# NORME INTERNATIONALE

# INTERNATIONAL STANDARD

**CEI**
**IEC**
**9506-5**

Première édition
First edition
1999-07



**Systèmes d'automatisation industrielle –**
**Spécification de messagerie industrielle –**

**Partie 5:**
**Norme d'accompagnement pour les Automates Programmables**

**Industrial automation systems –**
**Manufacturing message specification –**

**Part 5:**
**Companion Standard for Programmable Controllers**

Numéro de référence
Reference number
ISO/IEC 9506-5:1999

## Numéros des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

## Publications consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

## Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement (Catalogue en ligne)*
- **Bulletin de la CEI**
  Disponible à la fois au «site web» de la CEI* et comme périodique imprimé.

## Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique*, la CEI 60417: *Symboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles,* et la CEI 60617: *Symboles graphiques pour schémas.*

* Voir adresse «site web» sur la page de titre.

## Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

## Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

## Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

## Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

# NORME INTERNATIONALE

# INTERNATIONAL STANDARD

**CEI**
**IEC**
**9506-5**

Première édition
First edition
1999-07



## Systèmes d'automatisation industrielle – Spécification de messagerie industrielle –

## Partie 5: Norme d'accompagnement pour les Automates Programmables

## Industrial automation systems – Manufacturing message specification –

## Part 5: Companion Standard for Programmable Controllers



International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http: //www.iec.ch

CODE PRIX
PRICE CODE XA

*Pour prix, voir catalogue en vigueur*
*For price, see current catalogue*

# SOMMAIRE

# CONTENTS

Articles Pages

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

# SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – SPÉCIFICATION DE MESSAGERIE INDUSTRIELLE –

## Partie 5: Norme d'accompagnement pour les Automates Programmables

## AVANT-PROPOS

1) La CEI (Commission Electrotechnique Internationale) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI, entre autres activités, publie des Normes internationales. Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible un accord international sur les sujets étudiés, étant donné que les Comités nationaux intéressés sont représentés dans chaque comité d'études.
3) Les documents produits se présentent sous la forme de recommandations internationales. Ils sont publiés comme normes, rapports techniques ou guides et agréés comme tels par les Comités nationaux.
4) Dans le but d'encourager l'unification internationale, les Comités nationaux de la CEI s'engagent à appliquer de façon transparente, dans toute la mesure possible, les Normes internationales de la CEI dans leurs normes nationales et régionales. Toute divergence entre la norme de la CEI et la norme nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a fixé aucune procédure concernant le marquage comme indication d'approbation et sa responsabilité n'est pas engagée quand un matériel est déclaré conforme à l'une de ses normes.
6) L'attention est attirée sur le fait que certains des éléments de la présente Norme internationale peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et de ne pas avoir signalé leur existence.

La Norme internationale ISO/CEI 9506-5 a été établie par le sous-comité 65B: Dispositifs, du comité d'études 65 de la CEI: Mesure et commande dans les processus industriels.

Le texte de cette norme est basé sur les documents suivants:

| FDIS | Rapport de vote |
|---|---|
| 65B/385/FDIS | 65B/389/RVD |

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

L'annexe A est donnée uniquement à titre d'information.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

# INDUSTRIAL AUTOMATION SYSTEMS – MANUFACTURING MESSAGE SPECIFICATION –

## Part 5: Companion Standard for Programmable Controllers

## FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.

3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical reports or guides and they are accepted by the National Committees in that sense.

4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.

5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9506-5 has been prepared by subcommittee 65B: Devices, of IEC technical committee 65: Industrial-process measurement and control.

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

| FDIS | Report on voting |
|---|---|
| 65B/385/FDIS | 65B/389/RVD |

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

Annex A is for information only.

# INTRODUCTION

La présente norme est une norme d'accompagnement pour les Automates Programmables (CSPC) de l'ISO/CEI 9506. Elle décrit les classes de conformité CSPC qui regroupent les Automates Programmables (PC) jouant le rôle à la fois de Client et de Serveur dans une architecture Client-Serveur. De plus, les noms de variables propres aux PC ainsi que les définitions externes sont spécifiées dans cette norme, utilisant la notation définie par l'ISO/CEI 9506-1.

# INTRODUCTION

This standard is designed to be a Companion Standard for Programmable Controllers (CSPC) to ISO/IEC 9506. It defines CSPC conformance classes for Programmable Controllers (PCs) acting as both a Client and a Server in a Client-Server relationship. In addition, PC-specific variable names and external definitions are declared in this standard, using the notation defined in ISO/IEC 9506-1.

# SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – SPÉCIFICATION DE MESSAGERIE INDUSTRIELLE –

## Partie 5: Norme d'accompagnement pour les Automates Programmables

## 1 Domaine d'application

La présente partie de l'ISO/CEI 9506 traite de la communication entre les Automates Programmables (PC) et d'autres équipements qui utilisent la Spécification de Messagerie Industrielle (MMS) telle qu'elle est décrite dans l'ISO/CEI 9506-1 et l'ISO/CEI 9506-2. Il faut que cette communication entre les équipements soit écrite en syntaxe abstraite Automate Programmable (PC). La MMS, utilisée telle qu'elle est décrite dans cette norme, fournit des services aux applications, comme cela est expliqué dans l'ISO 7498.

La présente norme précise l'utilisation et les effets de la MMS sur les Automates Programmables. Un Automate Programmable est un système électronique à fonctionnement numérique, conçu pour être utilisé dans un environnement industriel, qui possède une mémoire programmable lui permettant de stocker de façon interne les instructions qui permettront aux utilisateurs de mettre en oeuvre des fonctions spécifiques telles que de logique, de séquencement, de synchronisation, de comptage et d'arithmétique pour contrôler, via des entrées ou sorties numériques ou analogiques, différents types de machines ou de processus. L'Automate Programmable ainsi que les équipements associés sont conçus de façon à être facilement intégrés à un système de contrôle industriel et à offrir une utilisation aisée de toutes les fonctions dont ils ont été dotés.

Le Client peut être un PC ou tout autre équipement. Le Serveur, quant à lui, est un PC. Quel que soit le type de l'équipement, toute communication MMS n'utilisant pas la syntaxe abstraite CSPC n'entre pas dans le cadre de cette norme. Est exclue également toute communication non MMS entre équipements.

## 2 Références normatives

Les documents normatifs suivants contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente partie de l'ISO/CEI 9506. Pour les références datées, les amendements ultérieurs ou les révisions de ces publications ne s'appliquent pas. Toutefois, les parties prenantes aux accords fondés sur la présente partie de l'ISO/CEI 9506 sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Pour les références non datées, la dernière édition du document normatif en référence s'applique. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur.

CEI 61131-1, *Automates programmables – Partie 1: Informations générales*

CEI 61131-2, *Automates programmables – Partie 2: Spécifications et essais des équipements*

CEI 61131-3, *Automates programmables – Partie 3: Langages de programmation*

ISO/CEI 7498, *Systèmes de traitement de l'information – Interconnexion de systèmes ouverts – Modèle de référence de base*

ISO/CEI 8822:1994, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Définition du service de présentation*

# INDUSTRIAL AUTOMATION SYSTEMS – MANUFACTURING MESSAGE SPECIFICATION –

## Part 5: Companion Standard for Programmable Controllers

## 1 Scope

This part of ISO/IEC 9506 applies to the communication between Programmable Controllers and other devices using the Manufacturing Message Specification (MMS) defined in ISO/IEC 9506-1 and ISO/IEC 9506-2. Specifically, this inter-device communication must be done in the Programmable Controller (PC) abstract syntax. MMS, used as defined in this standard, provides services to application processes as explained in ISO 7498.

This standard specifies the use and effects of MMS on Programmable Controllers. A programmable controller is a digitally operating electronic system, designed for use in an industrial environment, which uses a programmable memory for the internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting, and arithmetic, to control, through digital or analog inputs and outputs, various types of machines or processes. Both the programmable controller and its associated peripherals are designed so that they can be easily integrated into an industrial control system and easily used in all their intended functions.

The Client may be a PC or some other device. The Server is a PC. Regardless of the devices involved, MMS communication outside of the CSPC abstract syntax is outside the scope of this standard. Likewise, any non-MMS communication between devices is outside the scope of this standard.

## 2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9506. 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 ISO/IEC 9506 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 IEC and ISO maintain registers of currently valid International Standards.

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

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

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

ISO/IEC 7498, *Information processing systems – Open Systems Interconnection – Basic Reference Model*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1)*

ISO/CEI 8825:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécifications de règles de base pour coder la notation de syntaxe abstraite numéro UNE (ASN.1)*

ISO/CEI 9506, *Systèmes d'automatisation industrielle – Spécification de messagerie industrielle*

ISO/CEI 9506-1:1990, *Systèmes d'automatisation industrielle – Spécification de messagerie industrielle – Partie 1: Définition de service* (publiée actuellement en anglais seulement)
*Amendement 1:*1993, *Echange de données* (publiée actuellement en anglais seulement)

ISO/CEI 9506-2: 1990, *Systèmes d'automatisation industrielle – Spécification de messagerie industrielle – Partie 2: Spécification de protocole* (publiée actuellement en anglais seulement)
*Amendement 1:*1993, *Echange de données* (publiée actuellement en anglais seulement)

ISO 2382-1:1993, *Technologies de l'information – Vocabulaire – Partie 1: Termes fondamentaux*

## 3 Définitions

Pour les besoins de la présente partie de l'ISO/CEI 9506, les définitions suivantes s'appliquent:

**3.1**
**alarme**
événement qui signale une condition spécifique

**3.2**
**programme d'application du PC**
ensemble logique de tous les éléments et sous-ensembles d'instructions de programmation nécessaire au traitement des signaux destiné à commander une machine ou un processus au moyen d'un système PC
[CEI 61131-1:1992, définition 2.1 modifiée]

**3.3**
**fonction de stockage des programmes d'application**
le stockage des programmes d'application prévoit des espaces de mémoire pour stocker une série d'instructions dont l'exécution périodique ou déclenchée par un événement détermine la progression de la machine ou du procédé

**3.4**
**client**
entité de communication paire qui utilise le VMD (équipement virtuel de production) dans un but particulier quelconque par l'intermédiaire d'une instance de demande de service

**3.5**
**reprise à froid**
redémarrage du système PC et de son programme d'application après que toutes les données dynamiques (variables telles que l'image des entrées-sorties, registres internes, temporisateurs, compteurs, etc. et contextes du programme) ont été réinitialisées à un état prédéterminé. Une reprise à froid peut être automatique (par exemple après une coupure de courant, une perte d'information dans la ou les parties dynamiques de la ou des mémoires, etc.) ou manuelle (par exemple bouton de réinitialisation, etc.)
[CEI 61131-1:1992, définition 2.56 1) modifiée]

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding rules for Abstract Syntax Notation One (ASN.1)*

ISO/IEC 9506, *Industrial automation systems – Manufacturing message specification*

ISO/IEC 9506-1:1990, *Industrial automation systems – Manufacturing message specification – Part 1: Service definition*
Amendment 1:1993, Data Exchange

ISO/IEC 9506-2:1990, *Industrial automation systems – Manufacturing message specification – Part 2: Protocol specification*
Amendment 1:1993, Data Exchange

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

## 3 Definitions

For the purpose of this part of ISO/IEC 9506, the following definitions apply:

**3.1**
**alarm**
event which signals a specific condition

**3.2**
**PC application program**
logical assembly of all the programming language elements and constructs necessary for the intended signal processing required for the control of a machine or process by a PC-system
[IEC 61131-1:1992, definition 2.1]

**3.3**
**application program storage function**
application program storage provides for memory locations to store a series of instructions whose periodic or event-driven execution determines the progression of the machine or the process

**3.4**
**client**
peer communicating entity which makes use of the VMD (Virtual Manufacturing Device) for some particular purpose via a service request instance
[ISO/IEC 9506-1:1990, definition 5.10]

**3.5**
**cold restart**
restart of the PC-system and its application program after all dynamic data (variables such as I/O image, internal registers, timers, counters, etc. and program contexts) are reset to a predetermined state. A cold restart may be automatic (e.g. after a power failure, a loss of information in the dynamic portion(s) of the memory(ies), etc.) or manual (e.g. push-button reset, etc.)
[IEC 61131-1:1992, definition 2.56 1)]

**3.6**
**fonction de communication**
la fonction de communication permet l'échange de données avec d'autres systèmes (équipements tiers) tels que d'autres systèmes PC, des commandes de robots, des ordinateurs, etc.

**3.7**
**commande**
action délibérée sur ou dans un système en vue d'atteindre des objectifs définis

NOTE – La commande peut comprendre la surveillance et la sauvegarde au-delà de l'action de commande elle-même.

**3.8**
**donnée**
représentation réinterprétable d'une *information* sous une forme conventionnelle convenant à la communication, à l'interprétation ou au traitement

NOTE – Les données peuvent être traitées par des moyens humains ou *automatiques*.

[ISO 2382-1:1993, définition 01.01.02]

**3.9**
**acquisition des données**
collecte de données devant servir à la surveillance du procédé et à la génération de comptes rendus

**3.10**
**fonction de stockage de données**
le stockage des données d'application fournit des espaces de mémoire pour stocker un tableau d'images d'entrées-sorties et des données (par exemple les valeurs fixées pour les temporisateurs, les compteurs, les conditions d'alarme, les paramètres et les recettes pour la machine ou le procédé) nécessaires lors de l'exécution du programme d'application

**3.11**
**représentation directe**
moyen de représenter une variable dans un système d'automate programmable à partir duquel une correspondance, définie par le fabricant, avec un *emplacement logique* ou physique peut être déterminée directement
[CEI 61131-3:1993, définition 1.3.23]

**3.12**
**domaine**
objet abstrait qui représente un sous-ensemble des capacités d'un VMD qui est utilisé dans un but spécifique

**3.13**
**téléchargement**
procédé de transfert du contenu d'un domaine, y compris des objets subordonnés, à un utilisateur de MMS, par l'intermédiaire de données de chargement

**3.14**
**gestion d'événement**
gestion de conditions d'événement, d'actions d'événement et de déroulements d'événement

**3.15**
**exécution**
processus de réalisation des opérations d'une partie de programme d'application spécifiée
[CEI 61131-1:1992, définition 2.22 modifiée]

**3.6**
**communication function**
communication function provides the data exchange with other systems (third party devices) such as other PC-systems, robot controllers, computers, etc.

**3.7**
**control**
purposeful action on or in a system, to meet specified objectives

NOTE – Control may include monitoring and safeguarding besides the control action itself.

**3.8**
**data**
re-interpretable representation of *information* in a formalized manner suitable for communication, interpretation, or processing

NOTE – Data can be processed by humans or by *automatic* means.

[ISO 2382-1:1993, definition 01.01.02]

**3.9**
**data acquisition**
collection of data for the purpose of process monitoring and report generation

**3.10**
**data storage function**
application data storage provides for memory locations to store I/O image table and data (e.g. set values for timers, counters, alarm conditions, parameters and recipes for the machine or the process) required during the execution of the application program

**3.11**
**direct representation**
means of representing a variable in a programmable controller program from which a manufacturer-specified correspondence to a physical or *logical location* may be determined directly
[IEC 61131-3:1993, definition 1.3.23]

**3.12**
**domain**
abstract object that represents a subset of the capabilities of a VMD which is used for a specific purpose
[ISO/IEC 9506-1:1990, definition 3.4.8]

**3.13**
**download**
process of transferring the content of a domain, including any subordinate objects, via load data to an MMS-user
[ISO/IEC 9506-1:1990, definition 3.4.10]

**3.14**
**event management**
management of event conditions, event actions, and event enrollments
[ISO/IEC 9506-1:1990, definition 3.4.11]

**3.15**
**execution**
process of performing the operations of a specified portion of an application program
[IEC 61131-1:1992, definition 2.22]

**3.16**
**fonction d'interface pour les capteurs et les actionneurs**
la fonction d'interface pour les capteurs et les actionneurs qui convertit

– les signaux d'entrée et/ou les données provenant de la machine/du procédé à des niveaux appropriés pour le traitement;

– les signaux de sortie et/ou les données provenant de la fonction de traitement des signaux à des niveaux appropriés à la conduite des actionneurs et/ou des dispositifs d'affichage

**3.17**
**commande verrouillée**
commande visant à synchroniser les échanges de données entre deux machines. A différents moments, l'une des machines attend que l'autre lui envoie des données spécifiques

**3.18**
**local**
interne au PC; opposé de «distant»

**3.19**
**interface homme-machine (MMI)**
périphérique catalogué par le constructeur et équipé de boutons-poussoirs, de voyants, de claviers, d'écrans ou de dispositifs équivalents, destinés à servir d'interface opérateur, tels que pupitre de contrôle/commande de moteur, interface opérateur à usage général, etc.

Les interfaces homme-machine peuvent faire partie de l'installation permanente (par exemple, être montés en face avant, sur des portes, des panneaux, etc.) ou non.

[CEI 61131-1:1992, définition 2.33]

**3.20**
**système d'exploitation**
fonctions fournies par le constructeur, destinées à gérer les fonctions internes et inter-dépendantes du système PC
[CEI 61131-1:1992, définition 2.38]

**3.21**
**commande paramétrique**
commande qu'effectue le Client en écrivant des variables de commande résidant sur le PC

**3.22**
**fonctions d'alimentation**
les fonctions d'alimentation pourvoient à la conversion et à l'isolation de l'alimentation du système PC du secteur

**3.23**
**contexte de présentation**
association d'une syntaxe abstraite et d'une syntaxe de transfert

NOTE 1 – Du point de vue de l'utilisateur du service de présentation, un contexte de présentation représente un environnement selon lequel les valeurs de données de la syntaxe abstraite peuvent être transférées (sous forme d'une chaîne binaire) sans ambiguïté.

NOTE 2 – Quand la syntaxe abstraite l'autorise, une valeur de données de présentation peut contenir des champs imbriqués, dont chacun véhicule une valeur de données de présentation exprimée selon une syntaxe abstraite (éventuellement différente).

NOTE 3 – Du point de vue de l'utilisateur du service de présentation, un contexte de présentation représente une utilisation spécifique d'une syntaxe abstraite. Plusieurs contextes de présentation peuvent être définis pour la même syntaxe abstraite (avec la même syntaxe de transfert ou avec des syntaxes de transfert différentes); des valeurs de données de présentation transmises exprimées selon des contextes de présentation séparés sont remises selon ces contextes de présentation séparés.

