BS EN 61499-2:2013

BSI Standards Publication

Function blocks

Part 2: Software tool requirements

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

raising standards worldwide™

National foreword

This British Standard is the UK implementation of EN 61499-2:2013. It is identical to IEC 61499-2:2012. It supersedes BS EN 61499-2:2005 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee GEL/65, Measurement and control.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013

Published by BSI Standards Limited 2013

ISBN 978 0 580 79809 2

ICS 25.040; 35.240.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2013.

Amendments issued since publication

Amd. No.	Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 61499-2

February 2013

ICS 25.040; 35.240.50

Supersedes EN 61499-2:2005

English version

Function blocks - Part 2: Software tool requirements

(IEC 61499-2:2012)

Blocs fonctionnels - Partie 2: Exigences pour les outils logiciels (CEI 61499-2:2012)

Funktionsbausteine für industrielle Leitsysteme - Teil 2: Anforderungen an Software-Werkzeuge (IEC 61499-2:2012)

This European Standard was approved by CENELEC on 2012-12-12. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Management Centre: Avenue Marnix 17, B - 1000 Brussels

© 2013 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Ref. No. EN 61499-2:2013 E

Foreword

The text of document 65B/846/FDIS, future edition 2 of IEC 61499-2, prepared by IEC/TC 65B "Measurement and control devices" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61499-2:2013.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2013-09-12
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2015-12-12

This document supersedes EN 61499-2:2005.

EN 61499-2:2013 includes the following significant technical changes with respect to EN 61499-2:2005:

- the contents of Annex A have been updated to conform to the technical changes of the second edition of EN 61499-1;
- CDATA sections are now allowed for the textual contents of algorithms in Tables A.4 and A.5.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 61499-2:2012 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following note has to be added for the standard indicated :

IEC 61499-4	NOTE	Harmonised as EN 61499-4.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 61131-3	2003	Programmable controllers - Part 3: Programming languages	EN 61131-3	2003
IEC 61499-1	2012	Function blocks - Part 1: Architecture	EN 61499-1	2013
ISO/IEC 8824	Series	Information technology - Abstract Syntax Notation One (ASN.1)	-	-

CONTENTS

INTRODUCTION

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 (withdrawn)
- Part 4: Rules for compliance profiles

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 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 IEC 61499-1.

It is beyond the scope of this standard 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 of IEC 61499.

2 Normative references

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

IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

IEC 61499-1:2012, *Function blocks – Part 1: Architecture*

ISO/IEC 8824 (all parts), *Information technology – Abstract Syntax Notation One (ASN.1)*

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, segment type,* or *system configuration*

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:

- the type or types of *library element* to which the software tool applies;
- 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 the XML syntax defined in Annex D.

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 interfaces, data interfaces* and *services* as defined in IEC 61499-1;
- *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.

Software tools may provide additional features, beyond those illustrated in IEC 61499-1, in the graphic display of declarations.

EXAMPLE 1 In the display of an Execution Control Chart (ECC), the tool may provide, along with the display of each transition, a cardinal number indicating the order (as defined in IEC 61499-1) in which the transition is evaluated.

EXAMPLE 2 A software tool may provide means of navigating a *mapping* from the display of a function block instance in an *application* to its corresponding display in a *resource*, and vice versa.

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.

EXAMPLE The software tool may provide convenient means for the user to change the order in which declarations are listed in their textual representation, for instance in a list of transitions in an Execution Control Chart (ECC), without the user having to edit the textual representation by manual means such as "cut and paste".

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:

- Checking the correctness of the syntax of declarations.
- 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.*
- 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:

- the production of an executable code ("firmware") for embedding in instances of resource types and device types;
- the creation and interconnection ("downloading") of function block instances in resources and devices, for instance by using the management facilities defined in subclause 6.3 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:

- the facilities described in preceding subclauses of this Clause;
- the information exchange facilities defined in 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 at 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.2.1, 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.

NOTE 1 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 prevail.

NOTE 2 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 standard.

A.2 DataType DTD

An XML document complying with the DTD in Table A.2 represents a **DataTypeDeclaration** object as described in C.2.2 of IEC 61499-1.

Table A.2 – DataType DTD (1 of 2)

```
<?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>
```

Table A.2 (2 of 2)

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

```
<!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 above DTD, and (where applicable) references to the formal syntax for their attributes, are given in Table A.3.

Table A.3 – DataType DTD elements (1 of 2)

Element attributes	Textual syntax (IEC 61131-3, Annex B)	Explanation
DataType		See IEC 61131-3
Name	data_type_name	
Comment	-- A comment 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	
VersionInfo	-- Possibly one of several entries: First entry – Most recent version Second entry – Immediately preceding released version... 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	
Class	ASN.1 tag class per ISO/IEC 8824	
Number	ASN.1 tag number per ISO/IEC 8824	
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, item 1	
BaseType	elementary_type_name	
InitialValue	constant	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	

Table A.3 (2 of 2)

Element Attributes	Textual Syntax (IEC 61131-3, Annex B)	Explanation
EnumeratedType	See IEC 61131-3 Tables 12 and 14, item 2	
InitialValue	enumerated_value	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
EnumeratedValue	See IEC 61131-3 Table 14, item 2	
Name	enumerated_value	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
SubrangeType	--	See IEC 61131-3 Tables 12 and 14, item 3
BaseType	integer_type_name	
InitialValue	signed_integer	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
Subrange	See IEC 61131-3 Tables 12 and 14, item 3	
LowerLimit	signed_integer	
UpperLimit	signed_integer	
ArrayType	See IEC 61131-3 Tables 12 and 14, item 4	
BaseType	non_generic_type_name	
InitialValues	array_initialization	
StructuredType	See IEC 61131-3 Tables 12, item 5 and 14, item 5 and item 6	
VarDeclaration		
Name	structure_element_name	
Type	non_generic_type_name	
ArraySize	[a]	
InitialValue	[b]	
Comment	A comment 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 per IEC 61131-3 without (* and *) delimiters	

[a] The syntax of this attribute 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 attribute is missing, the structure component is not an anonymously defined array.

[b] 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 SubapplicationTypeDeclaration** of the abstract superclass **LibraryElement**, respectively. The **DataTypeDeclaration** subclass is represented separately by the DTD given in Clause A.2.

Table A.4 – Library Element DTD (1 of 5)

```
<?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>
```

```
<!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
>
```

Table A.4 (2 of 5)

```
<!-- 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>
```

```
<!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>
```

Table A.4 (3 of 5)

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

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

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

