

Programmable controllers —

Part 5: Communications

The European Standard EN 61131-5:2001 has the status of a British Standard

ICS 17.200.20; 25.040.40

National foreword

This British Standard is the official English language version of EN 61131-5:2001. It is identical with IEC 61131-5:2000.

The UK participation in its preparation was entrusted by Technical Committee GEL/65, Measurement and control, to Subcommittee GEL/65/2, Elements of systems, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

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

From 1 January 1997, all IEC publications have the number 60000 added to the old number. For instance, IEC 27-1 has been renumbered as IEC 60027-1. For a period of time during the change over from one numbering system to the other, publications may contain identifiers from both systems.

Cross-references

Attention is drawn to the fact that CEN and CENELEC Standards normally include an annex which lists normative references to international publications with their corresponding European publications. The British Standards which implement these international or European publications may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

This British Standard, having been prepared under the direction of the Electrotechnical Sector Committee, was published under the authority of the Standards Committee and comes into effect on 15 June 2001

Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 106, an inside back cover and a back cover.

The BSI copyright date displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

© BSI 06-2001

ISBN 0 580 37338 X

EUROPEAN STANDARD

EN 61131-5

NORME EUROPÉENNE

EUROPÄISCHE NORM

February 2001

ICS 25.040.40;25.240.50

English version

**Programmable controllers
Part 5: Communications
(IEC 61131-5:2000)**

Contrôleurs programmables
Partie 5: Communications
(CEI 61131-5:2000)

Speicherprogrammierbare Steuerungen
Teil 5: Kommunikation
(IEC 61131-5:2000)

This European Standard was approved by CENELEC on 2001-01-01. 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 Central Secretariat 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 Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

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

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

© 2001 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Ref. No. EN 61131-5:2001 E



Documents
Document Sharing Hub

Foreword

The text of document 65B/411/FDIS, future edition 1 of IEC 61131-5, prepared by SC 65B, Devices, of IEC TC 65, Industrial-process measurement and control, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61131-5 on 2001-01-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2001-10-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2004-01-01

This part is to be read in conjunction with the other parts of EN 61131.

Annexes designated "normative" are part of the body of the standard.
In this standard, annexes A, B and ZA are normative.
Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61131-5:2000 was approved by CENELEC as a European Standard without any modification.

CONTENTS

	Page
1 Scope	8
2 Normative references	8
3 Definitions	9
4 Symbols and abbreviations	11
5 Models	11
5.1 PC network communication model	11
5.2 PC functional model	12
5.3 PC hardware model	14
5.4 Software model	14
6 PC communication services	15
6.1 PC subsystems and their status	15
6.1.1 PC summary status	16
6.1.2 I/O subsystem	17
6.1.3 Processing unit	18
6.1.4 Power supply subsystem	19
6.1.5 Memory subsystem	19
6.1.6 Communication subsystem	20
6.1.7 Implementer specific subsystems	20
6.1.8 Presentation of status information	21
6.2 Application specific functions	23
6.2.1 Device verification	24
6.2.2 Data acquisition	24
6.2.3 Control	25
6.2.4 Synchronization between user applications	25
6.2.5 Alarm reporting	26
6.2.6 Application program execution and I/O control	26
6.2.7 Application program transfer	28
6.2.8 Connection management	29
7 PC communication function blocks	29
7.1 Overview of the communication function blocks	29
7.1.1 Device verification	29
7.1.2 Data acquisition	30
7.1.3 Control	30
7.1.4 Alarm reporting	30
7.1.5 Connection management	30
7.2 Semantic of communication FB parameters	30
7.3 Device verification	35
7.4 Polled data acquisition	41
7.5 Programmed data acquisition	44
7.5.1 USEND/URCV function blocks	44
7.5.2 BSEND / BRCV Function Blocks	49
7.6 Parametric control	55
7.7 Interlocked control	58
7.8 Programmed alarm report	65

7.9	Connection management	73
7.10	Example for the use of communication function blocks	77
7.10.1	Establishing a communication channel	77
7.10.2	Transferring data	78
7.10.3	Using a timer to supervise communication	79
8	Compliance and implementer specific features and parameters	80
8.1	Compliance.....	80
8.2	Implementation specific features and parameters	81
Annex A	(normative) Mapping to ISO/IEC 9506-5.....	82
A.1	General	82
A.2	Application specific functions	83
A.2.1	Device verification.....	83
A.2.2	Data acquisition	83
A.2.3	Parametric control.....	83
A.2.4	Interlocked control.....	83
A.2.5	Synchronization between user applications	83
A.2.6	Alarm reporting	83
A.2.7	Application program execution and I/O control	83
A.2.8	Application program transfer	84
A.2.9	Connection management.....	84
A.3	PC object mapping.....	84
A.3.1	VMD	84
A.3.2	Named Variables.....	84
A.3.3	Unnamed Variables.....	86
A.3.4	Program Invocations	86
A.3.5	Domains	86
A.4	Communication function block mapping to MMS objects and services.....	87
A.4.1	Using communication channels	87
A.4.2	Rules for data type compatibility	87
A.4.3	Device verification.....	88
A.4.4	Polled data acquisition	89
A.4.5	Programmed data acquisition	90
A.4.6	Parametric control.....	93
A.4.7	Interlocked control.....	95
A.4.8	Programmed alarm report	97
A.4.9	Connection management.....	101
A.4.10	101	
A.5	Implementation specific features and parameters	103
Annex B	(normative) PC behavior using ISO/IEC 9506-2.....	104
B.1	PC communications mapping to MMS	104
B.2	Implementation specific features and parameters	105
Annex ZA	(normative) Normative references to international publications with their corresponding European publications	106
Figure 1	– Scope of this part of IEC 61131.....	8
Figure 2	– PC communication model.....	12
Figure 3	– Programmable controller functional model.....	13
Figure 4	– Programmable controller hardware model	14

Figure 5 – PC software model.....	15
Figure 6 – Programmable controller power supply.....	19
Figure 7 – Type description of status information	21
Figure 8 – Interlocked control timeline	25
Figure 9 – Function REMOTE_VAR	32
Figure 10 – Principle of status signalling	33
Figure 11 – Timing diagram of the ERROR and STATUS outputs	33
Figure 12 – STATUS function block	36
Figure 13 – USTATUS function block.....	37
Figure 14 – Timing diagram of the STATUS function block.....	37
Figure 15 – State diagram of STATUS function block.....	38
Figure 16 – State diagram of USTATUS function block	39
Figure 17 – READ function block	42
Figure 18 – Timing diagram of READ function block.....	43
Figure 19 – State diagram of READ function block.....	43
Figure 20 – Programmed data acquisition data flow	44
Figure 21 – USEND function block.....	45
Figure 22 – URCV function block	46
Figure 23 – Timing diagram of USEND and URCV function blocks	46
Figure 24 – State diagram of USEND function block	47
Figure 25 – State diagram of URCV function block.....	48
Figure 26 – BSEND function block	50
Figure 27 – BRCV function block	51
Figure 28 – Timing diagram of BSEND and BRCV function blocks.....	52
Figure 29 – State diagram of BSEND function block.....	53
Figure 30 – State diagram of BRCV function block.....	54
Figure 31 – WRITE function block	56
Figure 32 – Timing diagram of WRITE function block.....	57
Figure 33 – State diagram of WRITE function block	57
Figure 34 – SEND function block	59
Figure 35 – RCV function block	60
Figure 36 – Timing diagram of SEND and RCV function blocks	61
Figure 37 – State diagram of SEND function block	62
Figure 38 – State diagram of RCV function block	64
Figure 39 – NOTIFY function block	66
Figure 40 – ALARM function block	67
Figure 41 – Timing diagram of ALARM function block	68
Figure 42 – State diagram of NOTIFY function block.....	69
Figure 43 – State diagram of ALARM function block.....	71
Figure 44 – CONNECT function block.....	73
Figure 45 – Timing diagram of CONNECT function block	74
Figure 46 – State diagram of CONNECT function block.....	75
Figure 47 – Example in function block diagram language	80