[ISO/CEI 8822:1994, définition 3.4.7]

**3.16**
**interface function to sensors and actuators**
interface function to sensors and actuators converts

– the input signals and/or data obtained from the machine/process to appropriate signal levels for processing;

– the output signals and/or data from the signal processing function to appropriate signal levels to drive actuators and/or displays

**3.17**
**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

**3.18**
**local**
internal to the PC; opposite of remote

**3.19**
**man-machine interface (MMI)**
manufacturer's catalogued peripheral equipped with pushbuttons, lamps, keyboards, displays or equivalent, intended as operator interface, such as motor control/monitor panel, general purpose operator interface, etc.

MMIs may be part of the permanent installation (e.g. mounted on front panels, doors, boards, etc.) or not.
[IEC 61131-1:1992, definition 2.33]

**3.20**
**operating system**
manufacturer's provided functions intended to manage internal PC-system interdependent functions
[IEC 61131-1:1992, definition 2.38]

**3.21**
**parametric control**
control by the Client writing to control variables residing in the PC

**3.22**
**power supply functions**
power supply functions provide for conversion and isolation of the PC-system power from the mains supply

**3.23**
**presentation context**
association of an abstract syntax with a transfer syntax

NOTE 1 – From the viewpoint of the presentation-service-user, a presentation context represents an environment in which the presentation data values of the abstract syntax can be transferred (as a bitstring) without ambiguity.

NOTE 2 – Where the abstract syntax permits it, a presentation data value may contain embedded fields, each of which carries a presentation data value from a (possibly different) abstract syntax.

NOTE 3 – From the viewpoint of the presentation-service-user, a presentation context represents a specific use of an abstract syntax. Multiple presentation contexts may be defined for the same abstract syntax (with the same or different transfer syntaxes); presentation data values transmitted in these separate presentation contexts are also delivered in these separate presentation contexts.

[ISO/IEC 8822:1994, definition 3.4.7]

**3.24**
**unité de traitement**
partie du système PC qui assure le stockage du programme d'application et des données, et l'exécution du programme d'application. Un système PC dispose d'une ou de plusieurs unités de traitement

**3.25**
**invocation de programme (PI)**
objet abstrait qui représente un élément dynamique correspondant de façon optimale à une tâche d'exécution dans un environnement multitâche, composé d'un ensemble de domaines

**3.26**
**automate programmable (PC)**
système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en oeuvre de fonctions spécifiques, telles que des fonctions logiques, de séquencement, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties tout-ou-rien ou analogiques divers types de machines ou de processus. Le PC et ses périphériques associés sont conçus pour pouvoir facilement s'intégrer à un système d'automatisme industriel et être facilement utilisés dans toutes les fonctions prévues
[CEI 61131-1:1992, définition 2.50]

**3.27**
**système d'automate programmable (système PC)**
configuration réalisée par l'utilisateur et comportant un automate programmable et des périphériques associés, nécessaires au système automatisé prévu. Elle consiste en des équipements interconnectés au moyen de câbles ou de raccords enfichables pour l'installation permanente, et au moyen de câbles ou d'autres moyens pour des périphériques portables et transportables

**3.28**
**outil de programmation et de mise au point (PADT)**
périphérique catalogué servant de moyen de programmation, d'essai, de mise en service et de recherche de défauts pour l'application du système PC, de documentation et stockage du programme, et pouvant être éventuellement utilisé comme interfaces homme-machine (MMI). Les outils de programmation et de mise au point (PADT) sont dits raccordables lorsqu'ils peuvent être à tout moment raccordés à, ou retirés de leur interface associée sans aucun risque pour les opérateurs et pour l'application. Dans tous les autres cas, les outils de programmation et de mise au point sont considérés comme fixes
[CEI 61131-1:1992, définition 2.52]

**3.29**
**fonctions de programmation, de mise au point des programmes d'application et d'essai**
ces fonctions prévoient la création et le chargement, la surveillance, l'essai et la mise au point d'un programme d'application, aussi bien que sa documentation et son archivage

**3.30**
**déclaration pro forma de conformité de réalisation (PICS)**
les quatre parties de la PICS sont les suivantes:

a) la première partie de la PICS doit fournir des informations sur la réalisation et le système;

b) la deuxième partie de la PICS doit indiquer quels sont les services CBB fournis;

c) la troisième partie de la PICS doit indiquer quels sont les paramètres CBBs utilisés, ainsi que leurs valeurs, si nécessaire;

d) la quatrième partie de la PICS doit fournir les valeurs utilisées localement.

**3.24**
**processing unit**
portion of a PC system 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

**3.25**
**program invocation (PI)**
abstract object representing a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment, which is composed of a set of domains
[ISO/IEC 9506-1:1990, definition 3.4.26]

**3.26**
**programmable controller (PC)**
digitally operating electronic system, designed for use in an industrial environment which uses a programmable memory for the internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting, and arithmetic, to control, through digital or analog inputs and outputs, various types of machines or processes. Both the PC and its associated peripherals are designed so that they can be easily integrated into an industrial control system and easily used in all their intended functions
[IEC 61131-1:1992, definition 2.50]

**3.27**
**programmable controller system (PC-system)**
user-built configuration, consisting of a programmable controller and associated peripherals, that is necessary for the intended automated system. It consists of units interconnected by cables or plug-in connections for permanent installation and by cables or other means for portable and transportable peripherals
[IEC 61131-1:1992, definition 2.51]

**3.28**
**programming and debugging tool (PADT)**
catalogued peripheral to assist in programming, testing, commissioning and trouble-shooting the PC-system application, program documentation and storage and possibly to be used as MMIs. PADTs are said to be pluggable when they may be plugged or unplugged at any time into their associated interface, without any risk to the operators and the application. In all other cases, PADTs are said to be fixed
[IEC 61131-1:1992, definition 2.52]

**3.29**
**programming, debugging and testing function**
these functions provide for application program generation and loading, monitoring, testing and debugging as well as for application program documentation and archiving

**3.30**
**protocol implementation conformance statement (PICS)**
there are four parts to the PICS:

a) part one of the PICS shall indicate information about the implementation and system;

b) part two of the PICS shall indicate which service CBBs are implemented;

c) part three of the PICS shall indicate which parameter CBBs are implemented, and their values if required by the CBB;

d) part four of the PICS shall indicate the local implementation values.

[ISO/IEC 9506-2:1990, definition 18.1]

**3.31**
**recette**
description des procédures ou données spécifiques à ces procédures, ou encore les deux à la fois, servant à réaliser un produit qui utilise le procédé ou l'équipement auquel est associé le contrôleur, et qui est différente pour chaque produit

**3.32**
**distant**
externe au PC; opposé de «local»

**3.33**
**serveur**
entité de communication paire qui se comporte comme un VMD pour une instance de demande de service donnée

**3.34**
**fonction de traitement des signaux**
la fonction de traitement des signaux comporte le stockage du programme d'application, le stockage des données, le système d'exploitation, l'exécution des fonctions du programme d'application.

Elle traite des signaux provenant de capteurs aussi bien que du stockage interne de données et émet des signaux vers les actionneurs aussi bien que vers le stockage interne des données conformément au programme d'application

**3.35**
**non sollicité**
réalisé sans avoir été demandé par une requête explicite

**3.36**
**télésauvegarde**
procédé de transfert du contenu d'un domaine, y compris de tous les objets subordonnés, par l'intermédiaire d'un chargement de données provenant d'un utilisateur distant, de façon à permettre un téléchargement ultérieur

**3.37**
**variable**
un ou plusieurs éléments de donnée référencés sous le même nom ou la même description

**3.38**
**équipement virtuel de production (VMD)**
portion d'un processus de traitement de l'information (ensemble des fonctions matérielles et logicielles) qui rend accessibles les ressources et les fonctionnalités de l'équipement réel de production aux processus d'application clients chargés de son suivi ou de son pilotage (ou des deux). Tout processus d'application peut inclure zéro, un ou plusieurs VMD, mais un processus d'application ne contenant aucun VMD ne peut se comporter en Serveur.

Tout VMD représente un équipement virtuel de production dans un AP, et chaque VMD est logiquement disjoint de tous les autres VMD
[ISO/CEI 9506-1, article 7]

**3.39**
**reprise à chaud**
reprise après une défaillance d'alimentation, avec un ensemble de données dynamiques prédéterminé et programmé par l'utilisateur et un contexte du programme d'application prédéterminé par le système. Une reprise à chaud est identifiée par une signalisation d'état ou tout autre moyen équivalent mis à la disposition du programme d'application, indiquant que l'interruption d'alimentation du système PC a été détectée en fonctionnement
[CEI 61131-1:1992, définition 2.5.63]

**3.31**
**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 that is different from a previous product

**3.32**
**remote**
external to the PC; opposite of local

**3.33**
**server**
peer communicating entity which behaves as a VMD for a particular service request instance
[ISO/IEC 9506-1:1990, definition 5.10]

**3.34**
**signal processing function**
signal processing function consists of the application program storage, the data storage, the operating system, the execution of the application program functions.

It processes signals obtained from sensors as well as internal data storage and generates signals to actuators as well as internal data storage in accordance with the application program

**3.35**
**unsolicited**
performed without an explicit request

**3.36**
**upload**
process of transferring the content of a domain, including any subordinate objects, via load data from a remote user, in such a manner as to allow subsequent download
[ISO/IEC 9506-1:1990, definition 3.4.37]

**3.37**
**variable**
one or more data elements that are referred to together by a single name or description
[ISO/IEC 9506-1:1990, definition 3.4.38]

**3.38**
**virtual manufacturing devices (VMD)**
that portion of an information processing task which makes available – for control, or monitoring, or both – a set of resources and functionnality associated with a seal manufacturing device. An application process may contain zero or more VMDs. If it does not define a VMD it may not act as an MMS-server.

Each VMD represents a virtual manufacturing device within that AP, and each VMD is logically separate from all other VMDs
[ISO/IEC 9506-1, clause 7]

**3.39**
**warm restart**
restart after a power failure with a user programmed predetermined set of dynamic data and system predetermined application program context. A warm restart is identified by a status flag or equivalent means made available to the application program indicating that the power failure shut down of the PC-system was detected in the run mode
[IEC 61131-1:1992, definition 2.5.63)]

## 4 Abréviations

Certaines abréviations sont fréquemment utilisées dans cette norme. Quelques uns de ces termes sont définis ou référencés à l'article 3.

ASN.1 : (Abstract Syntax Notation One) Notation un de syntaxe abstraite, telle que définie dans l'ISO/CEI 8824 et l'ISO/CEI 8825

CBB : (Conformance Building Block) Bloc élémentaire de conformité

Cnf : Primitive de confirmation d'un service de communications

CSPC : (Companion Standard for Programmable Controllers) Norme d'accompagnement pour Automates Programmables, sujet de la présente norme

I/O : Entrée/sortie

Ind : Primitive d'indication d'un service de communications

MMS : (Manufacturing Message Specification) Spécification de Messagerie Industrielle (ISO/CEI 9506-1 et ISO/CEI 9506-2)

PADT : (Programming And Debugging Tool) 0util de programmation et de mise au point

PC : (Programmable Controller) Automate Programmable

PICS : (Protocol Implementation Conformance Statement) Déclaration pro forma de conformité de réalisation

PU : (Processing Unit) Unité de traitement

Req : Primitive de demande d'un service de communications

Rsp : Primitive de réponse à un service de communications

VMD : (Virtual Manufacturing Device) Equipement virtuel de production

## 5 Description de l'application

Cet article décrit le contexte dans lequel s'inscrit un Automate Programmable (PC) dans un système et identifie les fonctions que le PC offre au système d'automatisation. Le paragraphe 5.1 fournit un modèle d'application montrant comment un PC est relié à d'autres équipements dans un système de Contrôle-Commande plus important. Elle décrit également les sous-systèmes d'un PC standard. Le paragraphe 5.2 dresse la liste des fonctions générales et décrit la fonction PC spécifique qui est fournie au système.

### 5.1 Modèles spécifiques à l'application

#### 5.1.1 Modèle de communication du PC

Un Automate Programmable fournit certaines fonctions d'application spécifiques au reste du système d'automatisation. Il a aussi besoin de fonctions provenant d'autres Automates Programmables.

Le schéma suivant illustre les équipements dans un réseau de communication, et montre trois équipements (dénommés Client) qui demandent des fonctions PC du PC 2. Les deux PC illustrés en gras s'inscrivent dans le contexte de cette norme.

## 4 Abbreviations

These are some abbreviations frequently used in this standard. Some of these terms are defined or referenced in clause 3.

| | | |
|---|---|---|
| ASN.1 | : | Abstract Syntax Notation One, as specified in ISO/IEC 8824 and ISO/IEC 8825 |
| CBB | : | Conformance Building Block |
| Cnf | : | This is the Confirmation primitive of a communications service |
| CSPC | : | Companion Standard for Programmable Controllers (the present part of ISO/IEC 9506). |
| I/O | : | Input/Output |
| Ind | : | This is the Indication primitive of a communications service |
| MMS | : | Manufacturing Message Specification (ISO/IEC 9506-1 and ISO/IEC 9506-2) |
| PADT | : | Programming And Debugging Tool |
| PC | : | Programmable Controller |
| PICS | : | Protocol Implementation Conformance Statement |
| PU | : | Processing Unit |
| Req | : | This is the Request primitive of a communications service |
| Rsp | : | This is the Response primitive of a communications service |
| VMD | : | Virtual Manufacturing Device |

## 5 Application description

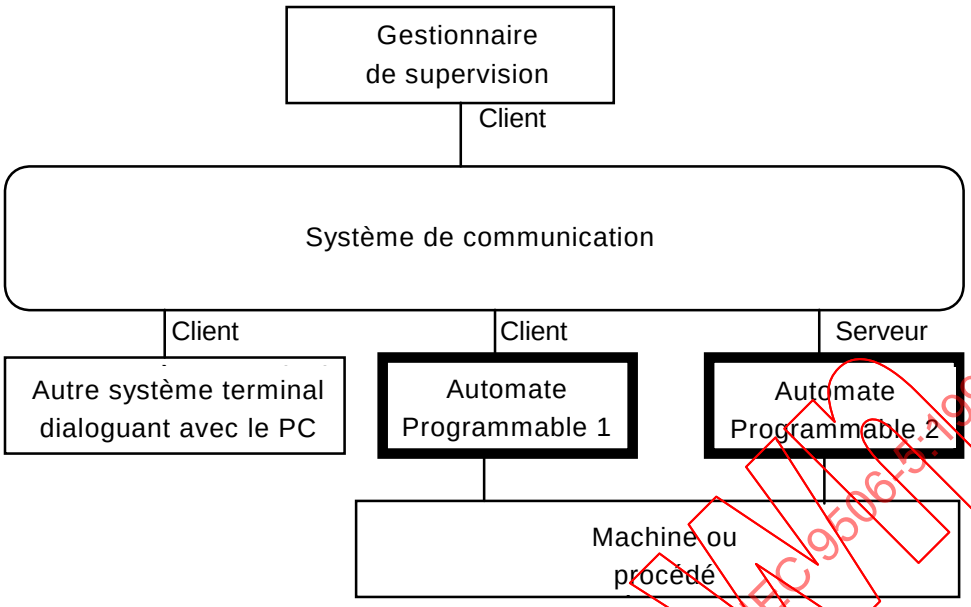
This clause provides the context of a Programmable Controller (PC) in a system and identifies the functions the PC provides to the automation system. Subclause 5.1 provides an application model to show how a PC relates to other devices in a larger control system. It also shows the subsystems of a typical PC. Subclause 5.2 contains a list of general functions and a description of the PC specific function the PC provides to the system.

### 5.1 Application-Specific Models

#### 5.1.1 PC Communication Model

A Programmable Controller supplies some specific application functions to the rest of the automation system. It also requests functions from other Programmable Controllers.

The following diagram illustrates the devices in a communication network, showing three possible devices that request PC functions (Clients) from PC 2. The two PCs which are highlighted in the figure below are in the scope of this standard.



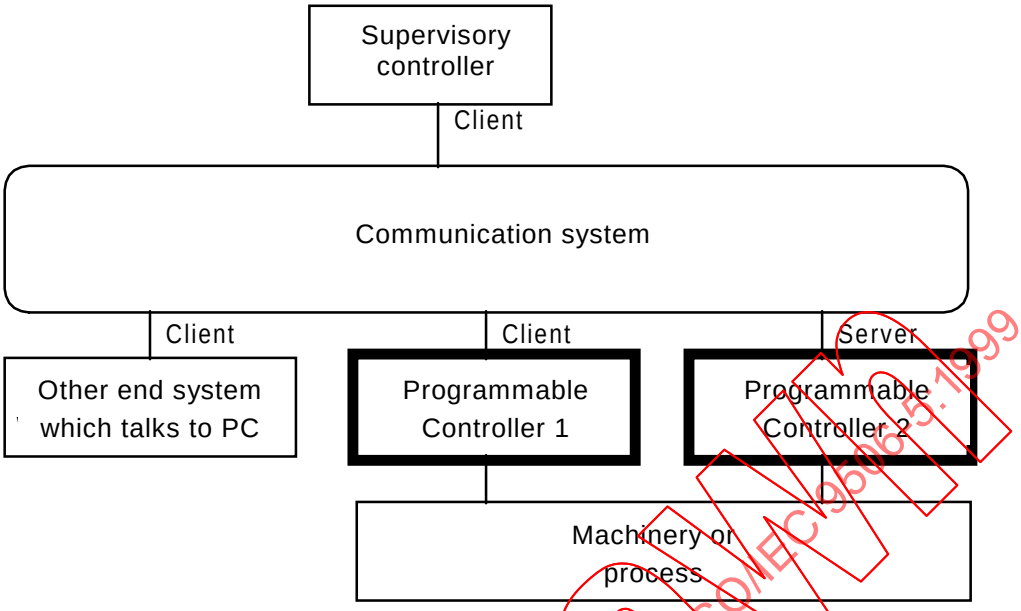

NOTE – Si l'on ne considère que le point de vue de la communication, le «gestionnaire de supervision» et l'«autre système terminal dialoguant avec le PC» qui sont mentionnés dans la figure 1 affichent le même comportement envers un Serveur de communication PC, à savoir qu'ils émettent des requêtes vers le PC.

**Figure 1 – Modèle de communication du PC**

### 5.1.2 Modèle fonctionnel du PC

Un PC dispose de plusieurs fonctions (voir figure 2). Pour un PC entrant dans le cadre de cette norme, une des fonctions de communication au moins assure l'interface avec l'environnement MMS.