```
<!ELEMENT ST (#PCDATA)>
<!ATTLIST ST
 Text CDATA #IMPLIED >
```

```
<!ELEMENT LD (Rung+)>
<!ELEMENT Rung (#PCDATA)>
<!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>
```

```
<!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>
```

Table A.4 (4 of 5)

```
<!-- 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>
```

```
<!-- System elements -->
<!ELEMENT System (Identification?, VersionInfo+, CompilerInfo?,
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>
```

Table A.4 (5 of 5)

```
<!-- 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>
```

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

<!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 (1 of 5)

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 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	c) [1]	
Destination	c)	
dx1, dx2, dy	See Annex B.	
FBType	A *function block type* as described in IEC 61499-1.	
Name	fb_type_name	
Comment	A comment per IEC 61131 without (* and *) delimiters	
Event	A *declaration* of an *event interface.*	
Name	event_input_name | event_output_name	d)
Type	event_type	
Comment	A comment 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	d)
VarDeclaration	A declaration of a variable.	
Name	input_variable_name | output_variable_name | internal_variable_name	e)
Type	identifier	
ArraySize	f)	
InitialValue	g)	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	

[1] See Table footnote at the end of Table A.1.

Table A.5 (2 of 5)

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
AdapterDeclaration	A declaration of a plug or socket interface of a function block type.	
Name	plug_name | socket_name	h) [1]
Type	adapter_type_name	
Comment	A comment 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 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	
Algorithm	An *algorithm* in a specified language a)	
Name	algorithm_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
ST	An *algorithm* in the IEC 61131-3 ST language.	
Text	Algorithm contents in the syntax of a statement_list per IEC 61131-3, Annex B.	l)
Rung	A rung of an algorithm in the LD language	
Output	b)	
Expression	See b) and l).	
Other	An algorithm in a language other than FBD, ST or LD.	
Language	The name of the programming language	a)
Text	Content of the algorithm in the syntax defined for the particular language	See a) and l).
Service	A *declaration* of a *service* per IEC 61499-1	
RightInterface	service_interface_name	
LeftInterface	service_interface_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	

[1] See Table footnote at the end of Table A.1.

Table A.5 (3 of 5)

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
ServiceSequence	A declaration of a service sequence per IEC 61499-1	
Name	sequence_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
InputPrimitive	An input *service primitive* 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* 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 per IEC 61131-3 without (* and *) delimiters	
ResourceType	A *declaration* of a *resource type* per IEC 61499-1	
Name	resource_type_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
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 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 per IEC 61131-3 without (* and *) delimiters	

Table A.5 (4 of 5)

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
System	A declaration of a system configuration per IEC 61499-1	
Name	system_name	
Comment	A comment 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 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 j) [1].
