputPrimitive Interface="resource" Event="change" />
        <OutputPrimitive Interface="ORANGE_SENSOR" Event="IND"
Parameters="PRESENT,GREEN" />
      </ServiceTransaction>
    </ServiceSequence>
  </Service>
</FBType>
```