La figure 2 est tirée de la CEI 61131-1. Elle sert à illustrer certaines fonctions (sous-systèmes) d'un PC standard.




NOTE – From the communication viewpoint the "Supervisory Controller" and the "Other End System Which Talks to PC" mentioned in figure 1 exhibit the same behavior to a PC communication server, that is they submit requests to the PC.

**Figure 1 – PC communication model**

### 5.1.2 PC Functional Model

A PC consists of several functions; see figure 2. For a PC within the scope of this standard, at least one communication function is an interface to the MMS environment.

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

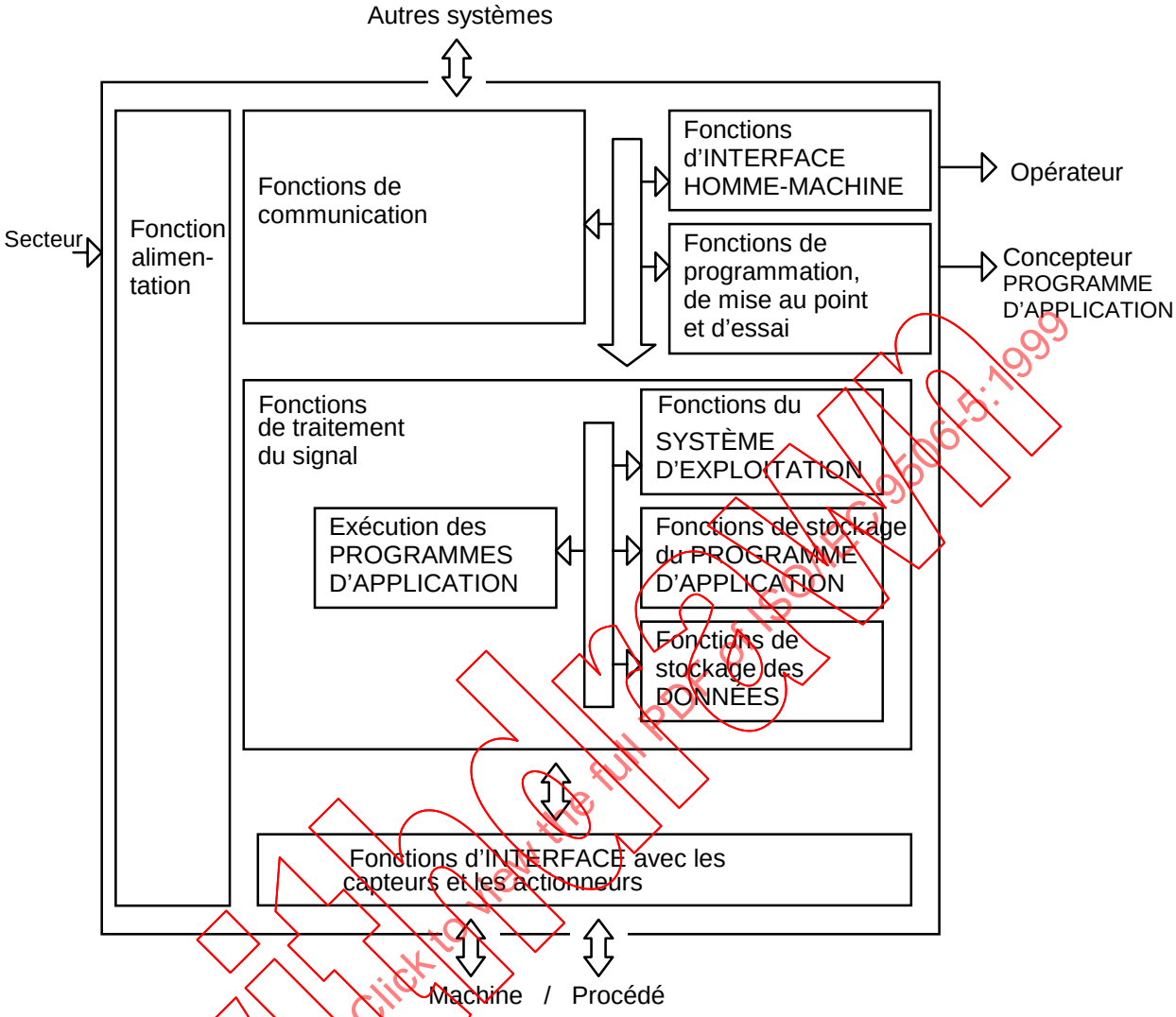



**Figure 2 – Modèle fonctionnel d'un Automate Programmable**

Il existe une fonction faisant partie du système PC, mais que l'on retrouve classiquement à l'extérieur du PC lui-même, et que l'on appelle PADT (outil de programmation et de mise au point). Ce PADT est conçu pour communiquer avec le PC via la fonction de communication. La présente norme d'accompagnement ne prescrit ni ne requiert l'utilisation de MMS pour cette communication.

La fonction d'interface aux capteurs et aux actionneurs peut avoir des entrées/sorties locales ou distantes par rapport à l'Unité de traitement principale (voir 5.1.3 pour le modèle de configuration matérielle du PC). La fonction d'interface aux capteurs et aux actionneurs possède deux attributs pour chaque programme d'application qui définit comment le PC surveille et commande la machine/le procédé. L'attribut d'entrée est caractérisé par les états suivants:

– les entrées fournies au programme d'application proviennent des capteurs;
– les entrées fournies au programme d'application sont maintenues en l'état.

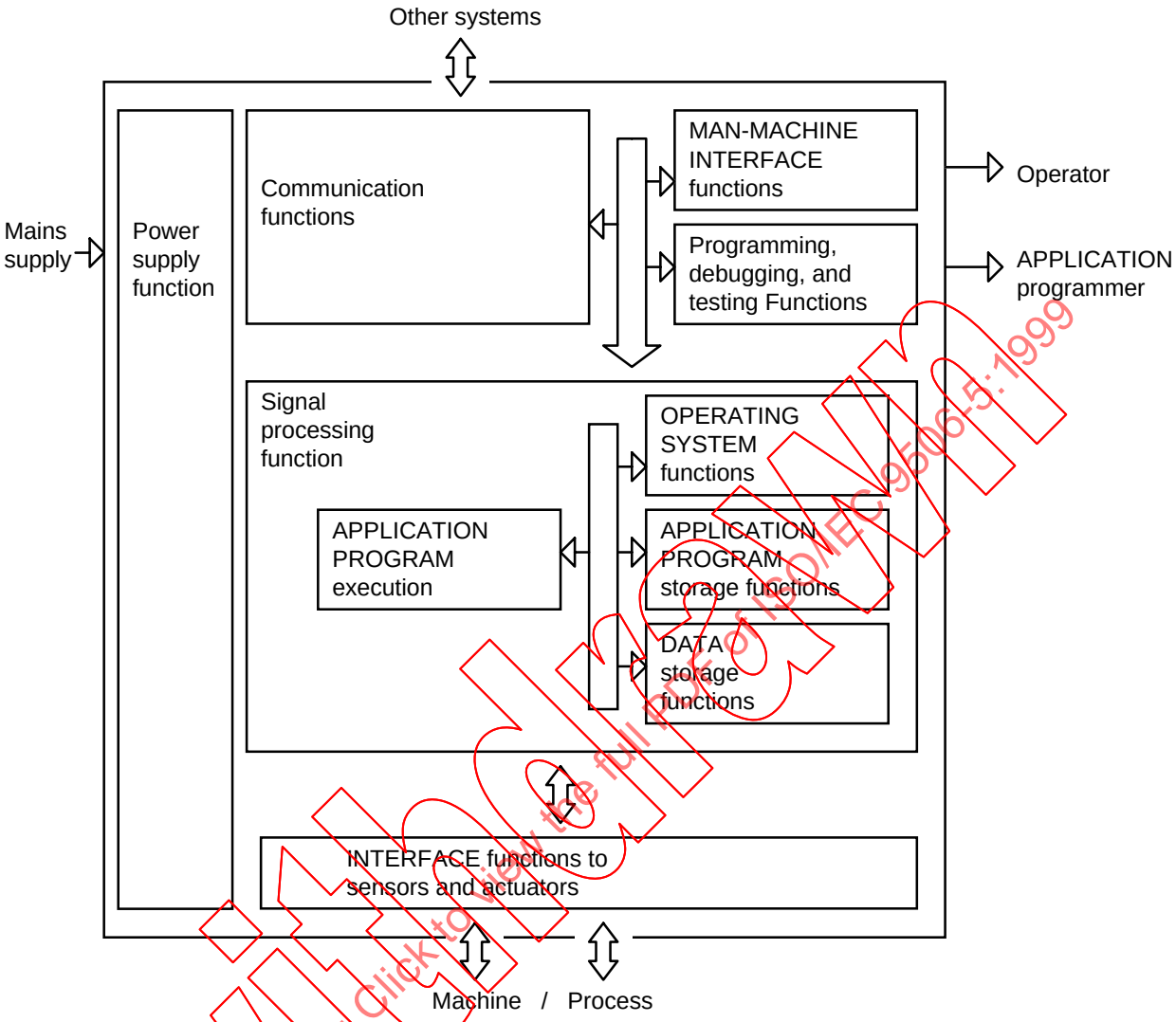



**Figure 2 – Programmable Controller functional model**

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 modeled as interacting with the PC via the communications function. This companion standard does not specify nor require use of MMS for this interaction.

The interface function to sensors and actuators can have I/O which are local or remote to the main processing unit (see 5.1.3 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.

L'attribut de sortie est caractérisé par les états suivants:

– les actionneurs sont commandés par le programme d'application;
– les actionneurs sont maintenus en l'état.

### 5.1.3 Modèle de configuration matérielle du PC

La figure 3 illustre le modèle de configuration matérielle du PC. Elle montre les modules qui constituent un PC. Un sous-système PC est doté d'un ou de plusieurs modules.

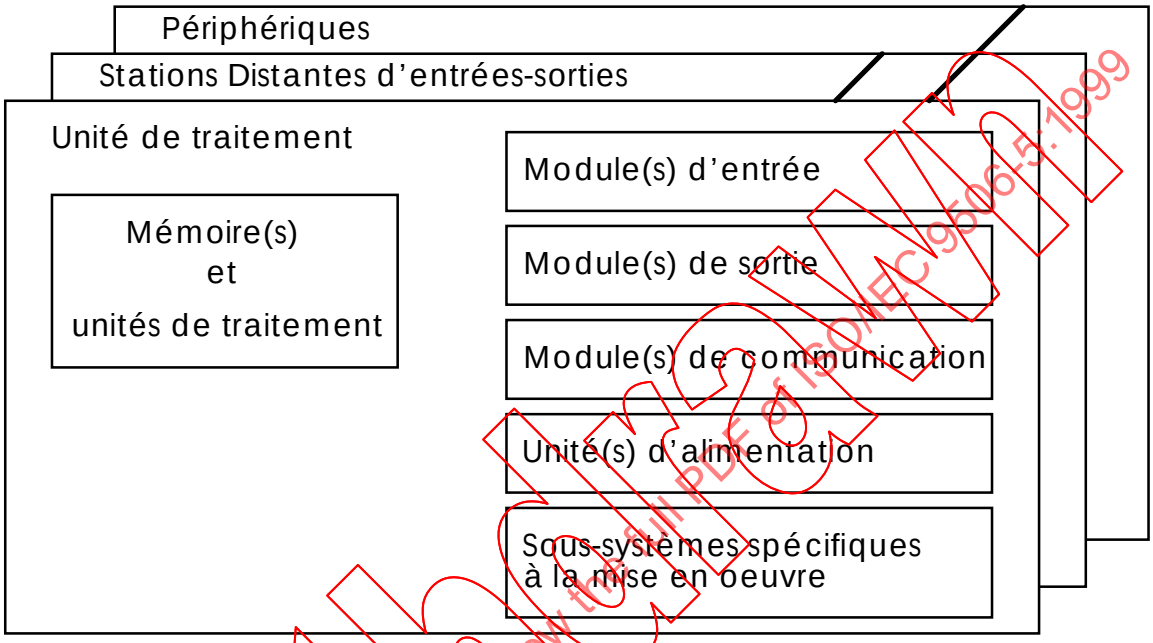



**Figure 3 – Modèle de configuration matérielle d'un Automate Programmable**

### 5.1.4 Etats du PC

Un PC peut fournir des statuts qui incluent des informations sur l'état et des indications d'anomalies.

Des statuts peuvent être établis sur certains sous-systèmes identifiés dans la figure 3. De plus, un statut récapitulatif fournit des informations générales concernant le PC. Les sous-systèmes qui fournissent des statuts sont les suivants:

a) les sous-systèmes d'entrées-sorties (modules d'entrée et de sortie inclus, ainsi que d'autres équipements d'entrées-sorties intelligents),
b) les unités de traitement,
c) les alimentations,
d) les sous-systèmes de mémorisation,
e) les sous-systèmes de communication, et
f) les sous-systèmes spécifiques à la mise en oeuvre.

NOTE – Les statuts sont prévus pour fournir des informations sur l'automate et sur les sous-systèmes matériels et logiciels, quelle que soit la configuration. Ils ne sont pas censés fournir des informations sur le procédé commandé ni sur le programme d'application du PC. Les données des statuts relatent l'état et la santé du PC et de ses sous-systèmes.

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.

### 5.1.3 PC Hardware Model

Figure 3 shows the PC hardware model. It shows the modules that make up a PC. A PC subsystem consists of one or more modules.

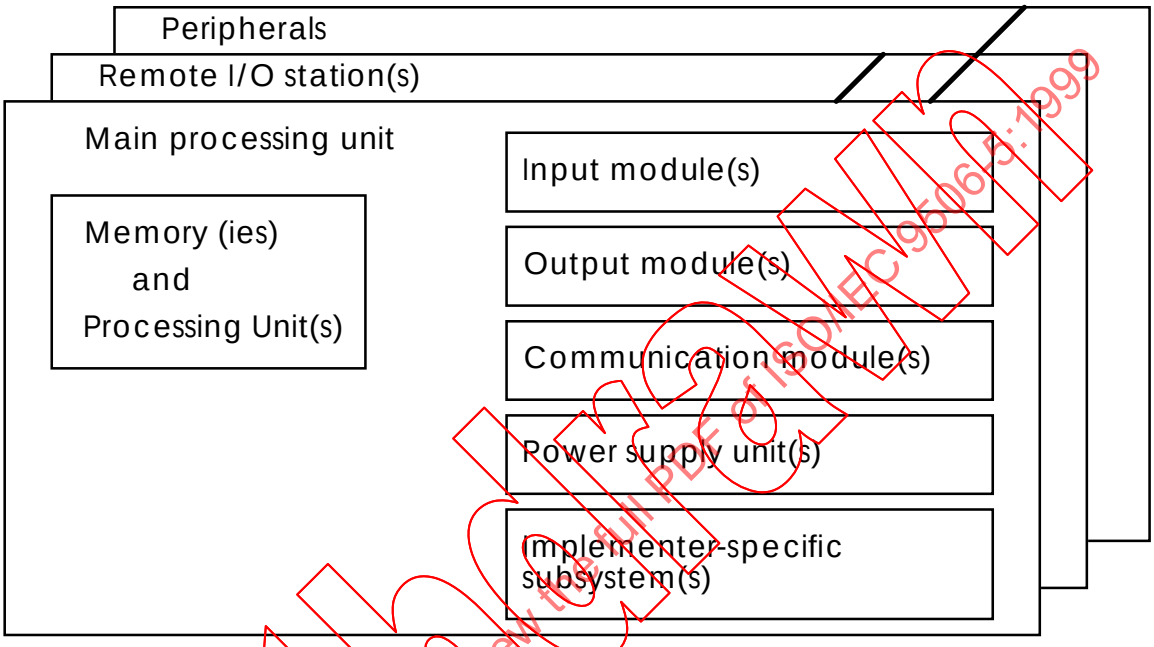



**Figure 3 – Programmable Controller hardware model**

### 5.1.4 PC Status

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

Status can be reported on some of the subsystems identified in figure 3. In addition, there is a summary status that provides general information about the PC. The subsystems that provide status are the following:

a) I/O Subsystems (includes Input and Output modules and other intelligent I/O devices),
b) Processing Units,
c) Power Supplies,
d) Memory Subsystems,
e) Communication Subsystems, and
f) 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.

Deux concepts relatifs aux statuts sont utilisés dans cette norme: la santé et l'état. La «santé» d'un PC ou de ses sous-systèmes est décrite par une, et une seule, des trois valeurs possibles. La sémantique de chacune des valeurs est définie ci-après. Elles sont classées par ordre décroissant de «santé»:

a) GOOD – le PC (ou le sous-système spécifié) n'a détecté aucun incident qui l'empêcherait de réaliser la fonction souhaitée;

b) WARNING – le PC (ou le sous-système spécifié) n'a détecté aucun incident qui l'empêcherait de réaliser la fonction souhaitée, mais il en a détecté au moins un qui pourrait limiter ses capacités. Cette limite peut être temporelle, de performance, etc. (voir les instructions suivantes pour une définition plus complète de ces limites);

c) BAD – le PC (ou le sous-système spécifié) a détecté au moins un incident qui pourrait l'empêcher de réaliser la fonction souhaitée.

L'«état» d'un système PC est indiqué par une liste d'attributs, chacun d'entre eux pouvant être VRAI ou FAUX. Aucun attribut, un seul attribut ou plusieurs d'entre eux peuvent être VRAI au même instant. La sémantique de chaque attribut est définie dans ce paragraphe.

Chaque information de statut peut être accompagnée par une extension définie par le concepteur. Voici quelques exemples d'information que le concepteur peut ajouter:

a) autres diagnostics d'anomalies (par exemple cycles d'écriture EEPROM dépassés);

b) autres états de fonctionnement (par exemple Auto-calibrage autorisé);

c) état de la clé locale (par exemple Auto-reprise autorisée).

Aucune implémentation n'est nécessaire à l'obtention de l'état des sous-systèmes. Toutes les instances des types similaires de sous-systèmes d'un même système sont signalées séparément. Le nom du sous-système peut être fourni, permettant ainsi de différencier les sous-systèmes de même type.

La correspondance des informations d'état avec les variables MMS est spécifiée à l'article 6.

La spécification détaillée de ces variables est décrite à l'article 8.