Device	A declaration of a device configuration per IEC 61499-1	
Name	device_instance_name	
Type	device_type_name	
Comment	A comment 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 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	i)
Type	event_type	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
SubApp	A *subapplication instance* as defined in IEC 61499-1.	
Name	subapp_instance_name	
Type	subapp_type_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	
x, y	See Annex B.	
SegmentType	A *declaration* of a *segment type* per IEC 61499-1	
Name	segment_type_name	
Comment	A comment per IEC 61131-3 without (* and *) delimiters	

[1] See Table footnote at the end of Table A.1.

Table A.5 (5 of 5)

Element Attributes	Syntax (IEC 61499-1, Annex B)	Explanation
Segment	A segment of a communication *network*	
Name	identifier	
Type	identifier	
Comment	A comment 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	k)
SegmentName	identifier	The segment to be linked
Comment	A comment 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 per IEC 61131-3 without (* and *) delimiters	

a) The specification of algorithms in languages other than FBD, ST and LD is beyond the scope of this standard.

b) 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 standard 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 in Clause C.1 for an illustration of this recommended usage.

c) 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.

d) 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.

e) 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.

f) 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.

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

h) The productions plug_name and socket_name apply when the associated

`AdapterDeclaration` element is part of a `Plugs` or `Sockets` element, respectively.

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

j) 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.

k) 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`.

l) To increase readability and reduce the probability of errors in encoding and parsing data, the use of CDATA sections instead of textual attributes is recommended for these elements.

Annex B
(informative)

Graphics model

B.1 Coordinate system

This Annex presents a simple graphics 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 **LibraryElement** 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.

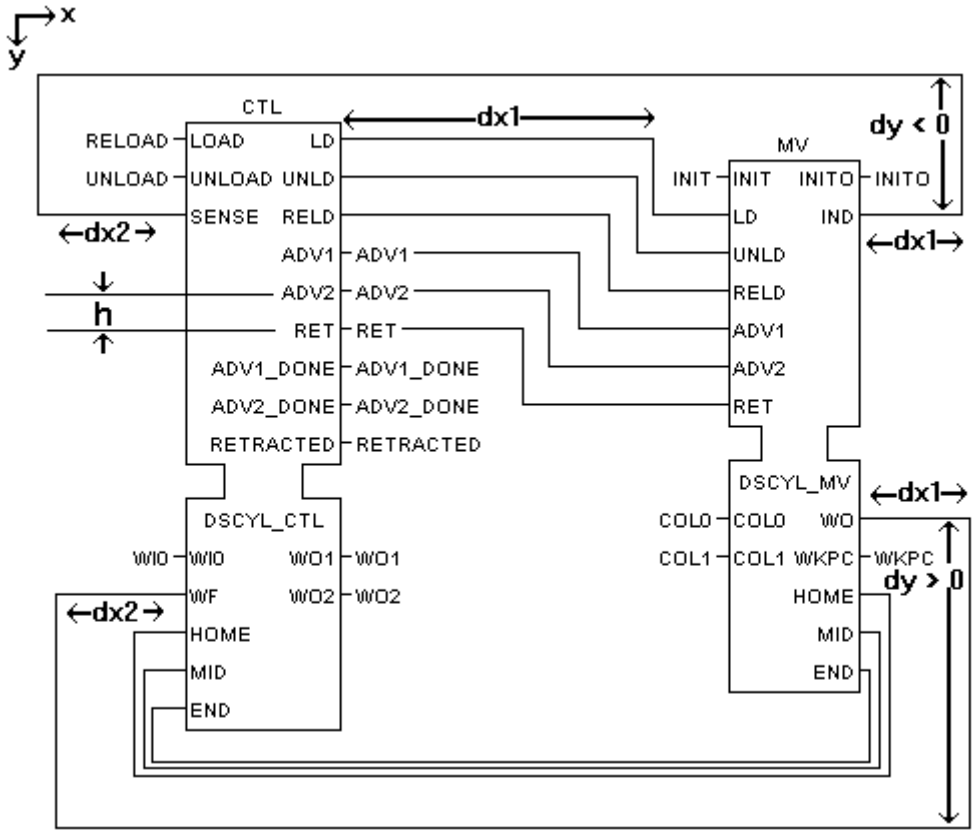


Figure B.1 – Graphics model

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 1 000 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 according to the following guidelines.

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 at 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 at a distance **dx1** from the right edge of the function block which provides the source of the connection; thence vertically at a distance **dy**; thence horizontally to an x-coordinate at a distance **dx2** left of the left edge of the function block which provides the destination of the connection; thence vertically at 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 Clause 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.

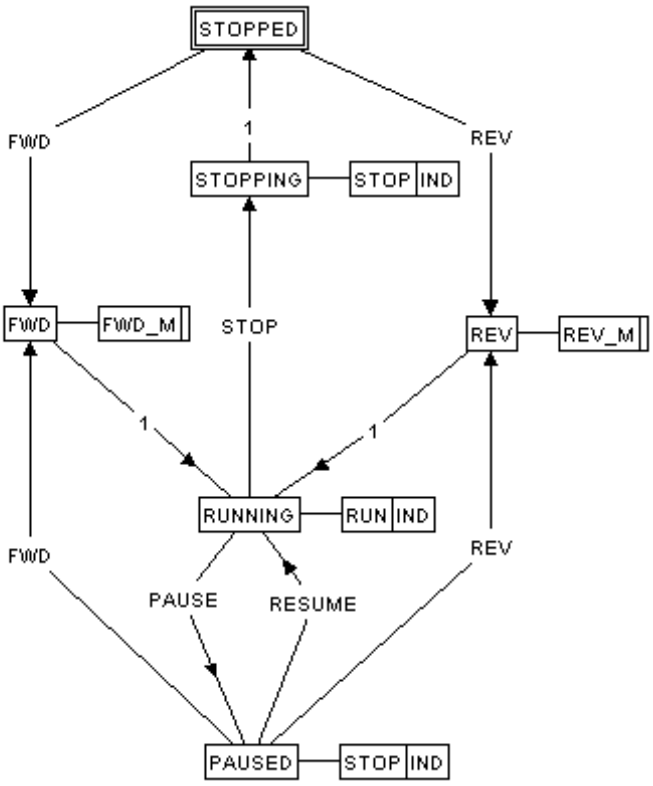


Figure B.2 – ECC drawing example

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 Clause 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)

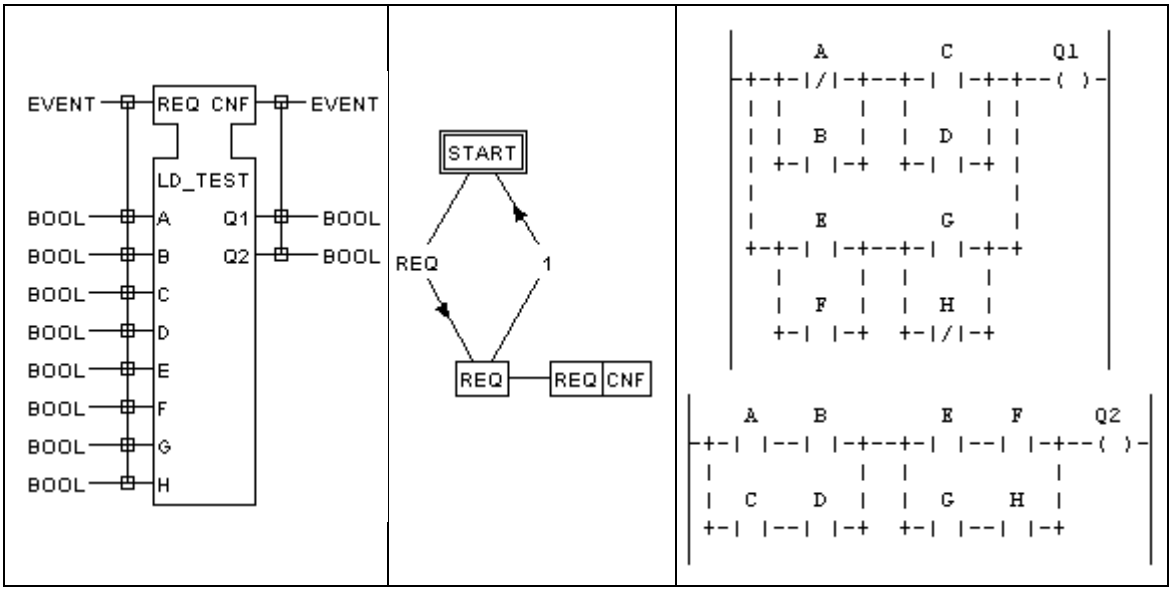
Examples

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.



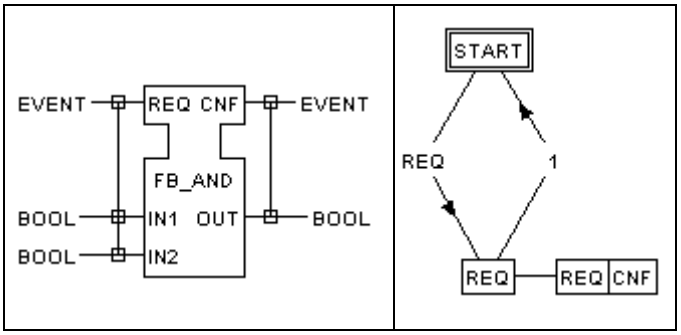
A corresponding XML document would be:

```
<?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:

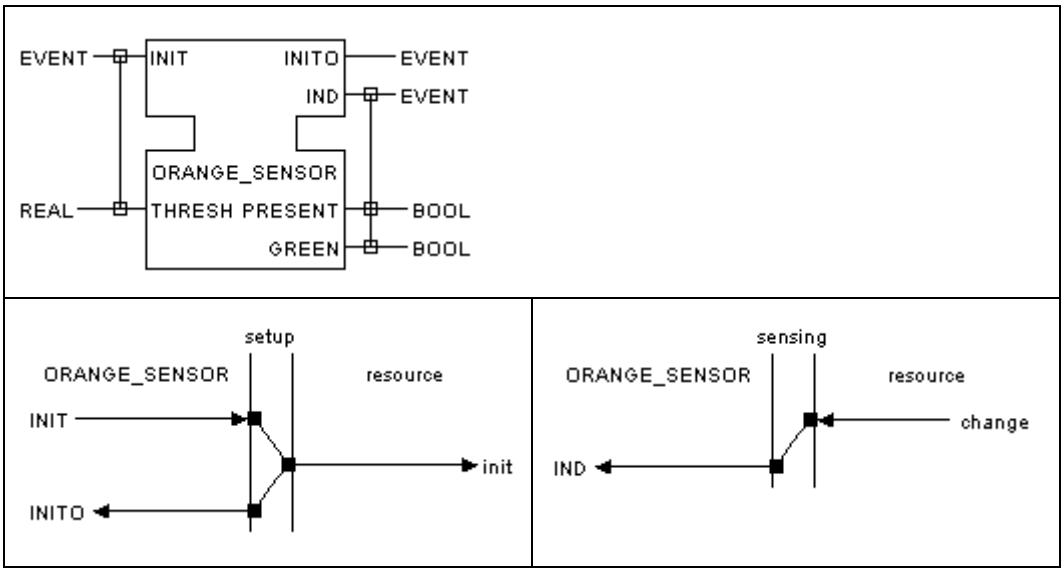
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="FB_AND" Comment="Boolean AND" >
  <Identification Standard="61499-1-D.1" Classification="Math"
ApplicationDomain="Any" Function="AND" Type="Boolean" />
  <VersionInfo Organization="IEC TC65/WG6" 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&amp;#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 &amp;#38; IN2);&amp;#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:



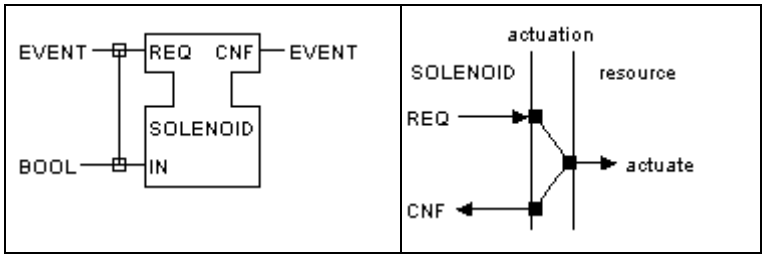
A corresponding XML document would be:

```
<?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>
```

EXAMPLE 2 A service interface function block type for the actuator of a simple solenoid valve could be expressed textually as follows.

```
FUNCTION_BLOCK SOLENOID (* Solenoid Valve *)
EVENT_INPUT
    REQ WITH IN; (* Set Actuator Status *)
END_EVENT
EVENT_OUTPUT
    CNF; (* Actuator Status Change Confirmed *)
END_EVENT
VAR_INPUT
    IN: BOOL; (* Actuator Value,1=OPEN,0=CLOSED *)
END_VAR
SERVICE SOLENOID/resource
SEQUENCE actuation
    SOLENOID.REQ(IN) -> resource.actuate(IN) -> SOLENOID.CNF();
END_SEQUENCE
END_SERVICE
END_FUNCTION_BLOCK
```

The interface and service sequence would appear graphically as follows.

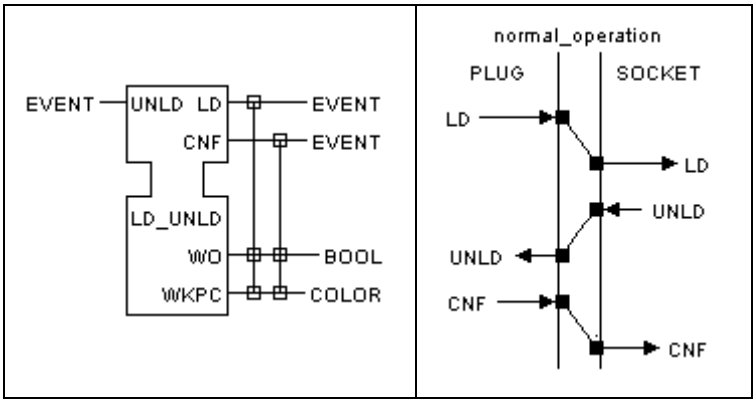


A corresponding XML document would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="SOLENOID" Comment="Solenoid Valve" >
  <Identification Classification="C0403" ApplicationDomain="Any" Function="Logic / I/O
Modules &#38; Controllers" Type="Actuators" Description="Solenoid Valve" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-02-
03" Remarks="Corrected service sequence" />
  <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="REQ" Comment="Set Actuator Status" >
        <With Var="IN" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" Comment="Actuator Status Change Confirmed" >
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="IN" Type="BOOL" Comment="Actuator Value,1=OPEN,0=CLOSED"
/>
    </InputVars>
  </InterfaceList>
  <Service RightInterface="resource" LeftInterface="SOLENOID" >
    <ServiceSequence Name="actuation" >
      <ServiceTransaction >
        <InputPrimitive Interface="SOLENOID" Event="REQ" Parameters="IN" />
        <OutputPrimitive Interface="resource" Event="actuate" Parameters="IN" />
        <OutputPrimitive Interface="SOLENOID" Event="CNF" />
      </ServiceTransaction>
    </ServiceSequence>
  </Service>
</FBType>
```

C.3 An adapter interface type

EXAMPLE An adapter interface for use in parts transfer simulations, its typical sequence of operation, and its corresponding XML document, may be as shown below.



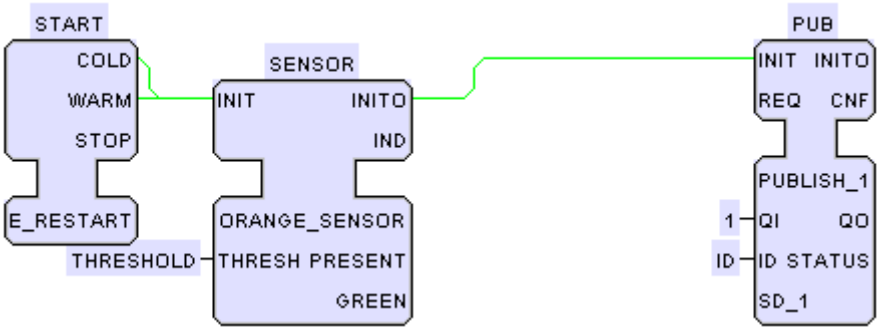
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AdapterType SYSTEM "../LibraryElement.dtd" >
<AdapterType Name="LD_UNLD" Comment="LOAD/UNLOAD Adapter Interface" >
  <Identification Standard="IEC 61499-2" />
  <VersionInfo Organization="Rockwell Automation" Version="0.0" Author="JHC"
Date="1999-11-17" Remarks="Generated by FBEditor application" />
  <CompilerInfo header="package fb.rt.omac;" >
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge" Version="2.0" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="UNLD" Comment="UNLOAD Request" >
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="LD" Comment="LOAD Request" >
        <With Var="WO" />
        <With Var="WKPC" />
      </Event>
      <Event Name="CNF" Comment="UNLD Confirm" >
        <With Var="WO" />
        <With Var="WKPC" />
      </Event>
    </EventOutputs>
    <OutputVars>
      <VarDeclaration Name="WO" Type="BOOL" Comment="Workpiece present" />
      <VarDeclaration Name="WKPC" Type="COLOR" Comment="Workpiece Color" />
    </OutputVars>
  </InterfaceList>
  <Service RightInterface="SOCKET" LeftInterface="PLUG" >
    <ServiceSequence Name="normal_operation" >
      <ServiceTransaction >
        <InputPrimitive Interface="PLUG" Event="LD" Parameters="WO,WKPC" />
        <OutputPrimitive Interface="SOCKET" Event="LD" Parameters="WO,WKPC" />
      </ServiceTransaction>
      <ServiceTransaction >
        <InputPrimitive Interface="SOCKET" Event="UNLD" />
        <OutputPrimitive Interface="PLUG" Event="UNLD" />
      </ServiceTransaction>
      <ServiceTransaction >
        <InputPrimitive Interface="PLUG" Event="CNF" />
        <OutputPrimitive Interface="SOCKET" Event="CNF" />
      </ServiceTransaction>
    </ServiceSequence>
  </Service>
</AdapterType>
```