Table 1 – Status presenting entities	16
Table 2 – PC summary status	17
Table 3 – Status of I/O subsystem	18
Table 4 – Status of processing unit	18
Table 5 – Status of power supply	19
Table 6 – Status of memory	20
Table 7 – Status of communication subsystem	20
Table 8 – Status of implementer specific subsystem	21
Table 9 – Presentation of status information	21
Table 10 – Device verification features	24
Table 11 – Data acquisition features	24
Table 12 – Control features	25
Table 13 – Alarm reporting features	26
Table 14 – Startable and stoppable units	26
Table 15 – Meaning of I/O State	27
Table 16 – I/O state	27
Table 17 – Execution and I/O control features	27
Table 18 – Loadable units	28
Table 19 – Application program transfer features	28
Table 20 – Connection management features	29
Table 21 – Overview of the communication function blocks	29
Table 22 – Semantic of communication FB parameters	31
Table 23 – Values of the SCOPE parameter	32
Table 24 – Value and interpretation of the STATUS output	34
Table 25 – Transitions of the STATUS state diagram	38
Table 26 – Action table for STATUS state diagram	39
Table 27 – Transitions of USTATUS state diagrams	40
Table 28 – Action table of USTATUS state diagram	40
Table 29 – Transitions of the READ state diagram	43
Table 30 – Action table for READ state diagram	44
Table 31 – Transitions of the USEND state diagram	47
Table 32 – Action table for USEND state diagram	48
Table 33 – Transitions of URCV state diagrams	49
Table 34 – Action table of URCV state diagram	49
Table 35 – Transitions of the BSEND state diagram	53
Table 36 – Action table for BSEND state diagram	54
Table 37 – Transitions of BRCV state diagrams	55
Table 38 – Action table of BRCV state diagram	55
Table 39 – Transitions of the WRITE state diagram	58
Table 40 – Action table for WRITE state diagram	58
Table 41 – Transitions of the SEND state diagram	62
Table 42 – Action table for SEND state diagram	63
Table 43 – Transitions of RCV state diagrams	64

Table 44 – Action table of RCV state diagram	65
Table 45 – Transitions of the NOTIFY state diagram	69
Table 46 – Action table for NOTIFY state diagram	70
Table 47 – Transitions of the ALARM state diagram	72
Table 48 – Action table for ALARM state diagram	72
Table 49 – Transitions of the CONNECT state diagram	76
Table 50 – Action table for CONNECT state diagram	77
Table 51 – Table titles and relevant tables for compliance	80
Table 52 – Implementation specific features and parameters	81
Table A.1 – Type description mapping	85
Table A.2 – Mapping of the SCOPE and SC_ID parameter	85
Table A.3 – Size prefix of direct representation	86
Table A.4 – Transition mapping of the STATUS state diagram	88
Table A.5 – Action mapping for STATUS state diagram	88
Table A.6 – Transition mapping of USTATUS state diagram	88
Table A.7 – Action mapping of USTATUS state diagram	88
Table A.8 – Transition mapping of the READ state diagram	89
Table A.9 – Action mapping for READ state diagram	89
Table A.10 – Transition mapping of the USEND state diagram	90
Table A.11 – Action mapping for USEND state diagram	90
Table A.12 – Transition mapping of URCV state diagram	91
Table A.13 – Action mapping for URCV state diagram	91
Table A.14 – Transition mapping of the BSEND state diagram	92
Table A.15 – Action mapping for BSEND state diagram	92
Table A.16 – Transition mapping of BRCV state diagram	93
Table A.17 – Action mapping for BRCV state diagram	93
Table A.18 – Transition mapping of the WRITE state diagram	95
Table A.19 – Action mapping for WRITE state diagram	95
Table A.20 – Transition mapping of the SEND state diagram	95
Table A.21 – Action mapping for SEND state diagram	96
Table A.22 – Transition mapping of RCV state diagram	96
Table A.23 – Action mapping of RCV state diagram	97
Table A.24 – Transition mapping of the NOTIFY state diagram	99
Table A.25 – Action mapping for NOTIFY state diagram	99
Table A.26 – Transition mapping of the ALARM state diagram	100
Table A.27 – Action mapping for ALARM state diagram	101
Table A.28 – Transitions of the CONNECT state diagram	102
Table A.29 – Action mapping for CONNECT state diagram	102
Table A.30 – Implementation specific features and parameters	103
Table B.1 – CreateProgramInvocation service defaults	104
Table B.2 – Program Invocation service defaults for I/O State parameter	104
Table B.3 – Implementation specific features and parameters	105

PROGRAMMABLE CONTROLLERS –

Part 5: Communications

1 Scope

This part of IEC 61131 specifies communication aspects of a programmable controller. It specifies from the viewpoint of a PC how any device can communicate with a PC as a server and how a PC can communicate with any device. In particular, it specifies the behavior of the PC as it provides services on behalf of other devices and the services the PC application program can request from other devices. It is not intended to specify how any device can communicate with any device using a PC as a router or gateway. The behavior of the PC as a communication client and server is specified independent of the particular communication subsystem, but the communication functionality may be dependent on the capabilities of the communication subsystem used.

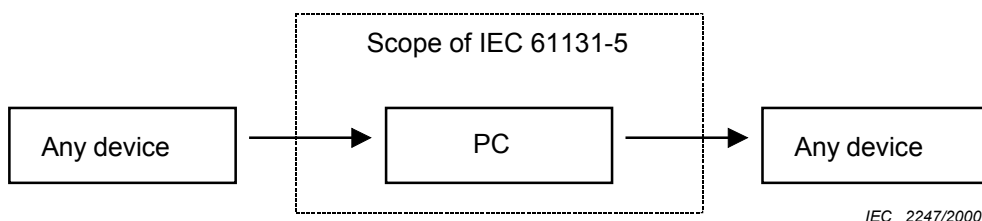


Figure 1 – Scope of this part of IEC 61131

The scope of this part is a subset of the "communication model" shown in figure 2 of IEC 61131-3; namely figures 2c and 2d are included in the scope of this part. Additionally, the means defined in this part of IEC 61131 may be used for communications within a program or between programs.

The mapping of the PC behavior to some particular communications subsystems is provided in the annexes.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61131. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61131 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

IEC 60050-351:1998, *International Electrotechnical Vocabulary – Part 351: Automatic control*

IEC 61131-1:1992, *Programmable controllers – Part 1: General Information*

IEC 61131-2:1992, *Programmable controllers – Part 2: Equipment requirements and tests*

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

ISO/IEC 2382-1:1993, *Information technology – Vocabulary – Part 1: Fundamental terms*

ISO/IEC 9506-1:1990, *Industrial automation systems – Manufacturing Message Specification – Part 1: Service definition*

ISO/IEC 9506-2:1990, *Industrial automation systems – Manufacturing Message Specification – Part 2: Protocol specification*

3 Definitions

For the purpose of this part of IEC 61131, the following definitions apply.

This part of IEC 61131 is based on the concepts of parts 1 to 3 of IEC 61131 and makes use of the following terms defined in other international standards.