#### 5.1.4.1 Sous-système d'entrées-sorties

L'objet I/O Subsystem définit les statuts d'un sous-système d'entrées-sorties.

```
Object: I/O Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: No Outputs Disabled (TRUE, FALSE)
  Attribute: No Inputs Disabled (TRUE, FALSE)
  Attribute: I/O Forced (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

##### 5.1.4.1.1 Subsystem Name

L'attribut Subsystem Name identifie le sous-système.

##### 5.1.4.1.2 Health

Cet attribut permet de connaître l'état de santé du sous-système d'entrées-sorties.

a) GOOD – indique qu'aucune anomalie n'a été détectée dans le sous-système d'entrées-sorties.

b) WARNING – indique qu'une anomalie mineure a été détectée dans le sous-système d'entrées-sorties. Une telle anomalie pourrait être, par exemple, l'apparition d'anomalies réparables dans la communication avec une machine d'entrées-sorties distante.

There are two concepts used in this standard 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, the following:

a) GOOD – the PC (or the specified subsystem) has not detected any problems which would prohibit it from performing the intended function;

b) WARNING – 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 – 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 subclause.

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

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 enabled).

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.

The mapping of the status information onto MMS Variables is specified in clause 6.

The detailed specification of these variables is in clause 8.

#### 5.1.4.1 I/O Subsystem

The object I/O Subsystem models the status information of an I/O subsystem.

```
Object: I/O Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: No Outputs Disabled (TRUE, FALSE)
  Attribute: No Inputs Disabled (TRUE, FALSE)
  Attribute: I/O Forced (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

##### 5.1.4.1.1 Subsystem Name

The Subsystem Name identifies the subsystem.

##### 5.1.4.1.2 Health

This attribute identifies the health of the I/O Subsystem.

a) GOOD – indicates that there have been no errors detected in this I/O subsystem.

b) 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 chassis.

c) BAD – indique qu'une anomalie grave a été détectée dans le sous-système d'entrées-sorties. Une perte de communication avec une machine d'entrées-sorties distante pourrait être un exemple d'une telle anomalie.

Les termes «anomalie grave» et «anomalie mineure» doivent faire l'objet d'une définition par le concepteur.

#### 5.1.4.1.3 No Outputs Disabled

Si cet attribut est VRAI, il indique que le PC peut changer l'état physique de toutes les sorties associées au sous-système d'entrées-sorties spécifié par le programme d'application ou d'autres moyens. Si l'attribut n'est pas VRAI, l'état physique de certaines sorties ne peut pas être modifié (l'état logique peut être modifié). Cet attribut est généralement utilisé lors de la mise au point des programmes d'application dans le PC.

#### 5.1.4.1.4 No Inputs Disabled

Si cet attribut est VRAI, il indique que le PC peut connaître l'état physique de toutes les entrées associées au sous-système d'entrées-sorties spécifié par le programme d'application ou d'autres moyens. S'il n'est pas VRAI, l'état physique de certaines entrées ne peut être connu. Cet attribut est généralement utilisé lors de la mise au point des programmes d'application dans lesquels les entrées peuvent être simulées.

#### 5.1.4.1.5 I/O Forced

Si cet attribut est VRAI, il indique qu'au moins un point d'entrées-sorties associé au sous-système d'entrées-sorties a été forcé. Lorsqu'une entrée est forcée, le programme d'application recevra la valeur spécifiée par le PADT et non la valeur réelle de la machine ou du procédé. Lorsqu'une sortie est forcée, la machine ou le procédé recevra la valeur fournie par le PADT et non celle générée par l'exécution du programme d'application.

#### 5.1.4.1.6 List Of Implementer Specified State

Cet attribut indique la valeur d'état qui doit être précisée par le concepteur dans la PICS.

### 5.1.4.2 Unité de traitement

L'objet Processing Unit définit les statuts d'une unité de traitement.

```
Object: Processing Unit
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: Running (TRUE, FALSE)
  Attribute: Local Control (TRUE, FALSE)
  Attribute: No Outputs Disabled (TRUE, FALSE)
  Attribute: No Inputs Disabled (TRUE, FALSE)
  Attribute: User Application Present (TRUE, FALSE)
  Attribute: Forced (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.2.1 Subsystem Name

L'attribut Subsystem Name identifie le sous-système.

#### 5.1.4.2.2 Health

Cet attribut définit l'état de santé de l'unité de traitement. Le concepteur doit spécifier les conditions indiquées par les valeurs GOOD, WARNING et BAD.

c) 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 chassis.

The definition of major fault and minor fault shall be provided by the implementer.

#### 5.1.4.1.3 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 debugging of application programs in the PC.

#### 5.1.4.1.4 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 of some inputs cannot be accessed. This is typically used in the debugging of application programs where the inputs can be simulated.

#### 5.1.4.1.5 I/O Forced

If TRUE, this attribute indicates that at least one I/O point associated with the I/O 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.

#### 5.1.4.1.6 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.1.4.2 Processing Unit

The object Processing Unit models the status information of a processing unit.

```
Object: Processing Unit
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: Running (TRUE, FALSE)
  Attribute: Local Control (TRUE, FALSE)
  Attribute: No Outputs Disabled (TRUE, FALSE)
  Attribute: No Inputs Disabled (TRUE, FALSE)
  Attribute: User Application Present (TRUE, FALSE)
  Attribute: Forced (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.2.1 Subsystem Name

The Subsystem Name identifies the subsystem.

#### 5.1.4.2.2 Health

This attribute identifies the health of the processing unit. The implementer shall specify the conditions which are indicated by the GOOD, WARNING and BAD values.

#### 5.1.4.2.3 Running

Si l'attribut est VRAI, il indique qu'au moins une partie de l'application utilisateur a été chargée et qu'elle est contrôlée par l'unité de traitement.

#### 5.1.4.2.4 Local Control

Si l'attribut est VRAI, il indique si la commande de passage en local est active. Si c'est le cas, le contrôle de l'unité de traitement à partir du réseau peut être limité. Cela peut être, par exemple, fortement lié à l'utilisation d'une clé de commutation locale.

#### 5.1.4.2.5 No Outputs Disabled

Si cet attribut est VRAI, il indique que l'unité de traitement peut changer l'état physique de toutes les sorties contrôlées par l'unité de traitement par le programme d'application ou d'autres moyens. Si l'attribut n'est pas VRAI, l'état physique de certaines sorties ne peut pas être modifié (l'état logique peut être modifié). Cet attribut est généralement utilisé lors de la mise au point des programmes d'application dans l'unité de traitement.

#### 5.1.4.2.6 No Inputs Disabled

S'il est VRAI, cet attribut indique que l'unité de traitement peut connaître l'état physique de toutes les entrées accessibles, depuis cette unité de traitement suite à l'exécution du programme d'application ou d'autres moyens. Si cet attribut est FAUX, l'état physique de certaines entrées n'est pas accessible. Cela est généralement utilisé dans la mise au point des programmes d'application qui permettent la simulation des entrées.

#### 5.1.4.2.7 User Application Present

Si l'attribut est VRAI, il indique que l'unité de traitement a au moins une application utilisateur présente.

#### 5.1.4.2.8 Forced

Si l'attribut est VRAI, il indique que l'unité de traitement a au moins une variable forcée. Lorsqu'une variable est forcée, le programme d'application utilisera la valeur spécifiée par le PADT au lieu de celle générée par l'exécution normale du programme.

#### 5.1.4.2.9 List Of Implementer Specified State

Cet attribut indique les valeurs d'état qui doivent être précisées par le concepteur dans la PICS.

### 5.1.4.3 Sous-système d'alimentation

Le PC peut fournir des statuts sur chacun des sous-systèmes d'alimentation; la figure 4 montre une configuration possible de sous-système d'alimentation d'un PC. Les prescriptions relatives aux alimentations des systèmes PC et le comportement de ces alimentations sont décrits dans la CEI 61131-1 et dans la CEI 61131-2.

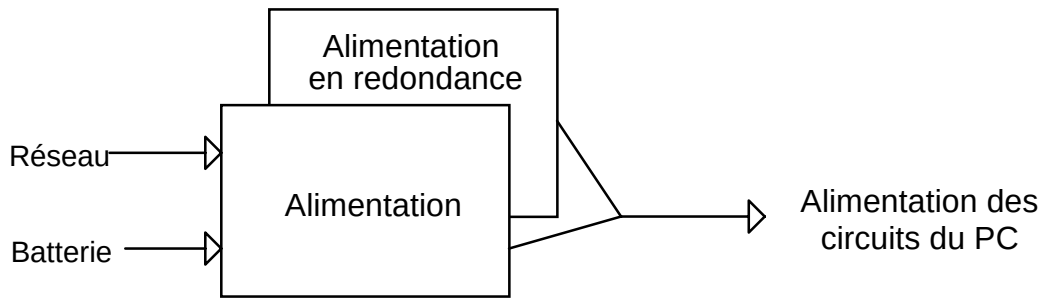



**Figure 4 – Alimentation d'un automate programmable**

#### 5.1.4.2.3 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.

#### 5.1.4.2.4 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.

#### 5.1.4.2.5 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 is not affected (logical state may be affected). This is typically used in the debugging of application programs in the PU.

#### 5.1.4.2.6 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 inputs cannot be accessed. This is typically used in the debugging of Application Programs where the inputs can be simulated.

#### 5.1.4.2.7 User Application Present

If TRUE, this attribute indicates that the Processing Unit has at least one User Application present.

#### 5.1.4.2.8 Forced

If TRUE, this attribute indicates that the Processing Unit has at least one variable forced. When a variable is forced, the application program will use the value specified by the PADT instead of the value generated by the normal program execution.

#### 5.1.4.2.9 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.1.4.3 Power Supply Subsystem

The PC can supply status about any of the power supply subsystems; see figure 4 for the assumed configuration of a PC Power Supply. The requirements on power supplies of PC systems and their behavior are described in IEC 61131-1 and IEC 61131-2.

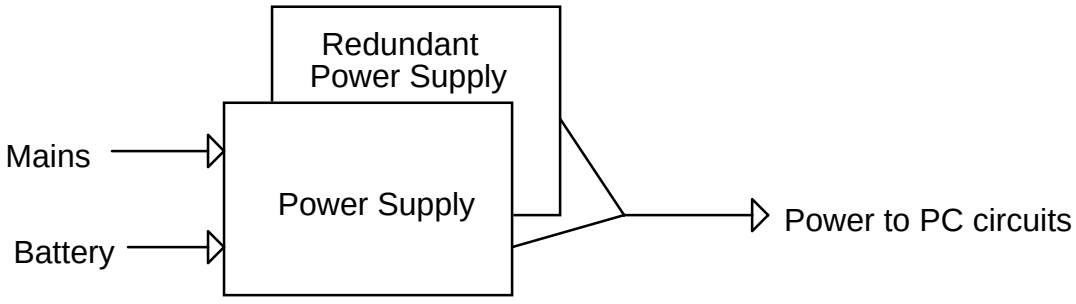



**Figure 4 – Programmable Controller power supply**

L'objet Power Supply définit les statuts du sous-système d'alimentation électrique.

```
Object: Power Supply
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: In Use (TRUE, FALSE)
  Attribute: Mains Operating (TRUE, FALSE)
  Attribute: Mains Low (TRUE, FALSE)
  Attribute: Battery Operating (TRUE, FALSE)
  Attribute: Battery Low (TRUE, FALSE)
  Attribute: Protection Tripped (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.3.1 Subsystem Name

Cet attribut identifie le sous-système.

#### 5.1.4.3.2 Health

Cet attribut décrit l'état de santé de l'alimentation.

a) GOOD – indique qu'aucun problème n'a été détecté dans l'alimentation qui l'empêcherait d'être opérationnelle durant un temps indéterminé.

b) WARNING – indique qu'un problème a été détecté dans l'alimentation qui risquerait de la rendre non opérationnelle prochainement.

c) BAD – indique que l'alimentation est hors service.

#### 5.1.4.3.3 In Use

Si cet attribut est VRAI, il indique que l'alimentation est en marche, c'est-à-dire qu'elle fournit de l'énergie électrique au PC.

#### 5.1.4.3.4 Mains Operating

Si cet attribut est VRAI, il indique que le secteur fournit de l'énergie électrique dans les tolérances spécifiées.

#### 5.1.4.3.5 Mains Low

Si cet attribut est VRAI, il indique que le secteur ne fournit pas de l'énergie électrique dans les tolérances spécifiées.

#### 5.1.4.3.6 Battery Operating

Si cet attribut est VRAI, il indique que la batterie fournit de l'énergie électrique dans les tolérances spécifiées.

#### 5.1.4.3.7 Battery Low

Si cet attribut est VRAI, il indique que la batterie ne fournit pas de l'énergie électrique dans les tolérances spécifiées.

#### 5.1.4.3.8 Protection Tripped

Si cet attribut est VRAI, il indique que le dispositif de protection de l'alimentation électrique a privé le PC d'une partie de l'alimentation.

The object Power Supply models the status information of a power supply subsystem.

```
Object: Power Supply
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: In Use (TRUE, FALSE)
  Attribute: Mains Operating (TRUE, FALSE)
  Attribute: Mains Low (TRUE, FALSE)
  Attribute: Battery Operating (TRUE, FALSE)
  Attribute: Battery Low (TRUE, FALSE)
  Attribute: Protection Tripped (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.3.1 Subsystem Name

The Subsystem Name identifies the subsystem.

#### 5.1.4.3.2 Health

This attribute identifies the health of the Power Supply.

a) GOOD – indicates that there have been no problems detected in the power supply to prevent it from remaining operable for an indefinite time.

b) WARNING – indicates that a problem has been detected in the power supply which may cause to become inoperable in a limited time.

c) BAD – indicates that the power supply is not operable.

#### 5.1.4.3.3 In Use

If TRUE, this attribute indicates that the power supply subsystem is in use, i.e. it supplies power to the PC.

#### 5.1.4.3.4 Mains Operating

If TRUE, this attribute indicates that the mains are supplying power within the range specified for the power supply.

#### 5.1.4.3.5 Mains Low

If TRUE, this attribute indicates that the mains are not supplying power within the range specified for the power supply.

#### 5.1.4.3.6 Battery Operating

If TRUE, this attribute indicates that the battery is supplying power within the range specified for the battery.

#### 5.1.4.3.7 Battery Low

If TRUE, this attribute indicates that the battery is not supplying power within the range specified for the battery.

#### 5.1.4.3.8 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.

#### 5.1.4.3.9 List Of Implementer Specified State

Cet attribut indique les valeurs d'état qui doivent être précisées par le concepteur dans la PICS.

### 5.1.4.4 Sous-système de mémorisation

L'objet Memory définit les statuts du sous-système de mémorisation.

```
Object: Memory
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: Protected (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.4.1 Subsystem Name

L'attribut Subsystem Name identifie le sous-système.

#### 5.1.4.4.2 Health

Cet attribut définit l'état de santé du sous-système de mémorisation.

a) GOOD – indique qu'aucune anomalie n'a été trouvée dans la mémoire associée à ce sous-système.

b) WARNING – indique qu'au moins une anomalie réparable a été détectée mais qu'aucune anomalie non réparable n'a été trouvée.

c) BAD – indique qu'au moins une anomalie non réparable a été trouvée.

#### 5.1.4.4.3 Protected

SI cet attribut est VRAI, il indique que la mémoire du sous-système de mémorisation considéré a été protégée, c'est-à-dire qu'elle n'est pas modifiable. Cela indique généralement que le programme d'application résidant dans ce sous-système de mémoire ne peut être modifié.

NOTE – Cet attribut décrit un état logique et non des caractéristiques physiques du sous-système. Si quelques parties de la mémoire seulement sont protégées, elles peuvent être considérées comme autant de sous-systèmes.

#### 5.1.4.4.4 List Of Implementer Specified State

Cet attribut indique les valeurs d'état qui doivent être précisées par le concepteur dans la PICS.

### 5.1.4.5 Sous-système de communication

L'objet Communication Subsystem définit les statuts d'un sous-système de communication.

```
Object: Communication Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: In Use (TRUE, FALSE)
  Attribute: Local Error (TRUE, FALSE)
  Attribute: Remote Error (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.5.1 Subsystem Name

L'attribut Subsystem Name identifie le sous-système.

#### 5.1.4.3.9 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.1.4.4 Memory Subsystem

The object Memory models the status information of a memory subsystem.

```
Object: Memory
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: Protected (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.4.1 Subsystem Name

The Subsystem Name identifies the subsystem.

#### 5.1.4.4.2 Health

This attribute identifies the health of the memory subsystem.

a) GOOD – no errors have been found in the memory associated with this subsystem.

b) WARNING – at least one correctable error has been detected and no uncorrectable errors have been detected.

c) BAD – at least one uncorrectable error has been detected.

#### 5.1.4.4.3 Protected

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.

NOTE – 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 can be reported as multiple subsystems.

#### 5.1.4.4.4 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.1.4.5 Communication Subsystem

The object Communication Subsystem models the status information of a communication subsystem.

```
Object: Communication Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: In Use (TRUE, FALSE)
  Attribute: Local Error (TRUE, FALSE)
  Attribute: Remote Error (TRUE, FALSE)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.5.1 Subsystem Name

The Subsystem Name identifies the subsystem.

#### 5.1.4.5.2 Health

Cet attribut définit l'état de santé du sous-système de communication.

a) GOOD – indique soit qu'aucune anomalie n'a été trouvée, soit qu'un nombre acceptable d'anomalies réparables a été détecté.

b) WARNING – indique qu'un nombre trop important d'anomalies réparables a été trouvé.

c) BAD – indique que le sous-système de communication n'est pas en mesure de communiquer avec toutes les équipements comme prévu.

La définition du «nombre acceptable» d'anomalies réparables doit être fournie par le concepteur.

#### 5.1.4.5.3 In Use

Si cet attribut est VRAI, il indique que le sous-système de communication est en fonctionnement. Dans le cas d'une interface de communication MMS, il indique qu'au moins une association d'application est établie. Autrement, le réalisateur doit fournir la sémantique de cet attribut.

#### 5.1.4.5.4 Local Error

Si cet attribut est VRAI, il indique que certaines erreurs internes empêchent le fonctionnement du sous-système de communication.

#### 5.1.4.5.5 Remote Error

Si cet attribut est VRAI, il indique que certaines erreurs trouvées sur d'autres équipements communiquant avec le sous-système empêchent le fonctionnement de ce sous-système.