C.4 Resource types

EXAMPLE 1 A resource type containing an instance of the ORANGE_SENSOR function block type defined in Clause C.2, plus an instance of the PUBLISH_1 type to transmit a change in condition, could be declared textually as shown below. This resource type also contains an instance of the E_RESTART type defined in Annex A of IEC 61499-1, interconnected to provide initialization of the other function block instances. The data outputs of the ORANGE_SENSOR block and the SD_1 input of the PUBLISH_1 block, and their corresponding event inputs and outputs, are left unconnected in order to allow application-specific logic to determine the value to be transmitted and the event to trigger the transmission.

```
  RESOURCE_TYPE ORANGE_RES (* A Configurable Orange Presence&Quality
  Sensor *)
  VAR_INPUT
      THRESHOLD: REAL; (* Adjustable Color Threshold *)
      ID: WSTRING; (* UDP Channel ID *)
  END_VAR
  FB_TYPES
    E_RESTART;
    ORANGE_SENSOR;
    PUBLISH_1;
  END_FB_TYPES
  FBS
      START: E_RESTART;
      SENSOR: ORANGE_SENSOR;
      PUB: PUBLISH_1(
          QI:= 1);
  END_FBS
  EVENT_CONNECTIONS
      START.COLD TO SENSOR.INIT;
      START.WARM TO SENSOR.INIT;
      SENSOR.INITO TO PUB.INIT;
  END_CONNECTIONS
  DATA_CONNECTIONS
      THRESHOLD TO SENSOR.THRESH;
      ID TO PUB.ID;
  END_CONNECTIONS
END_RESOURCE_TYPE
```

A graphical representation of this resource type's function block network is:



The resource type's external interface could be represented as:

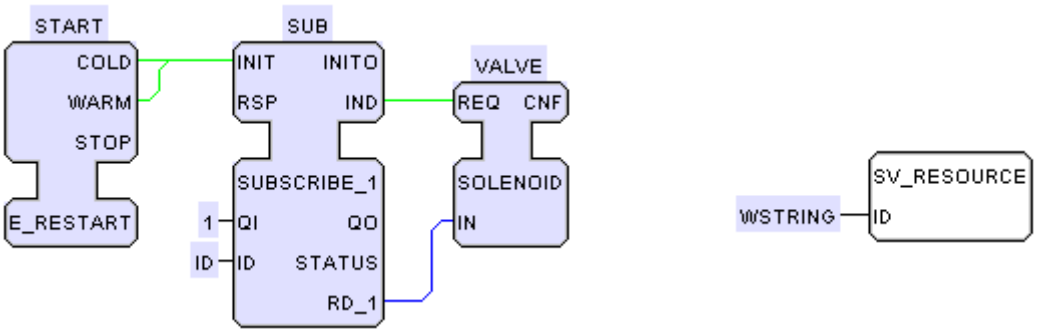
An equivalent XML document (with additional information for software tools) could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ResourceType SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd" >
<ResourceType Name="ORANGE_RES" Comment="A Configurable Orange Presence&#38;Quality
Sensor" >
  <Identification Standard="IEC 61499-1" Classification="C0202"
ApplicationDomain="Food Processing" Function="Detection" Type="Photoelectric Sensors"
Description="Orange Presence and Quality" />
  <VersionInfo Organization="IEC SC65B/WG15" Version="0.2" Author="JHC" Date="2011-02-
23" Remarks="Added THRESHOLD, ID parameters." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-06-
20" Remarks="Corrected &#34;FBType&#34; to &#34;FBTypeName&#34;." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <FBTypeName Name="E_RESTART" />
  <FBTypeName Name="ORANGE_SENSOR" />
  <FBTypeName Name="PUBLISH_1" />
  <VarDeclaration Name="THRESHOLD" Type="REAL" Comment="Adjustable Color Threshold" />
  <VarDeclaration Name="ID" Type="WSTRING" Comment="UDP Channel ID" />
  <FBNetwork >
    <FB Name="START" Type="E_RESTART" x="94.44444" y="11.111111" >
    </FB>
    <FB Name="SENSOR" Type="ORANGE_SENSOR" x="672.22217" y="122.22221" >
    </FB>
    <FB Name="PUB" Type="PUBLISH_1" x="2172.2222" y="11.111111" >
      <Parameter Name="QI" Value="1" />
    </FB>
    <EventConnections>
      <Connection Source="START.COLD" Destination="SENSOR.INIT" dx1="33.333332"
dx2="47.0588" dy="-70.5882" />
      <Connection Source="START.WARM" Destination="SENSOR.INIT" dx1="61.111107"
dx2="76.4706" dy="-188.2353" />
      <Connection Source="SENSOR.INITO" Destination="PUB.INIT" dx1="172.22221" />
    </EventConnections>
    <DataConnections>
      <Connection Source="THRESHOLD" Destination="SENSOR.THRESH" />
      <Connection Source="ID" Destination="PUB.ID" />
    </DataConnections>
  </FBNetwork>
</ResourceType>
```

EXAMPLE 2 A resource type containing an instance of the SOLENOID function block type defined in Clause C.2, plus an instance of the SUBSCRIBE_1 type to receive a command to change solenoid status, could be declared textually as shown below. This resource type also contains an instance of the E_RESTART type defined in Annex A of IEC 61499-1, interconnected to provide initialization of the other function block instances.