Definitions from other publications

IEC 60050-351

- control
- monitoring

IEC 61131-1

- application program (2.1)
- application program archiving (4.6.4)
- cold restart (2.56)
- input (2.25)
- main processing unit (2.32)
- modifying the application program (4.6.2.6)
- output (2.40)
- programmable controller (2.50)
- programmable controller system (2.51)
- testing the application program (4.6.2.5)
- warm restart (2.56)

IEC 61131-3

- access path (1.3.2)
- direct representation (1.3.23)
- invocation (1.3.43)
- program (verb, 1.3.60)
- sub-element (2.3.3.1)

ISO/IEC 2382-1

- data

ISO/IEC 9506-1

client
download
event (clause 15)
server
uninterruptible variable access (12.1.1.1)
upload
variable

Definitions of this part

3.1

alarm

event which signals a specific condition

3.2

data acquisition

collection of data for the purpose of process monitoring and report generation

3.3

direct operator interface

when the client can communicate to the operator interface via the communication system with no application program interaction

3.4

device verification

allows other devices to determine if the PC is able to perform its intended function in the control system

3.5

health

the health of a PC or its subsystems is specified by returning one, and only one, of the three possible values. They are, in order of decreasing health: GOOD, WARNING and BAD

3.6

interlocked control

control through the synchronization of data exchanges between two parties. At various points in time, one party is waiting for the other party to deliver some expected data

3.7

local

internal to the PC; opposite of remote

3.8

parametric control

control by the client writing to control variables residing in the PC

3.9

processing unit

part of the main processing unit. It is the portion of a PC system which is responsible for the storage of the application program and data and the execution of the application program. A PC system has one or more processing units

3.10

program verification

testing of a PC application program to verify that it performs the function(s) it was designed to do in the process environment

3.11

recipe

description of procedures, or data for those procedures, or both, for making a product which uses the process or machinery that the controller is attached to, which is different from a previous product

3.12

remote

external to the PC; opposite of local

3.13

state

the state of the PC system is indicated by a list of attributes, each of which may be TRUE or FALSE. Zero, one, or more of these attributes may be TRUE at the same time

3.14

unsolicited

performed without an explicit request

4 Symbols and abbreviations

These are some abbreviations frequently used in this part of IEC 61131. These terms are defined or referenced in clause 3 of this part of IEC 61131.

CFB	Communication function block
FB	Function block
I/O	Input and output
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MMS	Manufacturing Message Specification, ISO/IEC 9506-1 and ISO/IEC 9506-2
OSI	Open Systems Interconnection
PADT	Programming and debugging tool
PC	Programmable controller
PU	Processing unit

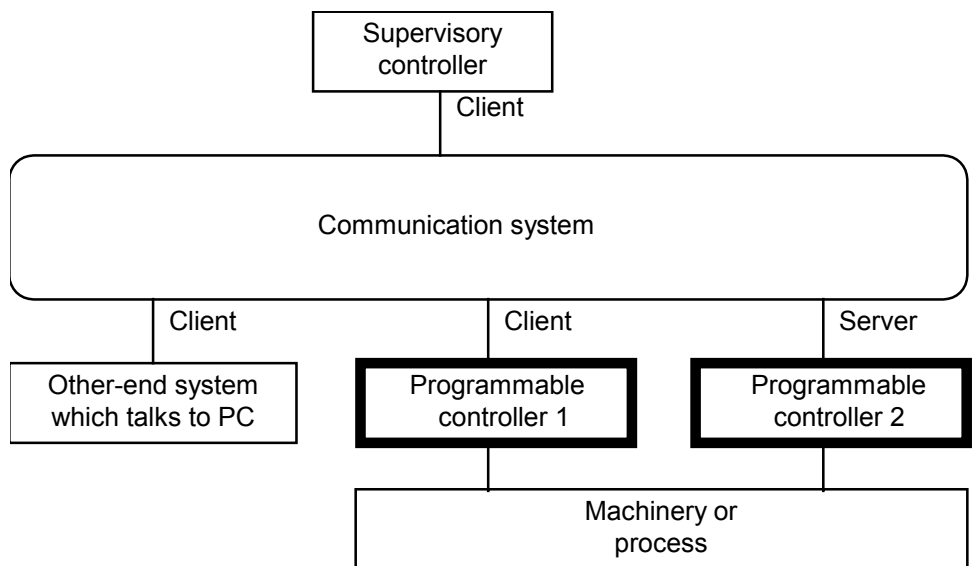
5 Models

This clause specifies the models which are used in the remainder of this part of IEC 61131.

5.1 PC network communication model

A programmable controller supplies some specific application functions to the rest of the control system. It may also request functions from other programmable controllers. The communication functions defined in this part of IEC 61131 are based on a communication subsystem that can report communication errors to the signal processing function of the PC (see 5.2).

The following diagram illustrates the devices in a communication network, showing three possible devices that request PC functions (clients) from PC 2. The two highlighted PCs are in the scope of this part of IEC 61131.



IEC 2248/2000

NOTE From the communication viewpoint the 'supervisory controller' and the 'other-end system which talks to PC' mentioned in this figure exhibit the same behavior to a PC communication server, i.e., they submit requests to the PC2.

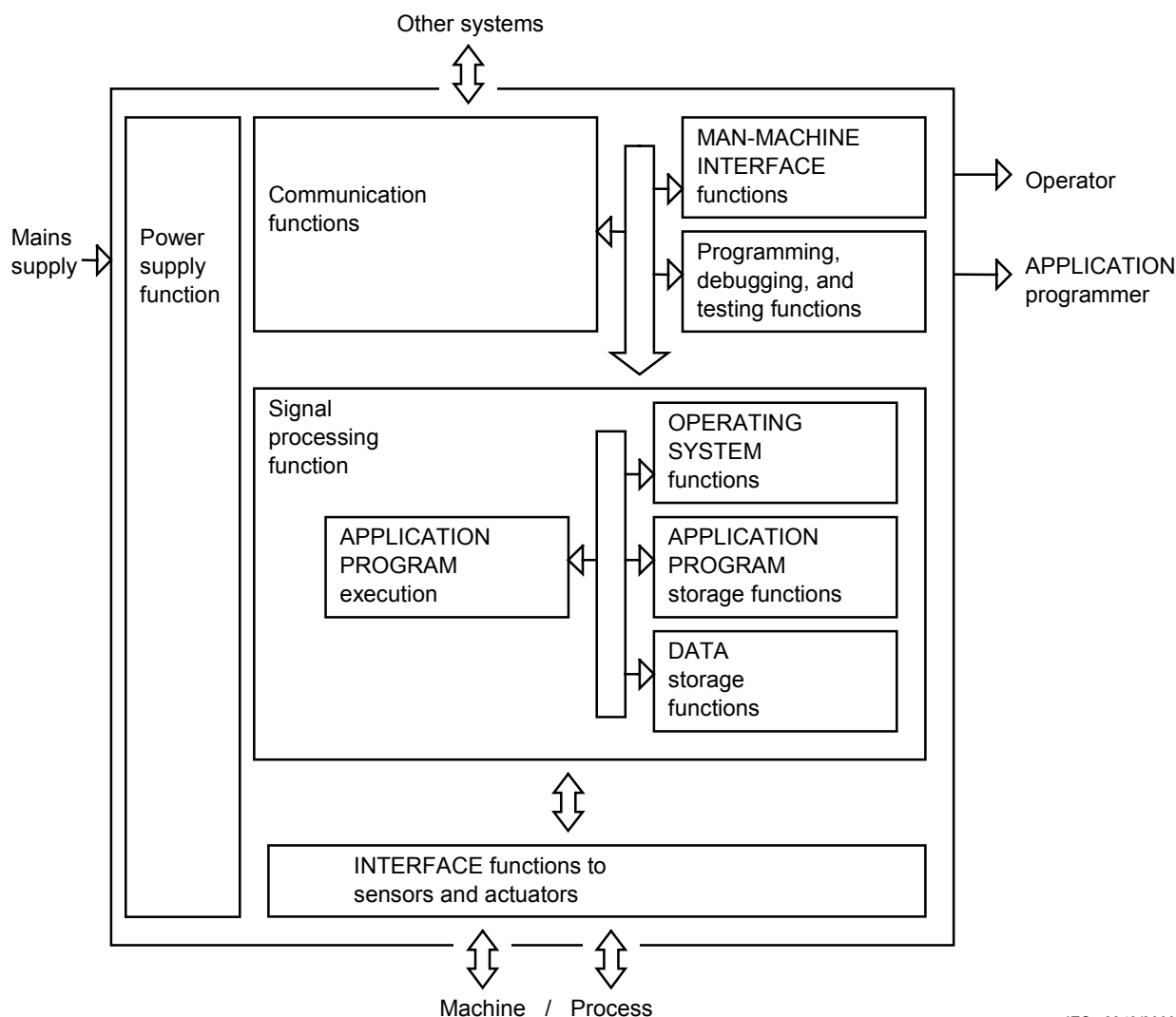
Figure 2 – PC communication model

A PC may use its client function to communicate with any device if it behaves like a PC.