NOTE – Le sous-système de communication renvoyant son état peut ne pas être en mesure de renvoyer son état de défectuosité tel qu'il est définit dans ce paragraphe. Cependant, une configuration PC pouvant compter plusieurs sous-systèmes de communication indépendants, les valeurs d'état de tous les sous-systèmes peuvent être définies ensemble dans l'objet Sous-système de communication.

Il est convenu que les informations spécifiques du concepteur fourniront également d'autres informations sur chaque interface particulière. Les interfaces de réseau ISO apportent également des informations via les fonctions de gestion de réseau.

#### 5.1.4.5.6 List Of Implementer Specified State

Cet attribut indique les valeurs d'état qui doivent être précisées par le concepteur dans la PICS.

### 5.1.4.6 Sous-systèmes spécifiques au concepteur

D'autres sous-systèmes au sein d'un système PC peuvent être définis comme sous-systèmes spécifiques au concepteur. Voici, en exemple, de tels sous-systèmes:

a) contrôleur de PID;

b) contrôleur de mouvement;

c) autres processeurs annexes.

L'objet Implementer Specific Subsystem définit les statuts d'un sous-système spécifique au concepteur.

```
Object: Implementer Specific Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.5.2 Health

This attribute identifies the health of the communication subsystem.

a) GOOD – indicates either no errors or an acceptable number of recoverable errors has occurred.

b) WARNING – indicates that more than an acceptable number of recoverable errors has occurred.

c) BAD – indicates that the communication subsystem is not able to communicate with all devices as intended.

The definition of "an acceptable number" of recoverable errors shall be provided by the implementer.

#### 5.1.4.5.3 In use

If TRUE, this attribute indicates that the communication subsystem is currently operating. In the case of a MMS communication interface this means that at least one application association is established. Otherwise the implementer shall define the semantic of this attribute.

#### 5.1.4.5.4 Local error

If TRUE, this attribute indicates that there are some errors, internal to the communication subsystem, that inhibit operation.

#### 5.1.4.5.5 Remote error

If TRUE, this attribute indicates that there are some errors, at devices being communicated with, that inhibit operation.

NOTE – The communication subsystem reporting its state may not be able to report its own bad state in the way defined in this subclause. But in a PC system may operate several independent communication subsystems, the states of all of them may be modeled with the object Communication Subsystem.

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.

#### 5.1.4.5.6 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.1.4.6 Implementer Specific Subsystems

Other subsystems of a PC system can be modeled as implementer specific subsystems. Some examples of these subsystems are the following:

a) PID controller;

b) Motion controller;

c) other auxiliary processors.

The object Implementer Specific Subsystem models the status information of an implementer specific subsystem.

```
Object: Implementer Specific Subsystem
  Key Attribute: Subsystem Name
  Attribute: Health (GOOD, WARNING, BAD)
  Attribute: List Of Implementer Specified State
```

#### 5.1.4.6.1 Subsystem Name

L'attribut Subsystem Name identifie le sous-système.

#### 5.1.4.6.2 Health

Cet attribut décrit l'état de santé du sous-système spécifique au concepteur.

a) GOOD – indique qu'aucune anomalie n'a été détectée dans ce sous-système.

b) WARNING – indique qu'une anomalie mineure a été détectée dans ce sous-système.

c) BAD – indique qu'une anomalie grave a été détectée dans ce sous-système.

La définition de «anomalie mineure» et «anomalie grave» doit être fournie par le concepteur.

#### 5.1.4.6.3 List Of Implementer Specified State

Cet attribut indique les valeurs d'état qui doivent être précisées par le concepteur dans la PICS.

### 5.2 Fonctions spécifiques à l'application

Ce paragraphe décrit les fonctions qu'un PC met à la disposition des clients dans un système d'automatisation, en utilisant un système de communication, tel qu'illustré à la figure 1.

### 5.2.1 Généralités

Le PC fournit au système d'automatisation les fonctions suivantes:

a) * vérification des équipements;

b) * acquisition de données;

c) * commande;

d) contrôle-commande et gestion des programmes;

e) synchronisation des applications utilisateur;

f) envoi de message d'alarme;

g) modification de recettes;

h) * gestion des connexions.

Chacune de ces fonctions est expliquée dans ce paragraphe. Les fonctions précédées d'une étoile (*) peuvent être appelées par le PC à partir d'autres PC. Certaines fonctions peuvent ne pas être disponibles sur certains PC.

Certaines applications peuvent traiter simultanément plusieurs catégories d'applications décrites ci-dessus, comme par exemple la commande de surpervision et l'acquisition de données.

Bien qu'ils soient généralement fournis par les PC, les éléments suivants n'entrent pas dans le cadre de cette norme d'accompagnement:

i) interface de l'opérateur;

j) programmation, mise au point et vérification des programmes d'application.

Les PC peuvent utiliser les équipements d'Interface Opérateur. Ces équipements sont utilisés par un opérateur pour surveiller ou modifier le procédé contrôlé, ou encore les deux à la fois. Ils peuvent aussi permettre à un système Client de communiquer avec l'opérateur.

On appelle Interface Directe Opérateur lorsqu'un Client peut communiquer avec l'interface opérateur via le système de communication sans l'intervention du programme d'application.

#### 5.1.4.6.1 Subsystem Name

The Subsystem Name identifies the subsystem.

#### 5.1.4.6.2 Health

This attribute identifies the health of the implementer specific subsystem.

a) GOOD – indicates that there have been no errors detected in this subsystem.
b) WARNING – indicates that a minor fault has been detected in this subsystem.
c) BAD – indicates that a major fault has been detected in this subsystem.

The definition of "major fault" and "minor fault" shall be provided by the implementer.

#### 5.1.4.6.3 List Of Implementer Specified State

This attribute indicates state information which shall be specified by the implementer in the PICS.

### 5.2 Application-Specific Functions

The remainder of this subclause describes the functions which a PC provides to clients within an automation system, using the communication system, as illustrated in figure 1.

#### 5.2.1 Overview

The PC provides the following functions to the automation system:

a) * device verification;
b) * data acquisition;
c) * control;
d) program management and control;
e) synchronization between user applications;
f) alarming;
g) recipe manipulation;
h) * connection management.

Each of these is treated separately in the remainder of this subclause. The items preceded by an "*" may also be requested by a PC from other PCs. Not all functions are available in all PCs.

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 of the scope of this companion standard:

i) operator interface;
j) programming, debugging, 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.

La programmation consiste à créer un programme d'application du PC par l'écriture séquentielle d'instructions ou de blocs de fonctions. La mise au point des programmes d'application consiste à trouver des erreurs («bugs») dans un programme d'application existant et à les corriger. La vérification du programme consiste à tester le programme d'application du PC pour s'assurer qu'il exécute bien la ou les fonctions qu'il est censé exécuter dans l'environnement du procédé.

### 5.2.2 Vérification de l'équipement

Cette fonction permet à d'autres équipements de déterminer si le PC est capable de réaliser la fonction attendue dans le système automatisé. Les trois pôles d'intérêt dans la vérification de l'équipement sont les valeurs d'état, la détection des anomalies et la localisation des anomalies.

Un PC peut fournir des statuts, à savoir l'état de santé et les valeurs d'état, à tout équipement le demandant, ou établir sans avoir été sollicité un rapport d'état, en utilisant les fonctions dont dispose le sous-système de communication. Le rapport non sollicité est réalisé lorsque les attributs de santé ou d'état sur le récapitulatif d'états du PC sont modifiés. Se reporter à 5.1.4 pour connaître la définition de l'état de santé et des valeurs d'état de la configuration du PC ou de ses sous-systèmes.

### 5.2.3 Acquisition de données

Les données contenues dans un PC peuvent être présentées sous la forme de variables. Ces données peuvent provenir de sources différentes et peuvent avoir des significations très différentes. Elles peuvent être obtenues par le Client par différentes méthodes:

a) interrogée – le Client lit la valeur d'une ou plusieurs variables à l'instant où il le désire et sous les conditions qu'il détermine. L'accès aux variables peut être contrôlé par le PC. Seules les variables sélectionnées sont accessibles sur le réseau;

b) programmée – le PC met les données à la disposition du Client à l'instant et dans les conditions d'utilisation déterminés par le programme d'application du PC;

c) configurée – le Client peut configurer le sous-système de communications vers le PC pour qu'un transfert de données vers le Client ait lieu.

### 5.2.4 Commande

Deux types de commandes doivent être admis par les PC: la commande paramétrique et la commande verrouillée.

La commande est paramétrique lorsque le fonctionnement du PC se fait en attribuant des valeurs aux variables résidant sur le PC. Le changement de fonctionnement est déterminé par le programme d'application ou par d'autres mécanismes locaux.

L'accès aux variables peut être contrôlé par le PC qui détient les variables. Seules les variables sélectionnées sont accessibles sur le réseau.

La commande est verrouillée lorsque le Client demande au Serveur d'exécuter une opération de l'application et de l'informer du résultat de l'opération. Il y a deux aspects à ce service: la synchronisation du Client et du Serveur et l'échange mutuel de données.

Dans une commande verrouillée, l'échange de données a lieu à des points de synchronisation dans le programme d'application. Ce service peut être utilisé pour avoir l'effet d'un appel distant de procédure d'un programme d'application vers l'autre. Le schéma suivant illustre cette synchronisation:

Programming is the process of creating a PC application program on an instruction by instruction or a function block by function block basis. Debugging 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.

### 5.2.2 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. There are three topics of interest in the realm of Device Verification, namely state information, fault detection, and fault isolation.

A PC can provide status, which includes health and state information to a requesting device or initiate an unsolicited status report using services provided by the communication subsystem. The unsolicited status is provided when the health or state of the PC Summary Status changes. See 5.1.4 for the definition of health and state information of a PC system and of its subsystems.

### 5.2.3 Data Acquisition

Data contained in a PC 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 Client through one of several methods:

a) 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;

b) programmed – the data is provided by the PC to the Client at a time or condition determined by the PC application program;

c) configured – the communications subsystem to the PC can be configured by a Client to initiate a data transfer to the Client.

### 5.2.4 Control

Two methods of control shall be supported by PCs: 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 may be controlled by the PC which holds the variables. Only selected variables are accessible over the network.

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 following timeline illustrates this.

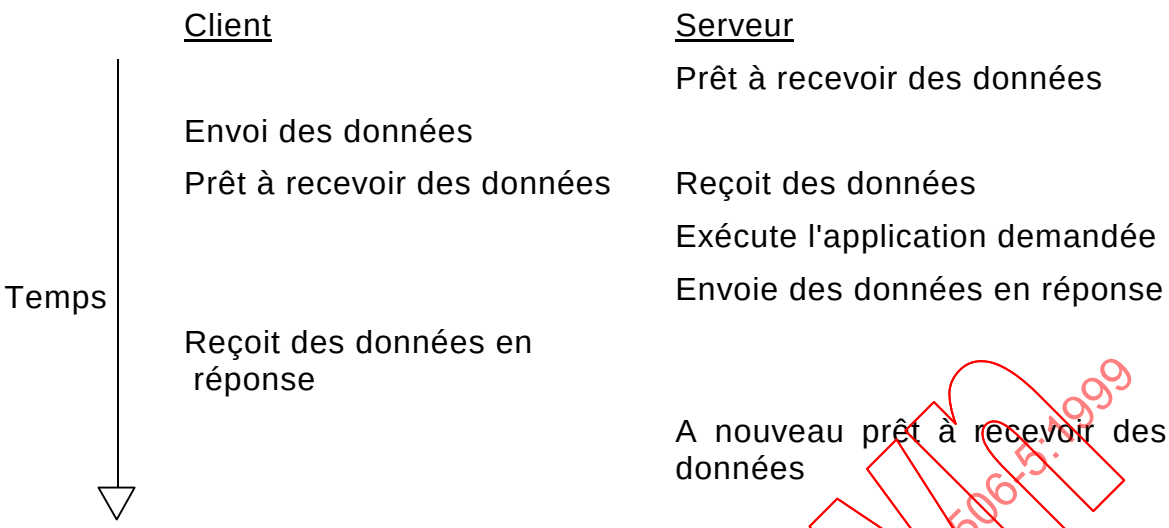



**Figure 5 – Synchronisation par commande verrouillée**

### 5.2.5 Synchronisation des applications utilisateur

Les applications utilisateur peuvent nécessiter un service de synchronisation. Une application utilisateur peut, par exemple, verrouiller l'exécution d'une autre application jusqu'à l'exécution complète d'un algorithme, en utilisant un sémaphore. Le service de verrouillage est fourni par une entité communicante dans l'environnement distribué. Les applications coopérantes choisissent un nom de sémaphore ainsi que les règles de son utilisation sur la base d'un accord mutuel: ces règles ne peuvent en aucun cas être forcées.

Du point de vue de la réalisation, l'interface réseau gère un sémaphore qui peut être invisible au reste du PC.

### 5.2.6 Envoi de message d'alarme

Le PC peut être en mesure d'envoyer des messages d'alarme à un Client quand une condition prédéfinie survient. Le Client peut émettre un accusé de réception de ces alarmes vers le PC. Cette fonction se distingue de l'acquisition normale de données dans le sens où l'état d'alarme est gardé en mémoire par le PC tant que le Client n'en accuse pas réception. Un récapitulatif des alarmes reçues, sans accusé de réception, peut être généré par le PC sur la demande du Client.

### 5.2.7 Contrôle-Commande et gestion des programmes

La gestion des programmes et le Contrôle-Commande englobe plusieurs fonctions d'application telles que

a) le contrôle d'exécution du programme et le contrôle des entrées-sorties associées;

b) le transfert de son contenu.

#### 5.2.7.1 Exécution et contrôle des entrées-sorties

L'exécution d'un programme d'application PC et le contrôle des entrées-sorties sont gérés par la fonction d'exécution du programme d'application (voir 5.1.2). Les programmes d'application PC peuvent être démarrés et arrêtés. Les programmes d'application peuvent être lancés soit à partir du début soit à partir de l'endroit où ils ont été arrêtés.

Un programme d'application PC peut être constitué d'un certain nombre de tâches. Elles peuvent être indépendantes ou être sous contrôle d'une autre tâche. Lorsqu'une tâche est lancée ou arrêtée, toutes celles qu'elle contrôle sont aussi lancées ou arrêtées.

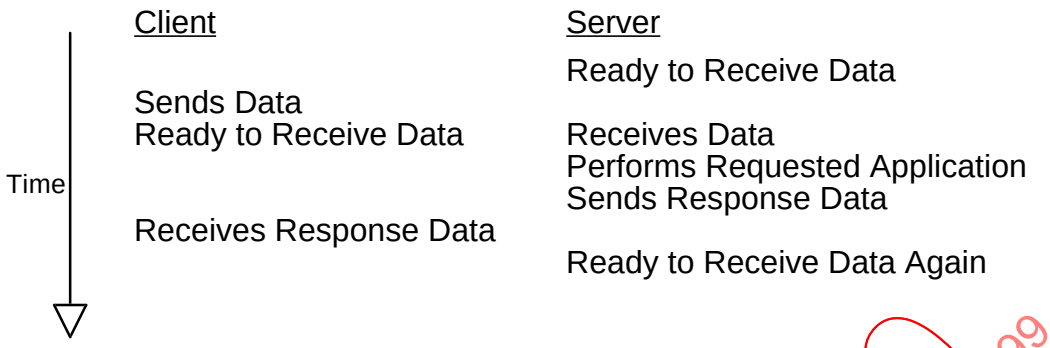



**Figure 5 – Interlocked control timeline**

### 5.2.5 Synchronization between User Applications

User applications may need a synchronization service. For example, a user application may lock the execution of another application until completion of an algorithm by taking control of a semaphore. The lock service is provided by a communicating entity in the distributed environment. The cooperating applications choose a semaphore name and a set of rules for its use by a gentleman's agreement: these rules cannot be enforced.

From an implementation point of view, the network interface manages a semaphore which may remain unknown to the rest of the PC.

### 5.2.6 Alarming

The PC can have the ability to signal alarm messages to a Client when a predetermined condition occurs. The Client may indicate an acknowledgment 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.

### 5.2.7 Program Management and Control

The area of Program Management and Control involves several application functions, including the follfowing:

a) controlling its execution and associated I/O;

b) transferring its contents.

#### 5.2.7.1 Execution and I/O Control

Execution of a PC application program is managed by the application program Execution function; see 5.1.2. 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.

A PC application program can consist of a number of tasks. These tasks may be independent or under the control of another task. When a task is started or stopped, all tasks under control of that task are also started or stopped.

La fonction d'interface vers les actionneurs (sorties) associée à un programme d'application en cours d'exécution peut utiliser soit les valeurs fournies par le programme d'application, soit les valeurs maintenues dans un état connu. Cet état de l'interface est spécifié au moment où l'état du programme d'application est modifié. Les sorties peuvent être modifiées soit pour avoir des valeurs d'état définies par le concepteur, soit pour conserver les valeurs de sortie en cours, soit pour avoir des valeurs remises à zéro ou pour positionner certaines sorties à des valeurs définies par l'utilisateur (active ou non active, celles qui n'ont pas été spécifiées conservant leur dernier état). Ces modifications s'effectuent grâce à un mécanisme défini par le concepteur (par exemple tableaux, procédures PC, etc.).

La fonction d'interface vers les capteurs (entrées) associée à un programme d'application en cours d'exécution peut soit fournir les données réelles provenant des capteurs, soit continuer d'utiliser les valeurs antérieures précédemment fournies. L'état des entrées est spécifié au moment où l'état du programme d'application est modifié.

NOTE – Il convient que le réseau ne dépende pas, par exemple, du remplacement d'un interrupteur d'arrêt d'urgence. Il convient de suivre des pratiques de sécurité normales.

#### 5.2.7.2 Transfert de programme d'application

Le transfert d'un programme d'application est effectué à l'aide des fonctions de stockage des programmes d'application et des données d'un PC (voir 5.1.2). Le transfert d'un programme d'application permet au Client de télésauvegarder le contenu de la mémoire programmable pour l'archivage ou la vérification ou de télécharger ce contenu pour restaurer le PC dans un état connu. Cette fonction permet également de figer le PC dans un état de sécurité avant que le contenu de sa mémoire programmable ne soit modifié, et de le redémarrer en toute sécurité une fois le transfert du programme d'application terminé.

Le lancement d'un transfert de programme est généralement effectué par un équipement autre qu'un PC. Les fonctions comprennent ce qui suit:

a) la télésauvegarde pour archivage;

b) la télésauvegarde pour vérification;

c) le téléchargement pour la restauration du système à un état antérieur connu et correct; et

d) le téléchargement d'un système développé hors réseau.

Il convient de prendre en considération les points suivants:

– la présente partie de l'ISO/CEI 9506 définit un moyen de réaliser des téléchargements et des télésauvegardes sur un PC entier, un sous-système du PC et une partie d'un sous-système;

– seuls les éléments arrêtés d'un programme d'application peuvent être téléchargés;

– le système ou la partie du système en cours de téléchargement n'est pas disponible à d'autres utilisations tant que le téléchargement n'est pas terminé;

– le réalisateur devra spécifier les opérations que d'autres Clients pourront effectuer sur un PC lorsqu'un Client le téléchargera.

#### 5.2.8 Modification de recette

La modification de recette peut nécessiter des services variés depuis la seule mise à jour de quelques données jusqu'au téléchargement d'importantes parties du programme d'application.

Pour les produits qui nécessitent seulement la mise à jour de peu de données, la commande paramétrique ou verrouillée peut suffire. En revanche, si la mise à jour concerne un grand nombre de données, le téléchargement des données du PC peut être utilisé.

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 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.

NOTE – The network should not be depended on as a replacement for a hardwired emergency stop switch. Normal safety practices should be followed.

#### 5.2.7.2 Application Program Transfer

Application program transfer is done using the application program Storage and Data Storage Functions of a PC (see 5.1.2). Application program transfer allows the Client to upload the contents of the programmable memory 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 the following:

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.

Some considerations include the following:

– this part of IEC 9506 defines a means to perform uploads and downloads on the whole PC, a subsystem of the PC and a portion of a subsystem;

– only stopped portions of the PC application program can be downloaded;

– the whole or the portion of the system being downloaded is not 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.

#### 5.2.8 Recipe Manipulation

Recipe Manipulation may vary in the services needed from only updating some data up to downloading large portions of the application program.

For those products which only require a small amount of data to change, the parametric or interlocked control service can be used. If a large amount of data needs to be changed, then a download of the data portion of the PC can be accommodated.

Pour les produits nécessitant des modifications procédurales, la télésauvegarde et le téléchargement de certaines parties du programme d'application du PC s'avèrent nécessaires ainsi que cela est décrit en 5.2.7.

Se reporter à l'annexe A pour examiner des exemples de modifications de recette.

### 5.2.9 Gestion des connexions

La gestion des connexions fournit le moyen d'établir, de maintenir et de clore des connexions de communications avec un partenaire distant.

Les connexions sont commandées de façon explicite par le programme d'application ou sont accordées par le sous-système de communications si elles sont utiles et pendant le laps de temps nécessaire.

Si plusieurs requêtes doivent être faites au même équipement, elles peuvent toutes utiliser la même connexion.

## 6 Correspondance du contexte spécifique à l'application avec les objets MMS

Le PC fonctionne selon le contexte défini dans l'ISO/CEI 9506-1 (MMS). Les objets PC définis sur un PC ou par un programme d'application PC (voir 5.1) correspondent aux objets MMS. Cette correspondance est décrite dans les paragraphes suivants de l'article 6. La correspondance des objets PC et MMS sur un véritable Automate Programmable est décrite par le concepteur et n'entre pas dans le cadre de la présente norme.

### 6.1 Correspondance du modèle spécifique à l'application avec l'objet VMD

Chaque PC offrant des communications MMS est mis en correspondance avec un VMD au moins.

Lorsqu'une configuration comprend un ou plusieurs Automates Programmables et un sous-système de communication MMS qui leur est commun, elle peut être mise en correspondance avec un VMD (voir figure 6). Il est également possible de disposer de plus d'un VMD avec un Automate Programmable et un sous-système de communication, comme le montre la figure 7. Un PC peut avoir par exemple plus d'une PU, chacune d'entre elles étant mise en correspondance avec un VMD.

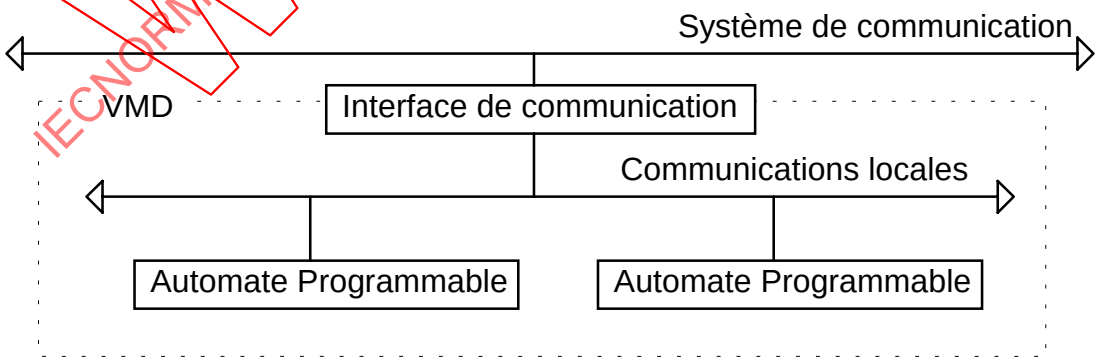



**Figure 6 – Deux PC mis en correspondance avec un VMD**

For those products which require procedural changes, uploading and downloading of portions of the PC application program are required, as described in 5.2.7.

See annex A for an example of Recipe Manipulation.

### 5.2.9 Connection Management

The Connection Management provides the means to install, to maintain and close communications connections to a remote communication partner.

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

If multiple requests are initiated to operate with a device, they may all work over the same connection.

## 6 Application-Specific Context Mapping

The PC is operating in the context of ISO/IEC 9506-1 (MMS). The PC objects defined in a PC or by a PC application program (see 5.1) are mapped onto MMS objects. This mapping is described in the following paragraphs of clause 6. The mapping of the PC objects and the MMS objects onto a real Programmable Controller is specified by the implementer and is beyond the scope of this standard.

### 6.1 Mapping of the Application-Specific Model to the VMD Object

Each PC that offers MMS Communications is mapped onto at least one VMD.

It is possible for a configuration consisting of one or more real Programmable Controllers with one common MMS communication subsystem to be mapped to one VMD, as in figure 6. It is also possible to implement more than one VMD with one real Programmable Controller and one communication subsystem, as in figure 7. For example, a PC may have more than one PU, each of which may be mapped onto a VMD.

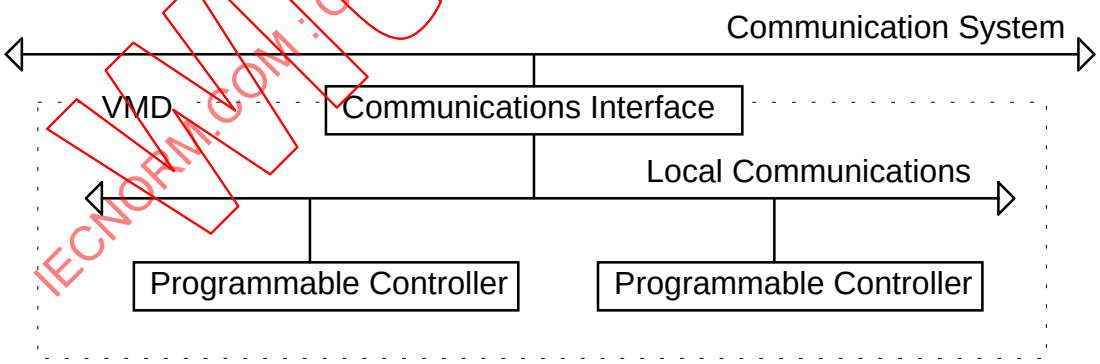



**Figure 6 – Two PCs mapped onto one VMD**

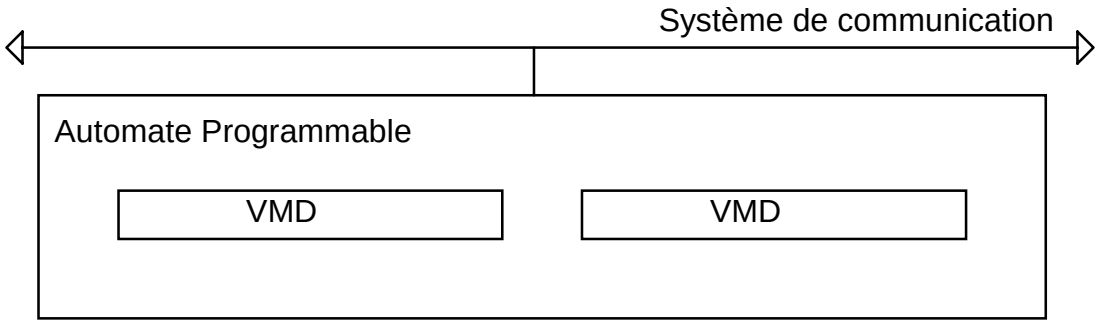



**Figure 7 – Un PC mis en correspondance avec deux VMD**

Aucune extension au modèle de VMD de MMS n'est apportée par la présente norme d'accompagnement.

### 6.2 Définition des objets spécifiques à l'application correspondant à des domaines

La fonction de stockage du programme d'application et la fonction de stockage des données du PC (voir figure 2) correspondent à un domaine au moins. Aucune extension au modèle de Domaine de MMS n'est apportée par la présente norme d'accompagnement.

### 6.3 Définition des objets spécifiques à l'application correspondant à des invocations de programme

La fonction d'exécution d'un programme d'application d'un PC (défini en 5.1.2) correspond au moins à une invocation de programme.