```
  RESOURCE_TYPE SV_RESOURCE (* A Remotely Activated
  Solenoid Valve Resource *)
  VAR_INPUT
      ID: WSTRING; (* UDP Channel ID *)
  END_VAR
  FBS
      START: E_RESTART;
      SUB: SUBSCRIBE_1(
          QI:= 1);
      VALVE: SOLENOID;
  END_FBS
  EVENT_CONNECTIONS
      START.COLD TO SUB.INIT;
      START.WARM TO SUB.INIT;
      SUB.IND TO VALVE.REQ;
  END_CONNECTIONS
  DATA_CONNECTIONS
      SUB.RD_1 TO VALVE.IN;
      ID TO SUB.ID;
  END_CONNECTIONS
END_RESOURCE_TYPE
```

Graphical representations of this resource's function block network and external interface can be as follows:



An equivalent XML document (with additional information for software tools) could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ResourceType SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd" >
<ResourceType Name="SV_RESOURCE" Comment="A Remotely Activated Solenoid Valve
Resource" >
  <Identification Standard="IEC 61499-1" Classification="C0403"
ApplicationDomain="Any" Function="Logic / I/O Modules &#38; Controllers"
Type="Actuators" Description="Solenoid Valve" />
  <VersionInfo Organization="IEC SC65B/WG15" Version="0.2" Author="JHC" Date="2011-02-
23" Remarks="Added ID parameter." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-06-
20" Remarks="Now uses LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <VarDeclaration Name="ID" Type="WSTRING" Comment="UDP Channel ID" />
  <FBNetwork >
    <FB Name="START" Type="E_RESTART" x="72.22222" y="11.111111" >
    </FB>
    <FB Name="SUB" Type="SUBSCRIBE_1" x="705.55554" y="11.111111" >
      <Parameter Name="QI" Value="1" />
    </FB>
    <FB Name="VALVE" Type="SOLENOID" x="1316.6666" y="122.22221" >
    </FB>
    <EventConnections>
      <Connection Source="START.COLD" Destination="SUB.INIT" dx1="33.333332"
dx2="41.1765" dy="-64.7059" />
      <Connection Source="START.WARM" Destination="SUB.INIT" dx1="61.111107"
dx2="64.7059" dy="-182.3529" />
      <Connection Source="SUB.IND" Destination="VALVE.REQ" dx1="38.888885" />
    </EventConnections>
    <DataConnections>
      <Connection Source="SUB.RD_1" Destination="VALVE.IN" dx1="133.33333" />
      <Connection Source="ID" Destination="SUB.ID" />
    </DataConnections>
  </FBNetwork>
</ResourceType>
```

C.5 Device types

EXAMPLE 1 A device type containing an instance of the ORANGE_RES resource type defined in Clause C.4 could be declared textually as shown below.

```
DEVICE_TYPE ORANGE_EYE
    (* Programmable Orange Presence+Quality Sensor *)
RESOURCE_TYPES
  ORANGE_RES;
END_RESOURCE_TYPES
RESOURCE R1: ORANGE_RES
END_RESOURCE
END_DEVICE_TYPE
```

An equivalent XML document (with additional information for software tools) could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeviceType SYSTEM "../LibraryElement.dtd" >
<DeviceType Name="ORANGE_EYE" Comment="Programmable Orange Presence+Quality Sensor" >
  <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="2000-06-
20" Remarks="Corrected &#34;ResourceTypeName&#34;" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <ResourceTypeName Name="ORANGE_RES" />
  <Resource Name="R1" Type="ORANGE_RES" >
  </Resource>
</DeviceType>
```

EXAMPLE 2 A device type containing an instance of the SV_RESOURCE type defined in Clause C.4 could be declared textually as shown below.

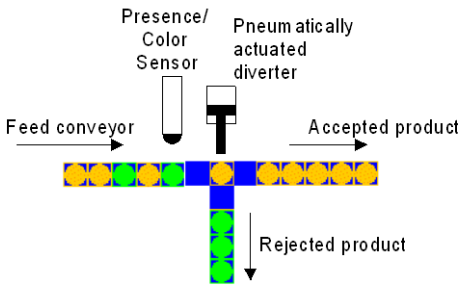
```
DEVICE_TYPE SOLENOID_VALVE
   (* A Remotely Activated Solenoid Valve *)
RESOURCE R1: SV_RESOURCE
END_RESOURCE
END_DEVICE_TYPE
```

An equivalent XML document (with additional information for software tools) could be:

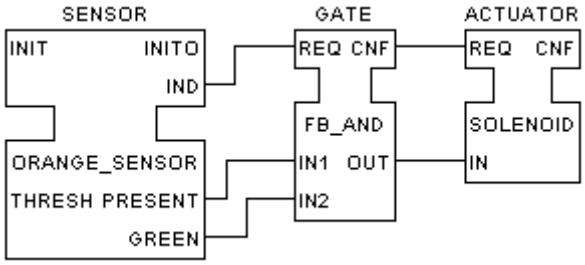
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DeviceType SYSTEM "../LibraryElement.dtd" >
<DeviceType Name="SOLENOID_VALVE" Comment="A Remotely
Activated Solenoid Valve" >
  <Identification Classification="C0403"
ApplicationDomain="Any" Function="Logic / I/O Modules
&#38; Controllers" Type="Actuators"
Description="Solenoid Valve" />
  <VersionInfo Organization="IEC TC65/WG6"
Version="0.1" Author="JHC" Date="2000-06-20"
Remarks="Now uses LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6"
Version="0.0" Author="JHC" Date="2000-02-02" />
  <CompilerInfo header="package fb.rt.part2;" >
  </CompilerInfo>
  <Resource Name="R1" Type="SV_RESOURCE" >
  </Resource>
</DeviceType>
```

C.6 A system configuration

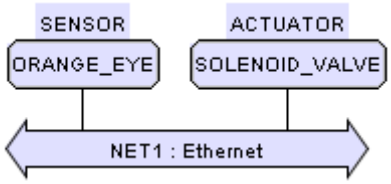
EXAMPLE Oranges are to be sorted by sensing their position and color and diverting any which are too green, as illustrated below.