5.2 PC functional model

A PC consists of several functions (see figure 3). For a PC within the scope of this part of IEC 61131, at least one communication function is present.

The following diagram is taken from IEC 61131-1, figure 1. It is designed to illustrate some of the subsystems of a typical PC.



IEC 2249/2000

Figure 3 – Programmable controller functional model

There is a function that is part of the PC system, but usually external to the PC itself, known as the programming and debugging tool (PADT). The PADT is modelled as interacting with the PC via the communications function.

The Interface Function to Sensors and Actuators can have I/O which are local or remote to the Main Processing Unit (see 5.3 for the hardware model). The Interface Function to Sensors and Actuators has two attributes for each Application Program which defines how the PC is monitoring and controlling the machine/process. The input attribute has the following states:

- inputs provided to the Application Program are being supplied by the sensors,
- inputs provided to the Application Program are being held in the current state.

The output attribute has the following states:

- the actuators are being controlled by the Application Program,
- the actuators are being held in the current state.

5.3 PC hardware model

The following figure shows the PC hardware model. It shows the modules that make up a PC. A PC subsystem consists of one or more modules. The following figure corresponds to figure B.1 of IEC 61131-1 and figure 1 of IEC 61131-2.

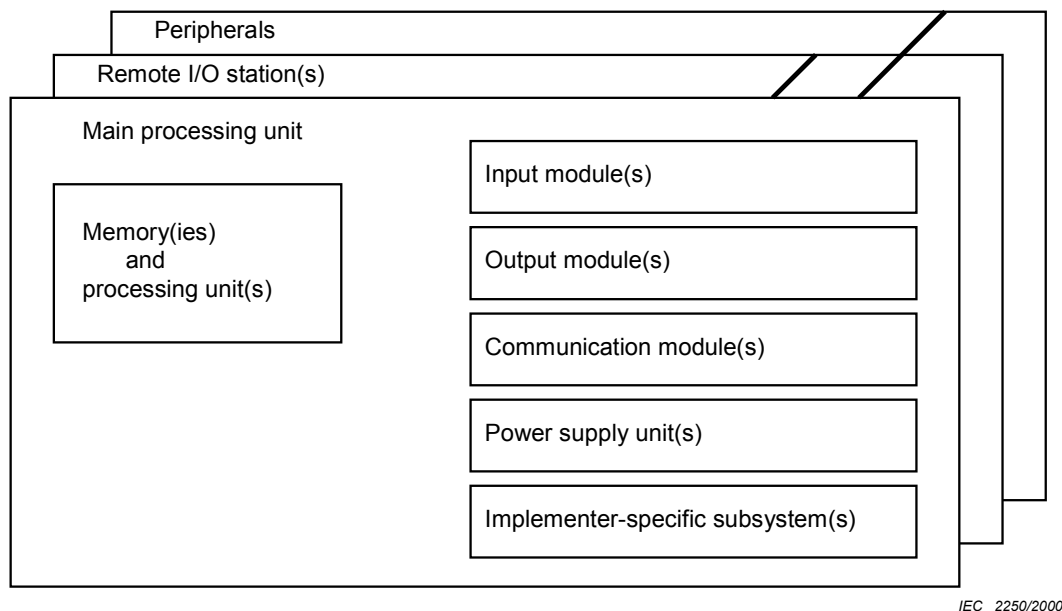


Figure 4 – Programmable controller hardware model

5.4 Software model

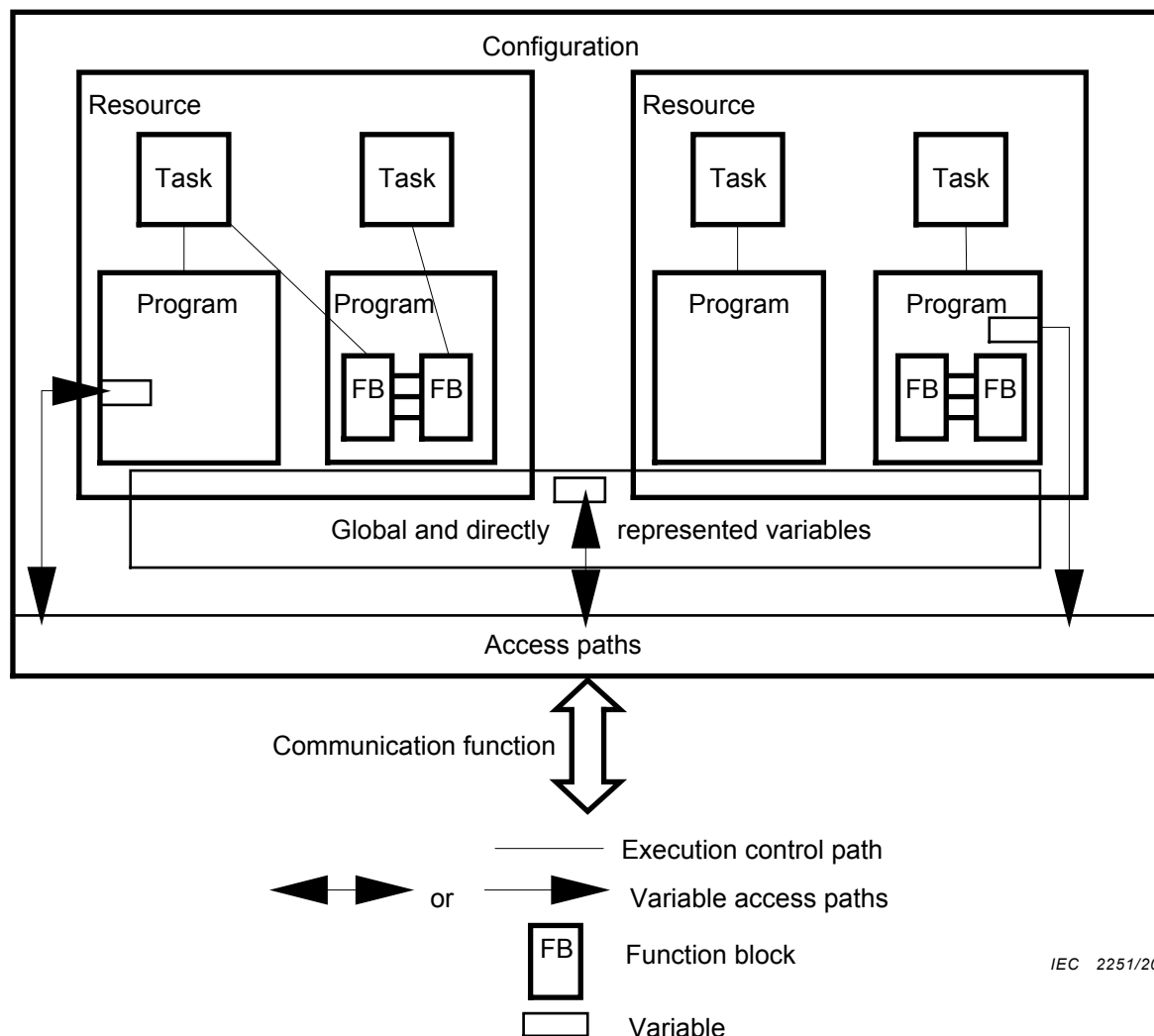
Figure 5 shows the PC software model defined in IEC 61131-3, figure 1. It illustrates the basic high-level language elements of the PC programming languages and their interrelationships. These consist of elements which are programmed using the languages defined in IEC 61131-3, i.e. programs and function blocks; and configuration elements, namely, configurations, resources, tasks, global variables, and access paths, which support the installation of programmable controller programs into programmable controller systems.

A configuration is the language element which corresponds to a programmable controller system as defined in IEC 61131-1. A resource corresponds to a "signal processing function" and its "man-machine interface" and "sensor and actuator interface" functions (if any) as defined in IEC 61131-1. A configuration contains one or more resources, each of which contains one or more programs executed under the control of zero or more tasks. A program may contain zero or more function blocks or other language elements as defined in IEC 61131-3.