```
Object: Program Invocation
  All MMS defined Attributes
  Attribute: Independent (TRUE, FALSE)
  constraint: Independent = (FALSE)
        Attribute: Program Invocation Reference
  Attribute: I/O State (controlled, holdOutputs, holdCurrentState,
                        implementerState, zeroOutputs, userSpecified)
```

#### 6.3.1 Independent

Cet attribut indique si l'invocation de programme est indépendante (VRAI) ou contrôlée (FAUX) par l'invocation de programme référencée par l'attribut Program Invocation Reference, qui est l'invocation de programme maître. Toute modification de l'état de l'invocation de programme maître par les fonctions MMS Start, Resume, Stop, Reset et Kill entraînera une tentative de modification identique de l'état de l'invocation de programme contrôlée (voir 5.2.7.1).

#### 6.3.2 Program Invocation Reference

Cet attribut précise quelle est l'invocation de programme commandant cette invocation de programme. Le concepteur doit préciser s'il est autorisé de référencer une invocation de programme dont l'attribut Independent est fixé à FAUX (autrement dit, si l'imbrication de références d'invocations de programmes est autorisée).

#### 6.3.3 I/O State

L'attribut I/O State décrit l'état de la fonction d'interface pour les capteurs et les actionneurs (définie en 5.1.2) commandée par cette invocation de programme. Les valeurs de cet attribut et leurs définitions sont décrites ci-dessous.

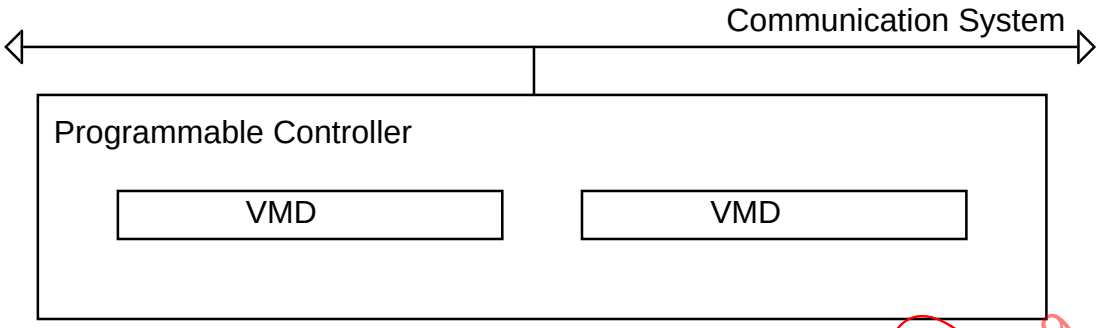



**Figure 7 – One PC mapped onto two VMDs**

The VMD model of MMS is not extended by this companion standard.

### 6.2 Definition of Application-Specific Objects which map to Domains

The application program Storage Function and the Data Storage Function of the PC (see figure 2) is mapped onto at least one Domain. The Domain model of MMS is not extended by this companion standard.

### 6.3 Definition of Application-Specific Objects which map to Program Invocations

The application program Execution Function of the PC (defined in 5.1.2) is mapped onto at least one Program Invocation.