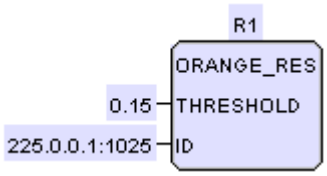
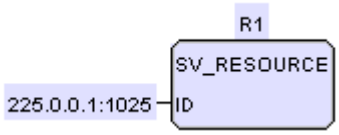
An *application* which can perform this task, using the ORANGE_SENSOR and SOLENOID function block types described in Clause C.2, could be as follows.



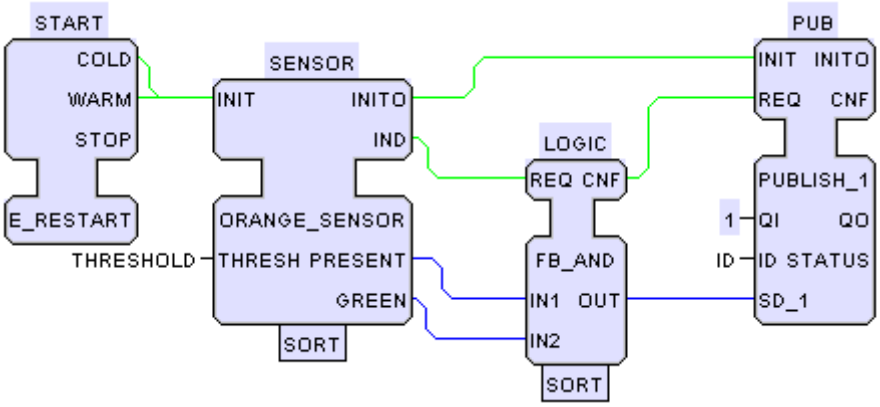
A *system* to implement this application, using the previously defined ORANGE_EYE and SOLENOID_VALVE device types, in conjunction with the Ethernet segment type defined in Clause C.7, could have the following top-level *configuration*:



The SENSOR:ORANGE_EYE and ACTUATOR:SOLENOID_VALVE devices could in turn be configured as follows:

SENSOR:ORANGE_EYE	ACTUATOR:SOLENOID_VALVE
	

The resource ACTUATOR.R1:SV_RESOURCE can be used unmodified as defined in Clause C.4, while the resource SENSOR.R1:ORANGE_RES can be configured as follows to implement the application logic. Note the additional tags at the bottom of the SENSOR and LOGIC blocks, indicating that they are associated with the SORT application; this is a "navigation" feature as described in EXAMPLE 2 of 4.4.


START
COLD
WARM
STOP
E_RESTART
SENSOR
INIT
INITO
IND
ORANGE_SENSOR
THRESHOLD
THRESH PRESENT
GREEN
SORT
LOGIC
REQ CNF
FB_AND
IN1 OUT
IN2
SORT
PUB
INIT INITO
REQ CNF
PUBLISH_1
1
QI QO
ID
ID STATUS
SD_1

A *system configuration* implementing the features described above could appear as follows in textual form:

```
SYSTEM ORANGE_SORTER (* Orange Sorter System Configuration *)
APPLICATION SORT
FBS
    SENSOR: ORANGE_SENSOR;
    GATE: FB_AND;
    ACTUATOR: SOLENOID;
END_FBS
EVENT_CONNECTIONS
    SENSOR.IND TO GATE.REQ;
    GATE.CNF TO ACTUATOR.REQ;
END_CONNECTIONS
DATA_CONNECTIONS
    SENSOR.PRESENT TO GATE.IN1;
    SENSOR.GREEN TO GATE.IN2;
    GATE.OUT TO ACTUATOR.IN;
END_CONNECTIONS
END_APPLICATION
DEVICE SENSOR: ORANGE_EYE
RESOURCE R1: ORANGE_RES(
        THRESHOLD:= 0.15,
        ID:= 225.0.0.1:1025)
FBS
    LOGIC: FB_AND;
END_FBS
EVENT_CONNECTIONS
    SENSOR.IND TO LOGIC.REQ;
    LOGIC.CNF TO PUB.REQ;
END_CONNECTIONS
DATA_CONNECTIONS
    SENSOR.PRESENT TO LOGIC.IN1;
    SENSOR.GREEN TO LOGIC.IN2;
    LOGIC.OUT TO PUB.SD_1;
END_CONNECTIONS
END_RESOURCE
END_DEVICE
DEVICE ACTUATOR: SOLENOID_VALVE
RESOURCE R1: SV_RESOURCE(
        ID:= 225.0.0.1:1025)
END_RESOURCE
END_DEVICE
MAPPING
 SORT.SENSOR ON SENSOR.R1.SENSOR;
 SORT.GATE ON SENSOR.R1.LOGIC;
 SORT.ACTUATOR ON ACTUATOR.R1.VALVE;
END_MAPPING
SEGMENTS
    NET1: Ethernet;
END_SEGMENTS
LINKS
    SENSOR => NET1;
    ACTUATOR => NET1;
END_LINKS
END_SYSTEM
```