Configurations and resources can be started and stopped via the "operator interface", "programming, testing, and monitoring", or "operating system" functions defined in IEC 61131-1. The mechanisms for the starting and stopping of configurations and resources via communication services are defined in this part of IEC 61131.

Programs, resources, global variables, access paths (and their corresponding access privileges), and configurations can be loaded or deleted by the "communication function" defined in IEC 61131-1. The loading or deletion of a configuration or resource shall be equivalent to the loading or deletion of all the elements it contains.

Access paths and their corresponding access privileges allow to access variables of a PC via communication services.



NOTE 1 This figure is illustrative only. The graphical representation is not normative.

NOTE 2 In a configuration with a single resource, the resource need not be explicitly represented.

Figure 5 – PC software model

6 PC communication services

This clause describes the concept of status information of a PC and provides a specification of the services the PC provides to the control system via the communication subsystem. (The next clause specifies how the PC application program can use the communication subsystem to interact with other devices.)

6.1 PC subsystems and their status

A PC can provide status, which includes state information and fault indications.

Status can be reported on some of the subsystems identified in the following figure. In addition, there is a summary status that provides general information about the PC.

Table 1 – Status presenting entities

No.	Status presenting entities
1	PC (as a whole)
2	I/O subsystem (includes Input and Output modules and other intelligent I/O devices)
3	Processing unit
4	Power supply subsystem
5	Memory subsystem
6	Communication subsystem
7	Implementer specific subsystems
NOTE The status is intended to provide information about the controller including its hardware and firmware subsystems, not considering configuration information. It is not intended to provide information about the controlled process nor the PC application program. The status data contains information concerning the state and the health of the PC and its subsystems.	

There are two concepts used in this part of IEC 61131 related to status: health and state. The "health" of a PC or its subsystems is specified by returning one and only one of the three possible values. The semantics associated with each value is specified below. They are, in order of decreasing health:

- a) GOOD – If TRUE, the PC (or the specified subsystem) has not detected any problems which would prohibit it from performing the intended function;
- b) WARNING – If TRUE, the PC (or the specified subsystem) has not detected any problems which would prohibit it from performing the intended function, but it has detected at least one problem which could place some limits on its abilities. The limit may be time, performance, etc. (see the following statements for further definition of these limits);
- c) BAD – If TRUE, the PC (or the specified subsystem) has detected at least one problem which could prohibit it from performing the intended function.

The "state" of the PC system is indicated by a list of attributes, each of which may be TRUE or FALSE. Zero, one, or more of these attributes may be TRUE at the same time. The semantics associated with each attribute is specified in the remainder of this clause.

Each of the status information can also have implementer specified attributes. Some examples of implementer specified attributes are:

- a) additional error diagnostics (e.g. EEPROM write cycles exceeded);
- b) additional operational states (e.g. auto-calibrate enabled);
- c) local key status (e.g. auto-restart required).

Implementations are not required to provide subsystem status. All instances of similar types of subsystems present in a system are reported separately. The name of the subsystem can be provided to allow differentiating subsystems of the same type.

6.1.1 PC summary status

The PC provides the following summary status information.

Table 2 – PC summary status

No.	Item	Description	
1	Health	GOOD	All subsystems in the PC indicate a GOOD health condition
2		WARNING	At least one subsystem indicates a WARNING health condition and no subsystem indicates a BAD health condition
3		BAD	At least one subsystem indicates a BAD health condition
4	Running	If TRUE, this attribute indicates if at least one part of the user application has been loaded and is under control of the PC	
5	Local control	If TRUE, this attribute indicates if local override control is active. If active, the ability to control a PC and its subsystems from the network may be limited. For example, this could be closely tied to the use of a local key switch	
6	No outputs disabled	If TRUE, this attribute indicates that the PC can change the physical state of all outputs as a result of application program execution or other means. If not TRUE, the physical state of some of the outputs are not affected (logical state may be affected). This is typically used in the testing and modifying of application programs in the PC	
7	No inputs disabled	If TRUE, this attribute indicates that the PC can access the physical state of all inputs as a result of application program execution or other means. If not TRUE, the physical state of some inputs cannot be accessed. This is typically used in the testing and modifying of application programs where the inputs can be simulated	
8	Forced	If TRUE, this attribute indicates that at least one I/O point associated with the PC has been forced. When an Input is forced, the application program will receive the value specified by the PADT instead of the actual value from the machine or process. When an output is forced, the machine or process will receive the value specified by the PADT instead of the value generated by execution of the application program. When a variable is forced, the application program will use the value specified by the PADT instead of that generated by the normal program execution	
9	User application present	If TRUE, this attribute indicates that the Processing Unit has at least one user application present	
10	I/O subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by an I/O subsystem	
11	Processing unit subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by a processing unit subsystem	
12	Power supply subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by a power supply subsystem	
13	Memory subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by a memory subsystem	
14	Communication subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by a communication subsystem	
15	Implementer specified subsystem	If TRUE, this attribute indicates "WARNING" or "BAD" which is caused by an implementer specified subsystem	

6.1.2 I/O subsystem

The PC provides the following status information of its I/O subsystem.

Table 3 – Status of I/O subsystem

No.	Item	Description	
1	Health	GOOD	indicates that there have been no errors detected in this I/O subsystem
2		WARNING	indicates that a minor fault has been detected in the I/O subsystem. An example of a minor fault is the occurrence of recoverable errors in the communication with a remote I/O station
3		BAD	indicates that a major fault has been detected in the I/O subsystem. An example of a major fault is losing communication with a remote I/O station
4	No outputs disabled	If TRUE, this attribute indicates that the PC can change the physical state of all outputs associated with the specified I/O subsystem as a result of application program execution or other means. If not TRUE, the physical state of some of the outputs is not affected (logical state may be affected). This is typically used in the testing and modifying of application programs in the PC	
5	No inputs disabled	If TRUE, this attribute indicates that the PC can access the physical state of all inputs associated with the specified I/O subsystem as a result of application program execution or other means. If not TRUE, the physical state some inputs cannot be accessed. This is typically used in the testing and modifying of application programs where the inputs can be simulated	
6	I/O forced	If TRUE, this attribute indicates that at least one I/O point associated with this subsystem has been forced. When an Input is forced, the application program will receive the value specified by the PADT instead of the actual value from the machine or process. When an output is forced, the machine or process will receive the value specified by the PADT instead of the value generated by execution of the application program	
NOTE The definition of "major fault" and "minor fault" shall be provided by the implementer.			

6.1.3 Processing unit

The PC provides the following status information of its processing unit.

Table 4 – Status of processing unit

No.	Item	Description
1 2 3	Health	This attribute identifies the health of the processing unit. The implementer shall specify the conditions when GOOD, WARNING or BAD are valid
4	Running	If TRUE, this attribute indicates if at least one part of the user application has been loaded and is under control of the processing unit
5	Local control	If TRUE, this attribute indicates if local override control is active. If active, the ability to control the processing unit from the network may be limited. For example, this could be closely tied to the use of a local key switch
6	No outputs disabled	If TRUE, this attribute indicates that the processing unit can change the physical state of all outputs controlled by this processing unit as a result of application program execution or other means. If not TRUE, the physical state of some of the outputs are not affected (logical state may be affected). This is typically used in the testing and modifying of application programs in the PU
7	No inputs disabled	If TRUE, this attribute indicates that the processing unit can access the physical state of all inputs accessible from this processing unit as a result of application program execution or other means. If not TRUE, the physical state of some inputs cannot be accessed. This is typically used in the testing and modifying of application programs where the inputs can be simulated
8	User application present	If TRUE, this attribute indicates that the Processing Unit has at least one User Application present
9	Forced	If TRUE, this attribute indicates that at least one variable associated with this Processing Unit has been forced. When a variable is forced, the application program will use the value specified by the PADT instead of that generated by the normal program execution.