```
Object: Program Invocation
  All MMS defined Attributes
  Attribute: Independent (TRUE, FALSE)
  Constraint: Independent = (FALSE)
            Attribute: Program Invocation Reference
  Attribute: I/O State (controlled, holdOutputs, holdCurrentState,
            implementerState, zeroOutputs, userSpecified)
```

#### 6.3.1 Independent

This attribute indicates whether this Program Invocation is independent (TRUE) or controlled (FALSE) by the Program Invocation referenced in the attribute Program Invocation Reference, the controlling Program Invocation. Changes to the state of the controlling Program Invocation as a result of the Start, Resume, Stop, Reset, and Kill MMS services will also attempt to cause the same changes to the state of the controlled Program Invocation. (See 5.2.7.1.)

#### 6.3.2 Program Invocation Reference

This attribute specifies the Program Invocation by which this Program Invocation is controlled. The implementer shall specify if it is permitted to reference a Program Invocation which has the Independent attribute set to FALSE (that is, if nesting the Program Invocation References is permitted).

#### 6.3.3 I/O State

The I/O State attribute contains the state of the Interface Function to Sensors and Actuators (defined in 5.1.2) controlled by this Program Invocation. The values of this attribute and the meaning of each value are as follows.

0 – controlled – Les actionneurs sont commandés par l'invocation de programme. Les entrées sont fournies à l'invocation de programme par les capteurs.

1 – holdOutputs – Les actionneurs ne sont pas commandés par l'invocation de programme, leur état courant est maintenu. Les entrées sont fournies à l'invocation de programme par les capteurs.

2 – holdCurrentState – Les actionneurs ne sont pas commandés par l'invocation de programme, leur état courant est maintenu. Les entrées ne sont pas fournies à l'invocation de programme par les capteurs et elles conservent leur état courant.

3 – implementerState – Les actionneurs ne sont pas commandés par l'invocation de programme; ils conservent l'état courant qui a été précisé par le concepteur. Le programme d'application n'est pas en cours, c'est pourquoi l'état de l'interface capteurs n'est pas spécifié.

4 – zeroOutputs – Les actionneurs ne sont pas commandés par l'invocation de programme et ils conservent l'état zéro. Le programme d'application n'est pas en cours, c'est pourquoi l'état de l'interface capteurs n'est pas spécifié.

5 – userSpecified – Les actionneurs ne sont pas commandés par l'invocation de programme et ils conservent l'état courant qui a été précisé par l'utilisateur. Le programme d'application n'est pas en cours, c'est pourquoi l'état de l'interface capteurs n'est pas spécifié.

## 6.4 Définition des objets spécifiques à l'application correspondant à d'autres objets abstraits MMS

### 6.4.1 Définition des objets spécifiques à l'application correspondant à des variables

Les portions accessibles des données gérées par les fonctions de stockage de données du PC (définies en 5.1.2) correspondent à des variables MMS.

Les statuts du PC et de ses sous-systèmes (définis en 5.1.4) correspondent à des variables MMS. Les attributs HEALTH et STATE sont fournis en une chaîne binaire unique. A chacune des trois valeurs HEALTH correspond un bit. Un et seulement un des bits HEALTH doit être un 1.

Les variables PC directement représentées (voir la CEI 61131-3) correspondent à des objets variables non nommées MMS d'un RC. La syntaxe de l'adresse symbolique du paramètre d'adresse de ces variables non nommées doit être précisée par le concepteur dans la PICS. Il doit également préciser les moyens de restreindre l'accès à ces variables, s'il en existe.

### 6.4.2 Définition des objets spécifiques à l'application correspondant à des objets de Gestion des événements

La fonction d'alarme du programme d'application du PC correspond aux objets de gestion d'événements MMS. Les modèles de gestion des événements de MMS ne sont pas développés dans la présente norme d'accompagnement.

### 6.4.3 Définition des objets spécifiques à l'application correspondant à des objets Échange de données

Une fonction de commande verrouillée correspond à un objet Échange de données. Aucune extension au modèle de d'échange de données de MMS n'est apportée par la présente norme d'accompagnement.

NOTE – L'objet Échange de données est décrit dans l'amendement 1 de l'ISO/CEI 9506-1.

0 – controlled – Actuators are being controlled by the Program Invocation. Inputs are being supplied to the Program Invocation by the Sensors.

1 – holdOutputs – Actuators are not being controlled by the Program Invocation, they are held in the current state. Inputs are being supplied to the Program Invocation by the Sensors.

2 – holdCurrentState – Actuators are not being controlled by the Program Invocation, they are held in the current state. Inputs are not being supplied to the Program Invocation by the Sensors, they are held in the current state.

3 – implementerState – Actuators are not being controlled by the Program Invocation, 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.

4 – zeroOutputs – Actuators are not being controlled by the Program Invocation, they are held in the zero state. The application program is not running, therefore the state of the Sensor Interface is not specified.

5 – userSpecified – Actuators are not being controlled by the Program Invocation, 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.

### 6.4 Definition of Application-Specific Objects which map to Other MMS Abstract Objects

#### 6.4.1 Definition of Application-Specific Objects which map to Variables

Accessible portions of the data managed by the Data Storage Functions (defined in 5.1.2) of the PC are mapped to MMS Variables.

The status of the PC and its subsystems (defined in 5.1.4) are mapped onto MMS Variables. The HEALTH and STATE attributes are provided in one bitstring. There is a bit for each of the three HEALTH values. Exactly one of the HEALTH bits shall be a one.

The directly represented PC variables (see IEC 61131-3) are mapped onto MMS Unnamed Variable objects of a PC. The syntax of the symbolic address of the address parameter of these Unnamed Variables is to be specified by the implementer in the PICS. The implementer shall also specify the means to restrict access to these variables, if any exists.

#### 6.4.2 Definition of Application-Specific Objects which map to the Event Management Objects

The PC application program Alarm function is mapped onto MMS Event Management Objects. The Event Management models of MMS are not extended by this companion standard.

#### 6.4.3 Definition of Application-Specific Objects which map to Data Exchange Objects

One Interlocked Control function is mapped onto one Data Exchange object. The Data Exchange model of MMS is not extended by this companion standard.

NOTE – The Data Exchange object is specified in amendment 1 of ISO/IEC 9506-1.

# 7 Services

Les services et protocoles MMS ont été développés pour être utilisés par un grand nombre d'équipements industriels.

Cet article définit les services et protocole pour les éléments qui ont été identifiés comme ayant besoin d'être définis dans la norme d'accompagnement de MMS.

Il convient que ces définitions soient utilisées lorsque la syntaxe abstraite définie dans la présente norme est négociée.

Le présent article précise ensuite comment les services MMS doivent être utilisés avec les PC.

La plupart des termes et abréviations cités dans cet article sont basés sur la terminologie donnée dans les descriptions des services et protocole MMS (voir l'article 5 de l'ISO/CEI 9506-1 et l'article 5 de l'ISO/CEI 9506-2). En général, se reporter à l'article 5 de l'ISO/CEI 9506-1 pour connaître les règles générales sur l'interprétation des tableaux de services décrits dans la présente norme.

## 7.1 Utilisation des services MMS

### 7.1.1 Additions à la procédure de service MMS

Tout service MMS non mentionné dans les paragraphes ci-après doit se comporter selon la description qui en est faite dans la norme MMS.

#### 7.1.1.1 Service Initiate

La procédure de service Initiate doit être étendue de façon à avoir une action sur le paramètre InitRequestDetail et à admettre le paramètre InitResponseDetail tel que défini ci-après.

NOTE – Le concepteur doit savoir que le temps nécessaire à l'exécution du service ExchangeData dépend du programme d'application du PC et non de la conception du PC. Il convient que cette information serve à déterminer les valeurs adéquates à attribuer aux paramètres OutstandingServicesCalled et OutstandingServicesCalling.

##### 7.1.1.1.1 Paramètre InitRequestDetail

La structure du paramètre InitRequestDetail est décrite dans le tableau 1.

**Tableau 1 – Paramètre InitRequestDetail**

| Parameter Name | Req | Ind |
|---|---|---|
| Proposed Version Number | M | M(=) |
| Proposed Parameter CBB | M | M(=) |
| Services Supported Calling | M | M(=) |

###### 7.1.1.1.1.1 Proposed Version Number

Ce paramètre de type entier doit contenir un nombre représentant le numéro de version mineur de l'ISO/CEI 9506-5, qui est spécifié en 7.1.2. Ce paramètre est le numéro de version mineur proposé qui sera utilisé dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2 de la présente norme, pour cette instance de communication. Un numéro supérieur à un indique la prise en compte, dans ce contexte de présentation, de toutes les versions mineures comprises entre un et le nombre proposé, celui-ci inclus.

La valeur de ce nombre peut être réduite par le fournisseur MMS s'il ne peut prendre en charge la valeur demandée. La valeur présente dans la primitive d'indication doit être inférieure ou égale à la valeur inscrite dans la primitive de demande, mais ne peut être inférieure à un.

# 7 Services

The MMS services and protocol were developed to be used by a wide range of manufacturing devices.

This clause defines the services and protocol for those elements identified as requiring companion standard definition in MMS.

These definitions should be used when the abstract syntax defined in this standard is negotiated.

This clause further clarifies how MMS services are to be used with PCs.

Many of the terms and abbreviations used in this clause use the terminology of MMS service and protocol descriptions (see clause 5 of ISO/IEC 9506-1 and clause 5 of ISO/IEC 9506-2). In particular, refer to clause 5 of ISO/IEC 9506-1 for general rules on how to interpret the service tables in this standard.

## 7.1 Use of the MMS Services

### 7.1.1 Additions to the MMS Service Procedure

All MMS services not mentioned in a subclause herein shall behave exactly as specified in the MMS standard.

#### 7.1.1.1 Initiate Service

The service procedure of the Initiate service shall be extended to act on the InitRequestDetail parameter and to support the InitResponseDetail parameter as specified below.

NOTE – The implementer should be aware that the time to execute the ExchangeData service is determined by the PC application program rather than the design of the PC. This fact should be used in determining the appropriate values for the OutstandingServicesCalled and OutstandingServicesCalling parameters.

##### 7.1.1.1.1 InitRequestDetail Parameter

The structure of the InitRequestDetail parameter is shown in table 1.

**Table 1 – InitRequestDetail parameter**

| Parameter Name | Req | Ind |
|---|---|---|
| Proposed Version Number | M | M(=) |
| Proposed Parameter CBB | M | M(=) |
| Services Supported Calling | M | M(=) |

###### 7.1.1.1.1.1 Proposed Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-5, which is specified in 7.1.2 of this standard. This parameter is the proposed minor version number which will be used in the presentation context derived from the abstract syntax defined in 7.1.2, for this instance of communications. Proposal of a number greater than one indicates support, in this presentation context, for all minor versions between one and the number proposed, inclusive.

The value of this number may be reduced by the MMS-provider if it cannot support the requested value. The value in the indication primitive shall be less than or equal to the value in the request primitive, but not less than one.

#### 7.1.1.1.1.2 Proposed Parameter CBB

Ce paramètre de type chaîne binaire doit spécifier l'ensemble des paramètres CBB qui peuvent être pris en charge dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2, sur l'association d'applications.

La valeur de ce paramètre dans la primitive de demande doit spécifier l'ensemble des paramètres CBB pris en compte par l'utilisateur MMS appelant.

La valeur de ce paramètre dans la primitive d'indication doit spécifier l'intersection de l'ensemble des paramètres CBB pris en compte par l'utilisateur MMS appelant et de l'ensemble des paramètres CBB pris en compte par le fournisseur MMS.

Les paramètres CBB possibles sont spécifiés à l'article 18 de l'ISO/CEI 9506-2. L'affectation d'un paramètre CBB à un bit particulier de type chaîne binaire CBB est décrite à l'article 8 de l'ISO/CEI 9506-2. Une valeur de un dans le bit affecté doit indiquer la prise en charge du CBB correspondant. Une valeur de zéro doit indiquer la non-prise en charge. Tous les bits doivent être codés et tout bit supplémentaire reçu doit être ignoré.

#### 7.1.1.1.1.3 Services Supported Calling

Ce paramètre de type chaîne binaire doit indiquer la prise en charge par l'utilisateur MMS appelant d'un ensemble de services à utiliser dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2, sur l'association d'application.

La valeur du paramètre dans la primitive d'indication doit spécifier l'intersection entre l'ensemble de services pris en charge par l'utilisateur MMS appelant et l'ensemble des services pris en charge par le fournisseur MMS.

L'affectation d'un service à un bit particulier de type chaîne binaire est spécifiée à l'article 19 de l'ISO/CEI 9506-2. Une valeur de un dans le bit assigné doit indiquer la prise en charge du service correspondant. Une valeur de zéro doit indiquer la non-prise en charge. Tous les bits doivent être codés et tout bit supplémentaire reçu doit être ignoré.

Le support de services confirmés doit être défini comme étant la capacité à recevoir une indication de demande et d'exécuter la procédure de service définie pour le rôle appelé. La prise en charge de services non confirmés doit être définie comme étant la capacité à accepter une primitive d'indication et de transmettre les paramètres à l'interface de service. La prise en charge d'un modificateur doit être définie comme étant la capacité à accepter une primitive d'indication qui contient le modificateur et d'exécuter correctement la procédure de service définie par le modificateur.

Si un service, confirmé ou non, ou un modificateur est pris en charge, une PDU de rejet ne doit pas être émise sur réception de ce service ou du service modifié, sauf dans le cas d'une erreur de protocole. Si un service, confirmé ou non, ou un modificateur n'est pas pris en charge, une PDU de rejet doit être émise sur réception de ce service ou du service modifié, avec le code de rejet SERVICE NON RECONNU.

#### 7.1.1.1.2 InitResponseDetail Parameter

La structure du paramètre InitResponseDetail est décrite dans le tableau 2.

#### 7.1.1.1.1.2 Proposed Parameter CBB

This parameter, of type bitstring, shall specify the set of parameter conformance building blocks (CBB) which are proposed to be supported in the presentation context derived from the abstract syntax defined in 7.1.2, on this application association.

The value of this parameter in the request primitive shall specify the set of Parameter CBBs supported by the Calling MMS-user.

The value of this parameter in the indication primitive shall specify the intersection of the set of Parameter CBBs supported by the Calling MMS-user and the set of Parameter CBBs supported by the MMS-provider.

The possible Parameter CBBs are specified in ISO/IEC 9506-2, clause 18. The assignment of a Parameter CBB to an individual bit of a CBB bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

#### 7.1.1.1.1.3 Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in 7.1.2 on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of services supported by the Calling MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the modifier and to execute properly the service procedure defined for the modifier.

If a confirmed service, an unconfirmed service, or modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

#### 7.1.1.1.2 InitResponseDetail Parameter

The structure of the InitResponseDetail parameter is shown in table 2.

**Tableau 2 – Paramètre InitResponseDetail**

| Parameter Name | Rsp | Cnf |
|---|---|---|
| Negotiated Version Number | M | M(=) |
| Negotiated Parameter CBB | M | M(=) |
| Services Supported Called | M | M |

#### 7.1.1.1.2.1 Negotiated Version Number

Ce paramètre de type entier doit contenir un nombre qui représente un numéro de version mineur de l'ISO/CEI 9506-5, qui est définie en 7.1.2. Ce paramètre doit être le numéro de version mineur proposé qui sera utilisé dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2, pour cette instance de communication. Ce nombre doit être inférieur ou égal au paramètre Proposed Version Number dans la primitive de demande. Il ne doit pas être inférieur à un.

#### 7.1.1.1.2.2 Negotiated Parameter CBB

Ce paramètre de type chaîne binaire doit spécifier l'ensemble de CBB de paramètres pris en charge dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2, sur l'association d'application.

La valeur de ce paramètre dans la primitive de réponse doit spécifier l'intersection de l'ensemble de CBB de paramètres pris en charge par l'utilisateur MMS appelé et l'ensemble de CBB de paramètres spécifiés par le paramètre CBB de paramètres proposés dans la primitive d'indication.

Les CBB de paramètres possibles sont spécifiés à l'article 18 de l'ISO/CEI 9506-2. L'affectation d'un CBB de paramètre à un bit particulier d'un CBB de type chaîne binaire est spécifiée à l'article 8 de l'ISO/CEI 9506-2. Une valeur de un dans le bit affecté doit indiquer la prise en charge du CBB correspondant. Une valeur de zéro doit indiquer la non-prise en charge. Tous les bits doivent être codés et tous les bits supplémentaires reçus doivent être ignorés.

#### 7.1.1.1.2.3 Services supported called

Ce paramètre de type chaîne binaire doit spécifier la prise en charge par l'utilisateur MMS appelé d'un ensemble de services dans le contexte de présentation résultant de la syntaxe abstraite définie en 7.1.2, sur l'association d'application.

La valeur du paramètre de la primitive de confirmation doit spécifier l'intersection de l'ensemble des services supportés par l'utilisateur MMS appelé et l'ensemble de services supportés par le fournisseur MMS.

L'affectation d'un service à un bit de type chaîne binaire est spécifiée à l'article 19 de l'ISO/CEI 9506-2. Une valeur de un pour le bit affecté doit indiquer la prise en charge du service correspondant. Une valeur de zéro indique la non-prise en charge. Tous les bits doivent être codés et tous les bits supplémentaires reçus doivent être ignorés.

La prise en charge de services confirmés doit être définie comme étant la capacité à recevoir une indication de demande et à exécuter la procédure de service définie pour le rôle de répondeur. La prise en charge de services non confirmés doit être définie comme étant la capacité à accepter une primitive d'indication et de transmettre les paramètres à l'interface de services. La prise en charge d'un modificateur doit être définie comme étant la capacité à accepter une primitive d'indication qui contient le modificateur et d'exécuter correctement la procédure de service définie pour le modificateur.

**Table 2 – InitResponseDetail parameter**

| Parameter Name | Rsp | Cnf |
|---|---|---|
| Negotiated Version Number | M | M(=) |
| Negotiated Parameter CBB | M | M(=) |
| Services Supported Called | M | M |

#### 7.1.1.1.2.1 Negotiated Version Number

This parameter, of type integer, shall contain a number which represents a minor version number of ISO/IEC 9506-5, which is specified in 7.1.2. This parameter shall be the minor version number which will be used in the presentation context derived from the abstract syntax defined in 7.1.2, for this instance of communications. This number shall be less than or equal to the Proposed Version Number parameter in the request primitive. It shall not be reduced to less than one.

#### 7.1.1.1.2.2 Negotiated Parameter CBB

This parameter, of type bitstring, shall specify the set of parameter conformance building blocks (CBB) which are supported in the presentation context derived from the abstract syntax defined in 7.1.2, on this application association.

The value of this parameter in the response primitive shall specify the intersection of the set of Parameter CBBs supported by the Called MMS-user and the set of Parameter CBBs specified by the Proposed Parameter CBB parameter in the indication primitive.

The possible Parameter CBBs are specified in ISO/IEC 9506-2, clause 18. The assignment of a Parameter CBB to an individual bit of a CBB bitstring type is specified in ISO/IEC 9506-2, clause 8. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

#### 7.1.1.1.2.3 Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS-user of a set of services for use in the presentation context derived from the abstract syntax defined in 7.1.2, on the application association.

The value of the parameter in the confirmation primitive shall specify the intersection of the set of services supported by the Called MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in ISO/IEC 9506-2, clause 19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the modifier and to execute properly the service procedure defined for the modifier.

Si un service, confirmé ou non, ou encore un modificateur est pris en charge, une PDU de rejet ne doit pas être émise sur réception de ce service ou du service modifié, sauf dans le cas d'une erreur de protocole. Si un service, confirmé ou non, ou encore un modificateur n'est pas pris en charge, une PDU de rejet doit être émise sur réception de ce service ou de ce service modifié avec code du rejet SERVICE NON RECONNU.

#### 7.1.1.2 Service Start

La procédure de service du service Start doit être élargie.

L'attribut I/O State de l'invocation de programme doit être défini en fonction du paramètre I/O State de l'argument CS-Start-Request. La fonction d'interface pour les capteurs et les actionneurs doit être telle que spécifiée dans le tableau suivant:

| I/O State | Interface capteur | Interface actionneur |
|---|---|---|
| Controlled | Fournit des données du capteur au programme d'application | Utilise des valeurs du programme d'application |
| Hold Outputs | Fournit des données du capteur au programme d'application | Garde les valeurs en cours |
| Hold Current State | Garde les valeurs en cours | Garde les valeurs en cours |

Si le paramètre I/O State est omis, la valeur par défaut (Controlled) doit être utilisée.

La procédure de service MMS, avec les élargissements prévus dans la présente norme d'accompagnement, doit être répétée pour toutes les invocations de programme dont l'attribut Independent est égal à FAUX, et dont l'attribut Program Invocation Reference est égal au nom de l'invocation de programme démarré. La valeur des paramètres Execution Argument et I/O State à utiliser pour démarrer chacune des invocations de programme doit être identique à celle utilisée pour démarrer cette invocation de programme. Les résultats des tentatives de démarrage de chacune des invocations de programme doivent être ignorés.

La structure de l'argument CS-Start-Request est décrite dans le tableau 3.

**Tableau 3 – Argument CS-Start-Request**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

#### 7.1.1.3 Service Resume

La procédure de service du service Resume doit être élargie.

L'attribut I/O State de l'invocation de programme doit être défini en fonction du paramètre I/O State de l'argument CS-Resume-Request. La fonction d'interface pour les capteurs et les actionneurs doit être telle que spécifiée dans le tableau suivant:

| I/O State | Interface capteur | Interface actionneur |
|---|---|---|
| Controlled | Fournit des données du capteur au programme d'application | Utilise des valeurs du programme d'application |
| Hold Outputs | Fournit des données du capteur au programme d'application | Garde les valeurs en cours |
| Hold Current State | Garde les valeurs en cours | Garde les valeurs en cours |

If a confirmed service, an unconfirmed service, or modifier is supported, then a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, then a Reject PDU shall be issued on receipt of that service or modified service with a reject code of UNRECOGNIZED SERVICE.

#### 7.1.1.2 Start Service

The service procedure of the Start service shall be extended.

The I/O State attribute of the Program Invocation shall be set according to the I/O State parameter of the CS-Start-Request argument. The Interface Function to Sensors and Actuators shall be directed as specified in the following table:

| I/O State | Sensor Interface | Actuator Interface |
|---|---|---|
| Controlled | Provide Sensor data to application program | Use application program values |
| Hold Outputs | Provide Sensor data to application program | Hold current values |
| Hold Current State | Hold current values | Hold current values |

If the I/O State parameter is omitted, the default (controlled) shall be used.

The MMS service procedure, with the extensions specified in this companion standard, shall be repeated for all Program Invocations which have the attribute Independent equal to FALSE, and the attribute ProgramInvocationReference equal to the name of the started Program Invocation. The value for the Execution Argument and I/O State parameters to be used for starting each of these Program Invocations shall be the same as that used to start this Program Invocation. The results of trying to start each of these Program Invocations shall be ignored.

The structure of the CS-Start-Request argument is shown in table 3.

**Table 3 – CS-Start-Request argument**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

#### 7.1.1.3 Resume Service

The service procedure of the Resume service shall be extended.

The I/O State attribute of the Program Invocation shall be set according to the I/O State parameter of the CS-Resume-Request argument. The Interface Function to Sensors and Actuators shall be directed as specified in the following table:

| I/O State | Sensor Interface | Actuator Interface |
|---|---|---|
| Controlled | Provide Sensor data to application program | Use application program values |
| Hold Outputs | Provide Sensor data to application program | Hold current values |
| Hold Current State | Hold current values | Hold current values |

Si la paramètre d'I/O State est omis, la valeur par défaut (Controlled) doit être utilisée.

La procédure de service MMS, avec les élargissements spécifiés dans la présente norme d'accompagnement, doit être répétée pour toutes les invocations de programme dont l'attribut Independent est égal à FAUX et l'attribut Program Invocation Reference égal au nom de l'invocation de programme reprise. Les valeurs des paramètres Execution Argument et I/O State qui doivent être utilisées par la reprise de chacune des invocations de programme doivent être identiques à celles utilisées pour relancer cette invocation de programme. Les résultats des tentatives de reprise de chaque invocation de programme doivent être ignorés.

La structure de l'argument CS-Resume-Request est décrite dans le tableau 4.

**Tableau 4 – Argument CS-Resume-Request**

| Parameter Name | Req | Ind |
|---|---|---|
| I/0 State | U | U(=) |

#### 7.1.1.4 Service Stop

La procédure de service du service Stop doit être élargie.

L'attribut I/O State de l'invocation de programme doit être défini en fonction du paramètre I/O State de l'argument CS-Stop-Request. La fonction d'interface pour les capteurs et les actionneurs doit être telle que spécifiée dans le tableau suivant:

| I/O State | Interface actionneur |
|---|---|
| Hold Current State | Garde les valeurs en cours |
| Implementer State | Fixe toutes les sorties contrôlées par ce programme d'application en fonction de la spécification des concepteurs, puis garde les valeurs en cours |
| Zero Outputs | Fixe toutes les sorties contrôlées par ce programme d'application à zéro, puis garde les valeurs en cours |
| User Specified | Fixe toutes les sorties contrôlées par ce programme d'application à des valeurs spécifiées par l'utilisateur, puis garde les valeurs en cours |

Si le paramètre I/O State est omis, la valeur par défaut (Implementer State) doit être utilisée.

La procédure de service MMS, avec les élargissements précisés dans la présente norme d'accompagnement, doit être répétée pour toutes les invocations de programme dont l'attribut Independent est égal à FAUX et l'attribut Program Invocation Reference égal au nom de l'invocation de programme arrêtée. La valeur du paramètre I/O State à utiliser pour arrêter chacune de ces invocations de programme doit être identique à celle utilisée pour arrêter cette invocation de programme. Les résultats des tentatives d'arrêt de chacune de ces invocations de programme doivent être ignorés.

La structure de l'argument CS-Stop-Request est décrit dans le tableau 5.

**Tableau 5 – Argument CS-Stop-Request**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

If the I/O State parameter is omitted, the default (controlled) shall be used.

The MMS service procedure, with the extensions specified in this companion standard, shall be repeated for all Program Invocations which have the attribute Independent equal to FALSE, and the attribute ProgramInvocationReference equal to the name of the resumed Program Invocation. The value for the Execution Argument and I/O State parameters to be used for resuming each of these Program Invocations shall be the same as that used to resume this Program Invocation. The results of trying to resume each of these Program Invocations shall be ignored.

The structure of the CS-Resume-Request argument is shown in table 4.

**Table 4 – CS-Resume-Request argument**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

#### 7.1.1.4 Stop Service

The service procedure of the Stop service shall be extended.

The I/O State attribute of the Program Invocation shall be set according to the I/O State parameter of the CS-Stop-Request argument. The Interface Function to Sensors and Actuators shall be directed as specified in the following table:

| I/O State | Actuator Interface |
|---|---|
| Hold Current State | Hold current values |
| Implementer State | Set all outputs controlled by this application program according to the implementers specification, then hold current values |
| Zero Outputs | Set all outputs controlled by this application program to zero, then hold current values |
| User Specified | Set the specified outputs controlled by this application program to the user specified values, then hold current values |

If the I/O State parameter is omitted, the default (implementer state) shall be used.

The MMS service procedure, with the extensions specified in this companion standard, shall be repeated for all Program Invocations which have the attribute Independent equal to FALSE, and the attribute ProgramInvocationReference equal to the name of the stopped Program Invocation. The value for the I/O State parameter to be used for stopping each of these Program Invocations shall be the same as that used to stop this Program Invocation. The results of trying to stop each of these Program Invocations shall be ignored.

The structure of the CS-Stop-Request argument is shown in table 5.

**Table 5 – CS-Stop-Request argument**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

#### 7.1.1.5 Service Reset

La procédure de service du service Reset doit être élargie.

La procédure de service MMS, avec les élargissements précisés dans la présente norme d'accompagnement, doit être répétée pour toutes les invocations de programme dont l'attribut Independent est égal à FAUX et l'attribut Program Invocation Reference égal au nom de l'invocation de programme réinitialisée. Les résultats des tentatives de réinitialisation de chacune de ces invocations de programme doivent être ignorés.

#### 7.1.1.6 Service Kill

La procédure de service du service Kill doit être élargie.

L'attribut I/O State de l'invocation de programme doit être définie en fonction du paramètre I/O State de l'argument CS-Kill-Request. La fonction d'interface pour les capteurs et les actionneurs doit être telle que spécifiée dans le tableau suivant:

| I/O State | Interface actionneur |
|---|---|
| Hold Current State | Garde les valeurs en cours |
| Implementer State | Fixe toutes les sorties contrôlées par ce programme d'application en fonction de la spécification des concepteurs, puis garde les valeurs en cours |
| Zero Outputs | Fixe toutes les sorties contrôlées par ce programme d'application à zéro, puis garde les valeurs en cours |
| User Specified | Fixe toutes les sorties contrôlées par ce programme d'application à des valeurs spécifiées par l'utilisateur, puis garde les valeurs en cours |

Si le paramètre I/O State est omis, la valeur par défaut (Implementer State) doit être utilisée.

La procédure de service MMS, avec les élargissements définis dans la présente norme d'accompagnement, doit être répétée pour toutes les invocations de programme dont l'attribut Independent est FAUX et l'attribut Program Invocation Reference égal au nom de l'invocation de programme arrêté. La valeur du paramètre I/O State à utiliser pour détruire chacune de ces invocations de programme doit être identique à celle utilisée pour détruire cette invocation de programme. Les résultats des tentatives de destruction de chacune de ces invocations de programmes doivent être ignorés.

La structure de l'argument CS-Kill-Request est définie dans le tableau 6.

**Tableau 6 – Argument CS-Kill-Request**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |

#### 7.1.1.7 Service GetProgramInvocationAttributes

La procédure du service GetProgramInvocationAttributes doit être élargie.

L'attribut I/O State est un paramètre de l'argument CS-GetProgram Invocation Attributes-Response. Si l'attribut Independent est FAUX, l'attribut Program Invocation Reference est aussi un paramètre de l'argument CS-GetProgram Invocation Attributes- Response.

La structure de l'argument CS-GetProgramInvocationAttributes-Response est définie dans le tableau 7.

#### 7.1.1.5 Reset Service

The service procedure of the Reset service shall be extended.

The MMS service procedure, with the extensions specified in this companion standard, shall be repeated for all Program Invocations which have the attribute Independent equal to FALSE, and the attribute ProgramInvocationReference equal to the name of the reset Program Invocation. The results of trying to reset each of these Program Invocations shall be ignored.

#### 7.1.1.6 Kill Service

The service procedure of the Kill service shall be extended.

The I/O State attribute of the Program Invocation shall be set according to the I/O State parameter of the CS-Kill-Request argument. The Interface Function to Sensors and Actuators shall be directed as specified in the following table:

| I/O State | Actuator Interface |
|---|---|
| Hold Current State | Hold current values |
| Implementer State | Set all outputs controlled by this application program according to the implementers specification, then hold current values |
| Zero Outputs | Set all outputs controlled by this application program to zero, then hold current values |
| User Specified | Set the specified outputs controlled by this application program to the user specified values, then hold current values |

If the I/O State parameter is omitted, the default (implementer state) shall be used.

The MMS service procedure, with the extensions specified in this companion standard, shall be repeated for all Program Invocations which have the attribute Independent equal to FALSE, and the attribute ProgramInvocationReference equal to the name of the killed Program Invocation. The value for the I/O State parameter to be used for killing each of these Program Invocations shall be the same as that used to kill this Program Invocation. The results of trying to kill each of these Program Invocations shall be ignored.

The structure of the CS-Kill-Request argument is shown in table 6.

**Table 6 – CS-Kill-Request argument**

| Parameter Name | Req | Ind |
|---|---|---|
| I/O State | U | U(=) |



#### 7.1.1.7 GetProgramInvocationAttributes Service

The service procedure of the GetProgramInvocationAttributes service shall be extended.

The I/O State attribute is reported as a parameter in the CS-GetProgramInvocationAttributes-Response argument. If the attribute Independent has a value of FALSE, the Program Invocation Reference attribute is also reported as a parameter in the CS-GetProgram InvocationAttributes-Response argument.

The structure of the CS-GetProgramInvocationAttributes-Response argument is shown in table 7.

**Tableau 7 – Argument CS-GetProgramInvocationAttributes-Response**

| Parameter Name | Rsp | Cnf |
|---|---|---|
| I/O State | M | M(=) |
| Program Invocation Reference | C | C(=) |

#### 7.1.1.8 Service CreateProgramInvocation

La procédure du service CreateProgramInvocation doit être élargie.

L'attribut I/O State est initialisé à ImplementerState. L'interface vers les actionneurs doit définir toutes les sorties contrôlées par ce programme d'application en fonction de la spécification des concepteurs, puis garder les valeurs en cours.

Si le paramètre Program Invocation Reference est présent, l'attribut Program Invocation Reference doit avoir la valeur du paramètre Program Invocation Reference et l'attribut Independent doit être égal à FAUX. Si l'invocation de programme référencée a l'attribut Independent FAUX et que la mise en oeuvre ne le permet pas, une erreur «Définition, Autre" doit être renvoyée.

Si le paramètre Program Invocation Reference n'est pas présent, l'attribut Independent doit être VRAI.

La structure de l'argument CS-CreateProgramInvocation-Request est définie dans le tableau 8.

**Tableau 8 – Argument CS-CreateProgramInvocation-Request**

| Parameter Name | Req | Ind |
|---|---|---|
| Program Invocation Reference | U | U(=) |

### 7.1.2 Additions à la définition des protocoles MMS

L'ISO-9506-MMS-CSPC-1 représente la version numéro 1 de la norme d'accompagnement pour les Automates Programmables.

NOTE – Certains termes utilisés dans les paragraphes ci-dessous ne sont pas définis dans cette partie de la norme. Ils sont répertoriés dans la section «Imports" suivante et sont définis dans l'ISO/CEI 9506-2.