A corresponding XML document would then be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE System SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd" >
<System Name="ORANGE_SORTER" Comment="Orange Sorter System Configuration" >
  <Identification Standard="61499-2" />
  <VersionInfo Organization="IEC SC65B/WG15" Version="0.2" Author="JHC" Date="2011-02-
23" Remarks="Moved parameters to Resource level." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-06-
20" Remarks="Now uses LibraryElement.dtd" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
03" />
  <Application Name="SORT" >
    <SubAppNetwork >
      <FB Name="SENSOR" Type="ORANGE_SENSOR" x="94.1176" y="11.7647" >
      </FB>
      <FB Name="GATE" Type="FB_AND" x="941.1765" y="11.7647" >
      </FB>
      <FB Name="ACTUATOR" Type="SOLENOID" x="1441.1764" y="11.7647" >
      </FB>
      <EventConnections>
        <Connection Source="SENSOR.IND" Destination="GATE.REQ" dx1="88.2353"
dx2="47.0588" dy="-188.2353" />
        <Connection Source="GATE.CNF" Destination="ACTUATOR.REQ" dx1="41.1765" />
      </EventConnections>
      <DataConnections>
        <Connection Source="SENSOR.PRESENT" Destination="GATE.IN1" dx1="64.7059"
dx2="176.4706" dy="982.3529" />
        <Connection Source="SENSOR.GREEN" Destination="GATE.IN2" dx1="123.5294"
dx2="158.8235" dy="870.5882" />
        <Connection Source="GATE.OUT" Destination="ACTUATOR.IN" dx1="152.9412" />
      </DataConnections>
    </SubAppNetwork>
  </Application>
  <Device Name="SENSOR" Type="ORANGE_EYE" x="122.22221" y="11.111111" >
    <Resource Name="R1" Type="ORANGE_RES" x="577.7778" y="11.111111" >
      <Parameter Name="THRESHOLD" Value="0.15" />
      <Parameter Name="ID" Value="225.0.0.1:1025" />
      <FBNetwork >
        <FB Name="LOGIC" Type="FB_AND" x="1538.8888" y="344.44443" >
        </FB>
        <EventConnections>
          <Connection Source="SENSOR.IND" Destination="LOGIC.REQ" dx1="44.444443"
dx2="58.8235" dy="-588.2353" />
          <Connection Source="LOGIC.CNF" Destination="PUB.REQ" dx1="61.111107" />
        </EventConnections>
        <DataConnections>
          <Connection Source="SENSOR.PRESENT" Destination="LOGIC.IN1" dx1="83.33333"
dx2="64.7059" dy="576.4706" />
          <Connection Source="SENSOR.GREEN" Destination="LOGIC.IN2" dx1="38.888885"
dx2="58.8235" dy="458.8235" />
          <Connection Source="LOGIC.OUT" Destination="PUB.SD_1" dx1="100.0" />
        </DataConnections>
      </FBNetwork>
    </Resource>
  </Device>
  <Device Name="ACTUATOR" Type="SOLENOID_VALVE" x="627.7778" y="11.111111" >
    <Resource Name="R1" Type="SV_RESOURCE" x="705.55554" y="22.222221" >
      <Parameter Name="ID" Value="225.0.0.1:1025" />
    </Resource>
  </Device>
  <Mapping From="SORT.SENSOR" To="SENSOR.R1.SENSOR" />
  <Mapping From="SORT.GATE" To="SENSOR.R1.LOGIC" />
  <Mapping From="SORT.ACTUATOR" To="ACTUATOR.R1.VALVE" />
  <Segment Name="NET1" Type="Ethernet" x="616.6666" y="411.11108" dx1="999.99994" >
  </Segment>
  <Link CommResource="SENSOR" SegmentName="NET1" >
  </Link>
  <Link CommResource="ACTUATOR" SegmentName="NET1" >
  </Link>
</System>
```

C.7 A SegmentType definition

EXAMPLE The `Ethernet` segment type used in Clause C.6 can be defined textually and in XML as shown below.

```
SEGMENT_TYPE Ethernet
VAR
    PHY: WSTRING:= "10BASE-T"; (* Physical medium *)
    Speed: REAL:= 10.0; (* Mbits/sec *)
    Length: REAL:= 25.0; (* Length in metres *)
END_VAR
END_SEGMENT_TYPE
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SegmentType SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd" >
<SegmentType Name="Ethernet" >
  <Identification Standard="IEEE 802.3" ApplicationDomain="Networking" />
  <VersionInfo Organization="Holobloc Inc" Version="0.0" Author="JHC"
Date="2011-02-09" />
  <CompilerInfo header="package fb.rt.net;" >
  </CompilerInfo>
  <VarDeclaration Name="PHY" Type="WSTRING" InitialValue="10BASE-T"
Comment="Physical medium" />
  <VarDeclaration Name="Speed" Type="REAL" InitialValue="10.0"
Comment="Mbits/sec" />
  <VarDeclaration Name="Length" Type="REAL" InitialValue="25.0" Comment="Length
in metres" />
</SegmentType>
```

Bibliography

IEC 61499-4, *Function blocks – Part 4: Rules for compliance profiles*

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

BSI is the independent national body responsible for preparing British Standards and other standards-related publications, information and services. It presents the UK view on standards in Europe and at the international level.

BSI is incorporated by Royal Charter. British Standards and other standardisation products are published by BSI Standards Limited.

Revisions

British Standards and PASs are periodically updated by amendment or revision. Users of British Standards and PASs should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using British Standards would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover. Similary for PASs, please notify BSI Customer Services.

Tel: +44 (0)20 8996 9001 Fax: +44 (0)20 8996 7001

BSI offers BSI Subscribing Members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of British Standards and PASs.

Tel: +44 (0)20 8996 7669 Fax: +44 (0)20 8996 7001
Email: plus@bsigroup.com

Buying standards

You may buy PDF and hard copy versions of standards directly using a credit card from the BSI Shop on the website **www.bsigroup.com/shop.** In addition all orders for BSI, international and foreign standards publications can be addressed to BSI Customer Services.

Tel: +44 (0)20 8996 9001 Fax: +44 (0)20 8996 7001
Email: orders@bsigroup.com

In response to orders for international standards, BSI will supply the British Standard implementation of the relevant international standard, unless otherwise requested.

Information on standards

BSI provides a wide range of information on national, European and international standards through its Knowledge Centre.

Tel: +44 (0)20 8996 7004 Fax: +44 (0)20 8996 7005
Email: knowledgecentre@bsigroup.com

BSI Subscribing Members are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration.

Tel: +44 (0)20 8996 7002 Fax: +44 (0)20 8996 7001
Email: membership@bsigroup.com

Information regarding online access to British Standards and PASs via British Standards Online can be found at **www.bsigroup.com/BSOL**

Further information about British Standards is available on the BSI website at **www.bsi-group.com/standards**

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that own copyright in the information used (such as the international standardisation bodies) has formally licensed such information to BSI for commerical publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. This does not preclude the free use, in the course of implementing the standard, of necessary details such as symbols, and size, type or grade designations. If these details are to be used for any other purpose than implementation then the prior written permission of BSI must be obtained. Details and advice can be obtained from the Copyright & Licensing Department.

Tel: +44 (0)20 8996 7070
Email: copyright@bsigroup.com

BSI

389 Chiswick High Road London W4 4AL UK

Tel +44 (0)20 8996 9001
Fax +44 (0)20 8996 7001
www.bsigroup.com/standards

raising standards worldwide™