6.1.4 Power supply subsystem

The PC can provide status information about any of the power supply subsystems; see figure 6 for the assumed configuration of a PC power supply. The requirements on power supplies of PC systems and their behavior is described in IEC 61131-1 and IEC 61131-2.

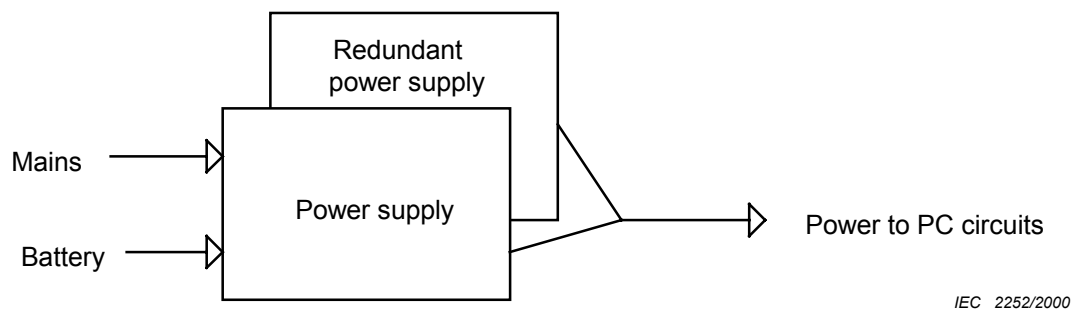


Figure 6 – Programmable controller power supply

Table 5 – Status of power supply

No.	Item	Description	
1	Health	GOOD	indicates that there have been no problems detected in the power supply to prevent it from remaining operable for an indefinite time
2		WARNING	indicates that a problem has been detected in the power supply which may cause to become inoperable in a limited time
3		BAD	indicates that the power supply is not operable
4	In use	If TRUE, this attribute indicates that the power supply subsystem is in use, i.e. it supplies power to the PC	
5	Mains operating	If TRUE, this attribute indicates that the mains are supplying power within the range specified for the power supply	
6	Mains low	If TRUE, this attribute indicates that the mains are not supplying power within the range specified for the power supply	
7	Battery operating	If TRUE, this attribute indicates that the battery is supplying power within the range specified for the power supply	
8	Battery low	If TRUE, this attribute indicates that the battery is not able to supply power within the range specified for the power supply	
9	Protection tripped	If TRUE, this attribute indicates that a protection device within the power supply has removed a portion of the power to the PC	

6.1.5 Memory subsystem

The PC provides the following status information of its memory subsystem.

Table 6 – Status of memory

No.	Item	Description	
1	Health	GOOD	No errors have been found in the memory associated with this subsystem
2		WARNING	At least one correctable error has been detected and no uncorrectable errors have been detected
3		BAD	At least one uncorrectable error has been detected
4	Protected ¹⁾	If TRUE, this attribute indicates that the memory in this memory subsystem has been protected in that it cannot be modified. This generally indicates that the application program located in this memory subsystem cannot be altered. ¹⁾ This attribute models a logical state not physical characteristics of the subsystem. If some portions of the memory are protected and some are not, these shall be reported as multiple subsystems.	

6.1.6 Communication subsystem

The PC provides the following status information of its communication subsystem.

Table 7 – Status of communication subsystem

No.	Item	Description	
1	Health	GOOD	indicates that either no errors or an acceptable number of recoverable errors has occurred
2		WARNING	indicates that more than an acceptable number of recoverable errors has occurred
3		BAD	indicates that the communication subsystem is not able to communicate with all devices as intended
4	In use	If TRUE, this attribute indicates that the communication subsystem is currently operating. For example in the case of an MMS communication interface this means that at least one application association is established. Otherwise, the implementer shall define the semantic of this attribute	
5	Local error	If TRUE, this attribute indicates that there are some errors, internal to the communication subsystem, that inhibit operation	
6	Remote error	If TRUE, this attribute indicates that there are some errors, at devices being communicated with, that inhibit operation	
NOTE 1 The communication subsystem reporting its state may not be able to report its own bad state in the way defined in this clause. But, within a PC system, several independent communication subsystems may operate, and all of them may provide status information. NOTE 2 It is intended that the implementer specific information will provide additional information about each particular interface. ISO network interfaces also provide additional information via network management functions.			

6.1.7 Implementer specific subsystems

Other subsystems of a PC system shall be modelled as implementer specific subsystems. Some examples of these subsystems are:

- a) PID controller;
- b) motion controller;
- c) other auxiliary processors.

Table 8 – Status of implementer specific subsystem

No.	Item	Description	
1	Health	GOOD	indicates that there have been no errors detected in this subsystem
2		WARNING	indicates that a minor fault has been detected in this subsystem
3		BAD	indicates that a major fault has been detected in this subsystem
NOTE The definition of "major fault" and "minor fault" shall be provided by the implementer.			

6.1.8 Presentation of status information

The status information shall be presented using variables with a pre-defined access path in the configuration declaration of the PC application program or shall be presented as a variable with direct representation to a remote communication partner.

Table 9 – Presentation of status information

No.	Presentation of status information
1	PC summary status as variable with pre-defined access path P_PCSTATE
2	PC summary status as variable with direct representation %S
3	PC summary status and status of all subsystems as variable with pre-defined access path P_PCSTATUS
4	Status information of each subsystem as a set of variables with direct representation %SC<n>
5	Type of each subsystem as a set of variables with direct representation %SU<n>
6	Name of each subsystem as a set of variables with direct representation %SN<n>
7	State of each subsystem as a set of variables with direct representation %SS<n>
8	Implementer specific status of each subsystem as a set of variables with direct representation %SI<n>

If the PC summary status shall be presented in a variable, it shall have the access path P_PCSTATE which shall be pre-defined in the configuration declaration. The variable shall be of type WORD and shall contain the PC summary status beginning with item number 1 at the least significant bit upwards. IEC 2253/2000

If the PC summary status shall be presented as a variable with direct representation, the direct representation shall be %S and shall be of type WORD. It shall contain the PC summary status beginning with item number 1 at the least significant bit upwards.

If the complete status information shall be presented as a variable, it shall have the access path P_PCSTATUS pre-defined in the configuration section. This variable shall have a structured type as follows:

```
ARRAY [0..p_NOS] OF
  STRUCT
    SUBSYSTEM : (SUMMARY, IO, PU, POWER, MEMORY, COMMUNICATION,
                  IMPLEMENTER);
    NAME      : STRING[<Max_Name_Len>];
    STATE     : ARRAY[0..15] OF BOOL;
    SPECIFIC  : ARRAY[0..p_BIT] OF BOOL;
  END_STRUCT;
```

Figure 7 – Type description of status information

The array element with the number 0 shall contain the PC summary status, each element with a higher number shall contain the status of one subsystem. The sub-element SUBSYSTEM shall contain the type of the PC or of a subsystem. The sub-element NAME shall contain the name of the subsystem.

shall contain the name of the PC or of a subsystem. The implementer shall specify the supported maximum length for name strings, i.e. the value of `Max_Name_Len`. The sub-element `STATE` shall contain the state information of the PC or of a subsystem as an array of `BOOL` in the same order as specified in tables 2 to 8. The implementer shall specify the number of elements of the array `P_PCSTATUS` i.e. the value of `p_NOS`, the supported types of subsystems, the semantic of the values in the sub-element `STATE` for the implementer specific subsystem, the size of the sub-element `SPECIFIC`, i.e. the value of `p_BIT`, and the semantic of the sub-element `SPECIFIC`.

The status information of each subsystem may be presented as a variable with direct representation `%SC<n>`, where `<n>` stands for a number between 0 (representing the PC summary status) and the number of subsystems `p_NOS`. The variable shall have the same internal representation as a variable with the type of the structure part of the type described in the figure above.

Additionally there may be a set of variables with direct representation %SU<n>, %SN<n>, %SS<n>, and %SI<n>. The <n> stands for a number between 0 (representing the PC summary status) and the number of subsystems p_NOS. The variables shall have the same internal representation as a variable with the type of one of the structure sub-elements of the type described in the figure above. In detail, the %SU<n> shall correspond to the sub-element SUBSYSTEM, %SN<n> to the sub-element NAME, %SS<n> to the sub-element STATE, and %SI<n> to the sub-element SPECIFIC.

6.2 Application specific functions

The remainder of this clause describes the functions which a PC provides to a control system, using the communication subsystem, as illustrated in figure 2.

PC communication function	PC as requester	PC as responder	Function block available
Device verification	yes	yes	yes
Data acquisition	yes	yes	yes
Control	yes	yes	yes
Synchronization between user applications	yes	yes	yes
Alarm reporting	yes	no	yes
Program execution and I/O control	no	yes	no
Application program transfer	no	yes	no
Connection management	yes	yes	yes

Each of these is treated separately in the remainder of this clause. Not all functions are available in all PCs. See clause 7 for the function block definitions.

There are some applications which combine the application categories defined below, for example, supervisory control and data acquisition.

The following elements, while usually provided by PCs, are outside the scope of this part of IEC 61131:

- a) operator interface;
- b) programming, testing, and modifying the application program, and program verification.

PCs have the ability to use operator interface devices. These devices are used by an operator to monitor or modify the controlled process or both. They may also be used by a client system to communicate with the operator.

Direct operator interface is when the client can communicate to the operator interface via the communication system with no application program interaction.

Programming is the process of creating a PC application program on an instruction by instruction or a function block by function block basis. Testing and modifying is the process of finding and removing errors ("bugs") in an existing application program by making changes to it. Program verification is the testing of a PC application program to verify that it performs the function(s) it was designed to do in the process environment.

6.2.1 Device verification

This function is provided to allow other devices to determine if the PC is able to perform its intended function in the automated system. A PC can provide status of itself and its subsystems. Status includes health and state information. A device may explicitly request status from the PC or the PC may initiate an unsolicited status report using services provided by the communication interface. See 6.1 for the definition of health and state information of a PC system and of its subsystems.

Table 10 – Device verification features

No.	Device verification
1	Provide status information
2	Initiate unsolicited status reports

6.2.2 Data acquisition

Data contained in a PC is presented as variables. This data may come from a variety of sources and may have a wide range of meanings. It can be obtained by the client through one of several methods.

- Polled – The client reads the value of one or more variables at a time or condition determined by the client. The access to the variables may be controlled by the PC. Only selected variables are accessible over the network.
- Programmed – The data is provided by the PC to the client at a time or condition determined by the PC application program.
- Configured – The communications interface to the PC can be configured by a client to initiate a data transfer to the client.

The kinds of variables in the PC which are visible to the communication system are:

- variables with direct representation;
- other variables which have access paths (see IEC 61131-3 for the definition of access paths).

If the directly represented variables are accessible for communication these variables shall use the direct representation as an identifier. The PC server (i.e. the PC which owns the variables) can interpret the identifier using an implementer defined algorithm.

NOTE Variables with direct representation can be used like "normal" variables while programming an application program. An additional symbolic name may be assigned to a directly represented variable using the AT construct in the variable declaration (see IEC 61131-3).

Typically there are thousands of these variables with direct representation even in a smaller PC. It is not reasonable to hold the name and the address of all these variables in an object dictionary of a PC.

The PC system may restrict access to variables with direct representation. The conditions (size, location, etc.) under which each data type supported by the PC can be uninterruptedly accessed shall be specified by the implementer.

Table 11 – Data acquisition features

No.	Data acquisition
1	Variables with direct representation are accessible
2	Access paths on configuration level
3	Access paths on program level
4	Means to restrict access to variables with direct representation
5	Conditions for uninterruptible access to variables

6.2.3 Control

A PC may support two methods of control: parametric and interlocked.

Parametric control is when the operation of the PC is directed by writing values to variables residing in the PC. This change in operation is determined by either the application program or other local mechanisms.

The access to the variables is controlled by the PC which holds the variables. Only those variables which have the READ_WRITE qualifier selected in the access path declaration are accessible over the network for parametric control.

Interlocked control is when the client requests the server to execute an application operation and to inform the client of the result of the operation. There are two aspects of this service, the synchronization of the client and server, and the exchange of data between them.

In interlocked control, this data exchange occurs at synchronization points in the application program. This service can be used to have the effect of a remote procedure call from one application program to another. The timeline shown in figure 8 illustrates this.

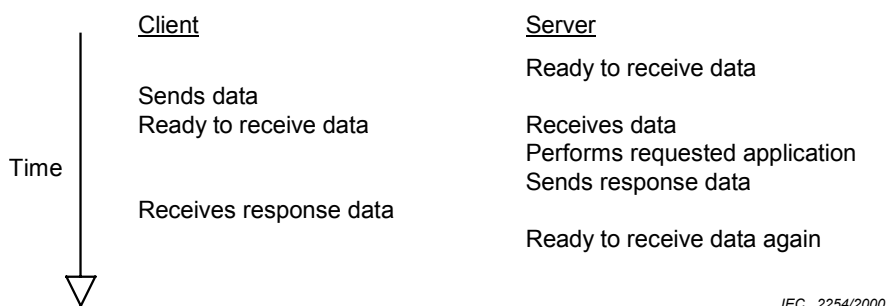


Figure 8 – Interlocked control timeline

The PC implements interlocked control using the SEND (client) and RCV (server) function blocks. Other devices may use other means to emulate the behavior of these function blocks to access this PC communication function.

Table 12 – Control features

No.	Control
1	Variables with direct representation are accessible
2	Access paths on configuration level
3	Access paths on program level
4	Means to restrict access to variables with direct representation
5	Conditions for uninterruptible access to variables
6	Interlocked control

6.2.4 Synchronization between user applications

User applications may need a synchronization service. For example, a user application may start the execution of another application after completion of an algorithm. The synchronization service is provided by the interlocked control mechanism (see 6.2.3).

6.2.5 Alarm reporting

The PC can have the ability to signal alarm messages to a client when a predetermined condition occurs. The client may indicate an acknowledgement of these alarms to the PC. This differs from normal data acquisition in that the state of an alarm point is remembered by the PC until it is acknowledged by the client. A summary of the unacknowledged alarms can be generated by the PC at the request of the client.

Table 13 – Alarm reporting features

No.	Alarm reporting
1	Signal messages
2	Receive acknowledgements
3	Generate summary of unacknowledged alarms

6.2.6 Application program execution and I/O control

Execution of a PC Application Program is managed by the Application Program Execution function (see 5.2). PC Application Programs can be started and stopped. PC Application Programs can be started either from an initial state or from the state they were in at the time they were stopped.

The PC application program in a PC system consists of one configuration and zero, one or more resources (see IEC 61131-3). Configurations and resources may be started and stopped. The resources are started and stopped when the configuration is started and stopped and they can be started or stopped independently of the configuration.

The Interface Function to Actuators (outputs) associated with a running Application Program can be directed to either use the values supplied by the Application Program or held in a known state. This Interface state is specified at the time the Application Program state is changed. The outputs can be directed to either be set to implementer specified states, hold the outputs in the current state, set all outputs to zero, or change some outputs to user specified states (on or off, with those not specified holding the last state) through an implementer specified mechanism (for example, tables, PC procedure, etc.).

The Interface Function to Sensors (inputs) associated with a running Application Program can be directed to either provide the actual data from the sensors or to continue to use previously supplied values. The input state is specified at the time the Application Program state is changed.

Table 14 – Startable and stoppable units

No.	Startable and stoppable unit
1	Configuration
2	Resource

The I/O (inputs and outputs) associated with a running resource shall either be controlled by the program or held in a known state, which is determined when the resource is started. Resources shall be able to be started either from an initial state (cold restart using START) or from the state they were in at the time they were stopped (warm restart using RESUME). The desired state of the outputs shall be able to be specified as part of the stopping process.

The state of the I/O can be set to the following values when a configuration or resource is started or stopped:

Table 15 – Meaning of I/O State

Value of I/O State	Meaning	Set by
Controlled	Actuators are being controlled by the application program or the part of the application program being started. Inputs are being supplied to the application program or the part of the application program being started by the interface function to sensors and actuators.	Starting
Hold outputs	Actuators are not being controlled by the application program or the part of the application program being started, they are held in the current state. Sensors are being supplied to the application program or the part of the application program being started by the interface function to sensors and actuators.	Starting
Hold current state	Actuators are not being controlled by the application program or the part of the application program being started or stopped, they are held in the current state. Sensors are not being supplied to the application program or the part of the application program being started, they are held in the current state.	Starting, stopping
Implementer state	Actuators are not being controlled by the application program or the part of the application program being stopped, they are held in a state, which was specified by the implementer. The application program is not running, therefore the state of the Sensor Interface is not specified.	Stopping
Zero outputs	Actuators are not being controlled by the application program or the part of the application program being stopped, they are held in the zero state. The application program is not running, therefore the state of the Sensor Interface is not specified.	Stopping
User specified	Actuators are not being controlled by the application program or the part of the application program being stopped, they are held in a state which was specified by the user. The application program is not running, therefore the state of the Sensor Interface is not specified.	Stopping

Table 16 – I/O state

No.	I/O state
1	Controlled
2	Hold outputs
3	Hold current state
4	Implementer specified
5	Zero outputs
6	User specified
NOTE The communication subsystem should not be depended upon as a replacement for hardwired emergency stop switches. Normal safety practices should be followed.	

Table 17 – Execution and I/O control features

No.	Execution and I/O control
1	Receive requests to start a startable and stoppable unit
2	Receive requests to stop a startable and stoppable unit
3	Receive requests to resume a startable and stoppable unit

6.2.7 Application program transfer

The application program is transferred using the application program storage and data storage function of a PC (see 5.2). Application program transfer allows the client to upload the complete contents of the programmable memory or portions thereof for archiving or verification or to download it for restoring the PC to a known state. This function also provides the ability to place the PC in a safe state before modifying the contents of its programmable memory, and restarting it in a safe manner when the application program transfer is completed.

The initiation of the program transfer is typically done by a device that is not a PC. The services include:

- a) upload for archive;
- b) upload for verification;
- c) download for restore to previously known good system; and
- d) download an off-line developed system.

The portions of the programmable memory which can be uploaded or downloaded are given in table 18.

NOTE A load unit contains the variable P_DDATE, their value is the date of the last modification of the load unit.

Table 18 – Loadable units

No.	Loadable units
1	Configuration
2	Resource
3	Programs
4	Global variables
5	Access paths on configuration level
6	Access paths on program level
7	Load units contain the variable P_DDATE

The implementer shall specify if other language elements, for example, function types or function block types are loadable and the conditions and restrictions for downloading and uploading these.

This part of IEC 61131 defines a means to perform uploads and downloads on the whole PC, a subsystem of the PC and a portion of a subsystem. The whole or the portion of the PC to be downloaded shall be explicitly stopped before the download. The whole or the portion of the system being downloaded shall not be available for other uses until the download is completed. The implementer shall specify what other clients can do with a PC when one client is downloading it.

Table 19 – Application program transfer features

No.	Application program transfer
1	Receive requests to download a loadable unit
2	Receive requests to upload a loadable unit

6.2.8 Connection management

The Connection Management provides the means to install, to maintain and close communications connections to a remote communication partner. If multiple requests are initiated to operate with a device, they may all work over the same connection. Not all communications subsystems require connections, for example, point-to-point communication.

Connections are controlled explicitly by the application program using the CONNECT function block or are provided by the communication subsystem if and when needed.

Table 20 – Connection management features

No.	Connection management
1	Install connections
2	Close connections
3	Using one connection for multiple requests

7 PC communication function blocks

7.1 Overview of the communication function blocks

The following PC communication functions and their corresponding function blocks are described below.

Table 21 – Overview of the communication function blocks

No.	Subclause	Name of communication function block or function
1	7.2 Semantic of communication FB parameters (addressing of remote variables)	REMOTE_VAR
2 3	7.3 Device verification	STATUS, USTATUS
4	7.4 Polled data acquisition	READ,
5 6 7 8	7.5 Programmed data acquisition	USEND, URCV, BSEND, BRCV
9	7.6 Parametric control	WRITE,
10 11	7.7 Interlocked control	SEND, RCV
12 13	7.8 Programmed alarm report	NOTIFY, ALARM
14	7.9 Connection management	CONNECT

The numbers given in the above table shall be used to state compliance to these communication function blocks (CFB).

7.1.1 Device verification

The STATUS and USTATUS function blocks are provided so that the PC can collect status from other devices. These are provided to allow the PC to determine if the other devices are able to perform their intended function in the automated system.

7.1.2 Data acquisition

Data contained in other devices may be presented as variables. This data may come from a variety of sources and may have a wide range of meanings. It can be obtained by the PC through one of two methods using communication function blocks.

- a) Polled – The PC uses the READ function block to obtain the value of one or more variables at a time or condition determined by the PC application program. The access to the variables may be controlled by the device being read.
- b) Programmed – The data is provided to the PC at a time or condition determined by the other device. The PC uses the URCV function block to provide the data to the PC application program. The PC uses the USEND to provide unsolicited data to other devices.

The conditions (size, location, etc.) under which each data type supported by the other device can be uninterruptedly accessed is determined by the other device.

7.1.3 Control

Two methods of control shall be supported by PCs: parametric and interlocked.

Parametric control is when the operation of the other device is directed by writing values to variables residing in it. The access to the variables may be controlled by the PC which holds the variables. The PC uses the WRITE function block to perform this action from the PC application program.

Interlocked control is when the client requests the server to execute an application operation and to inform the client of the result of the operation. The PC uses the SEND and RCV function blocks to implement the client and server roles, respectively.

7.1.4 Alarm reporting

The PC can have the ability to signal alarm messages to a client when a predetermined condition occurs. The client may indicate an acknowledgement of these alarms to the PC. The ALARM and NOTIFY function blocks are used by the PC application program to generate acknowledged and unacknowledged alarms, respectively.

7.1.5 Connection management

The PC application program uses the CONNECT function block to manage connections.

7.2 Semantic of communication FB parameters

The communication function blocks use a common semantic of their function block inputs and outputs. The meaning of these inputs and outputs is described below. Some communication function blocks have special input or output parameters, they are described where the communication function blocks themselves are described.

Table 22 – Semantic of communication FB parameters

Parameter name	Data type of the parameter	Interpretation
EN_R	BOOL	Enabled to receive data
REQ/RESP	BOOL	Perform function on raising edge
ID	COMM_CHANNEL	Identification of the communication channel
R_ID	STRING	Identification of the remote FB inside the channel
SD_i	ANY	User data to send
VAR_i	STRING or data type of the output of the function REMOTE_VAR	Identification of a variable of the remote communication partner
DONE	BOOL	Requested function performed (good and valid)
NDR	BOOL	New user data received (good and valid)
ERROR	BOOL	New non-zero status received
STATUS	INT	Last detected status (error or good)
RD_i	ANY	Last received user data

The ID input references the communication channel used by the instance of the communication function block, i.e. it determines the remote communication partner. The ID input is of COMM_CHANNEL type which shall be implementer defined.

NOTE The value given at the ID input of a communication function block instance is intended to hold or reference the information which is necessary to manage the communication to the remote communication partner. This information may be dependent on the implementation and the communication subsystem used.

The R_ID input is used to identify the corresponding instance of the communication function block at the remote partner, if the PC communication function is provided by a corresponding pair of function block instances.

One instance of a communication function block shall use the same communication channel and communicate to the same corresponding remote function block instance throughout its whole lifetime.

The variables to be read or written are identified using the VAR_i inputs of the READ and WRITE function blocks. The actual parameter is typically a string which contains the name of the remote variable.

Additionally the VAR_i parameter may also have an implementer defined data type named VAR_ADDR. A function REMOTE_VAR is defined to generate the access information for nested variables.