```
-------------------------------------------------------------
ISO-9506-MMS-CSPC-1 {iso standard 9506 part (5) mms-cspc-version1 (1)}
DEFINITIONS::= BEGIN
  IMPORTS                                   MMSpdu,
                                                 ParameterSupportOptions,
                                                 ServiceSupportOptions,
                                                 Integer16,
                                                 StatusResponse,
                                                 Identifier,
                                                 Data,
                                                 ObjectName

  FROM MMS-General-Module-1                      {iso standard 9506 part

(2) mms-general-module-version1 (2)};
InitRequestDetail::= SEQUENCE {

IMPLICIT Integer16,                         proposedVersionNumber       [0]

IMPLICIT                                    proposedParameterCBB        [1]

  ParameterSupportOptions

IMPLICIT                                    servicesSupportedCalling    [2]

                                                      ServiceSupportOptions
                                            }
```

**Table 7 – CS-GetProgramInvocationAttributes-Response argument**

| Parameter Name | Rsp | Cnf |
|---|---|---|
| I/O State | M | M(=) |
| Program Invocation Reference | C | C(=) |

### 7.1.1.8 CreateProgramInvocation Service

The service procedure of the CreateProgramInvocation service shall be extended.

The I/O State attribute is set to implementerState. Direct the Interface to Actuators to set all outputs controlled by this application program according the the implementers specification, then hold current values.

If the Program Invocation Reference parameter is present, then the Program Invocation Reference attribute is set to the value of the Program Invocation Reference parameter and the Independent attribute is set to FALSE. If the referenced Program Invocation has the Independent attribute set to FALSE and the implementation does not permit this, then an error of "Definition, Other" shall be returned.

If the Program Invocation Reference parameter is not present, then the Independent attribute is set to TRUE.

The structure of the CS-CreateProgramInvocation-Request argument is shown in table 8.

**Table 8 – CS-CreateProgramInvocation-Request argument**

| Parameter Name | Req | Ind |
|---|---|---|
| Program Invocation Reference | U | U(=) |

### 7.1.2 Additions to the MMS Protocol Definition

ISO-9506-MMS-CSPC-1 represents version number 1 of the MMS Companion Standard for Programmable Controllers.

NOTE – There are some terms that are used in the following productions which are not defined in this part of this standard. These are listed in the "IMPORTS" section of what follows and are defined in ISO/IEC 9506-2.

```
--------------------------------------------------------------

ISO-9506-MMS-CSPC-1 {iso standard 9506 part(5) mms-cspc-version1(1)}
DEFINITIONS ::= BEGIN
   IMPORTS  MMSpdu,
            ParameterSupportOptions,
            ServiceSupportOptions,
            Integer16,
            StatusResponse,
            Identifier,
            Data,
            ObjectName
   FROM MMS-General-Module-1
            {iso standard 9506 part(2) mms-general-module-version1(2)};
InitRequestDetail ::= SEQUENCE {
         proposedVersionNumber       [0] IMPLICIT Integer16,
         proposedParameterCBB        [1] IMPLICIT
                                  ParameterSupportOptions,
         servicesSupportedCalling    [2] IMPLICIT
                                  ServiceSupportOptions
         }
```

```
InitResponseDetail::= SEQUENCE {
  negociatedVersionNumber               [0]   IMPLICIT Integer16,
  negociatedParameterCBB                [1]   IMPLICIT
                                              ParameterSupportOptions,
  sercicesSupportedCalled               [2]   IMPLICIT
                                              ServiceSupportOptions
  }
CS-Status-Request::= NULL
CS-GetNameList-Request::= NULL
CS-Input-Request::= NULL
CS-Output-Request::= NULL
CS-InitiateDownloadSequence-Request::=NULL
CS-DownloadSegment-Request::= NULL
CS-TerminateDownloadSequence-Request::= NULL
CS-InitiateUploadSequence-Request::= NULL
CS-UploadSegment-Request::= NULL
CS-TerminateUploadSequence-Request::= NULL
CS-RequestDomainDownload-Request::= NULL
CS-RequestDomainUpload-Request::= NULL
CS-LoadDomainContent-Request::= NULL
CS-StoreDomainContent-Request::= NULL
CS-DeleteDomain-Request::= NULL
CS-GetDomainAttributes-Request::= NULL
CS-CreateProgramInvocation-Request::= Identifier
                                        -- Nothing shall be transmitted
if no identifier is given
CS-DeleteProgramInvocation-Request::= NULL
IoState::= INTEGER {
  controlled       (0),
  holdOutputs      (1),
  holdCurrentState (2),
  implementerState (3),
  zeroOutputs      (4),
  userSpecified          (5)
  }
CS-Start-Request::= OPTIONAL IoState -- Only values 0,1,2 are valid;
default is 0
CS-Stop-Request::= OPTIONAL IoState -- Only values 2,3,4,5 are valid;
default is 3
CS-Resume-Request::= OPTIONAL IoState -- Only values 0,1,2 are valid;
default is 0
CS-Reset-Request::= NULL
CS-Kill-Request::= OPTIONAL IoState -- Only values 2,3,4,5 are valid;
default is 3
CS-GetProgramInvocationAttributes-Request::= NULL
CS-DefineEventCondition-Request::= NULL
CS-DeleteEventCondition-Request::= NULL
CS-GetEventConditionAttributes-Request::= NULL
CS-ReportEventConditionStatus-Request::= NULL
CS-AlterEventConditionMonitoring-Request::= NULL
CS-TriggerEvent-Request::= NULL
CS-DefineEventAction-Request::= NULL
CS-DeleteEventAction-Request::= NULL
CS-GetEventActionAttributes-Request::= NULL
CS-ReportEventActionStatus-Request::= NULL
CS-DefineEventEnrollment-Request::= NULL
CS–DeleteEventEnrollment-Request::= NULL
CS-AlterEventenrollment-Request::= NULL
CS-ReportEventEnrollmentStatus-Request::= NULL
CS-GetEventEnrollmentAttributes-Request::= NULL
CS-AchnowledgeEventNotification-Request::= NULL
CS-GetAlarmSummary-Request::= NULL
CS–GetAlarmEnrollmentSummary-Request::=NULL
CS-ReadJournal-Request::= NULL
CS-WriteJournal-Request::= NULL
CS-InitializeJournal-Request::= NULL
CS-ReportJournalStatus-Request::= NULL
```

```
InitResponseDetail ::= SEQUENCE {
        negotiatedVersionNumber     [0] IMPLICIT Integer16,
        negotiatedParameterCBB      [1] IMPLICIT
                                ParameterSupportOptions,
        servicesSupportedCalled     [2] IMPLICIT
                                ServiceSupportOptions
        }
CS-Status-Request ::= NULL
CS-GetNameList-Request ::= NULL
CS-Input-Request ::= NULL
CS-Output-Request ::= NULL
CS-InitiateDownloadSequence-Request ::= NULL
CS-DownloadSegment-Request ::= NULL
CS-TerminateDownloadSequence-Request ::= NULL
CS-InitiateUploadSequence-Request ::= NULL
CS-UploadSegment-Request ::= NULL
CS-TerminateUploadSequence-Request ::= NULL
CS-RequestDomainDownload-Request ::= NULL
CS-RequestDomainUpload-Request ::= NULL
CS-LoadDomainContent-Request ::= NULL
CS-StoreDomainContent-Request ::= NULL
CS-DeleteDomain-Request ::= NULL
CS-GetDomainAttributes-Request ::= NULL
CS-CreateProgramInvocation-Request ::= Identifier
             --  Nothing  shall  be  transmitted  if  no  identifier  is
given
CS-DeleteProgramInvocation-Request ::= NULL
IoState ::= INTEGER  {
controlled          (0),
holdOutputs         (1),
holdCurrentState    (2),
implementerState    (3),
zeroOutputs         (4),
userSpecified       (5)
      }
CS-Start-Request   ::= OPTIONAL IoState   -- Only values 0, 1, 2 are
valid; default is 0
CS-Stop-Request    ::= OPTIONAL IoState -- Only values 2, 3, 4, 5 are
valid; default is 3
CS-Resume-Request  ::= OPTIONAL IoState   -- Only values 0, 1, 2 are
valid; default is 0
CS-Reset-Request ::= NULL
CS-Kill-Request ::= OPTIONAL IoState   -- Only values 2, 3, 4, 5 are
valid; default is 3
CS-GetProgramInvocationAttributes-Request ::= NULL
CS-DefineEventCondition-Request ::= NULL
CS-DeleteEventCondition-Request ::= NULL
CS-GetEventConditionAttributes-Request ::= NULL
CS-ReportEventConditionStatus-Request ::= NULL
CS-AlterEventConditionMonitoring-Request ::= NULL
CS-TriggerEvent-Request ::= NULL
CS-DefineEventAction-Request ::= NULL
CS-DeleteEventAction-Request ::= NULL
CS-GetEventActionAttributes-Request ::= NULL
CS-ReportEventActionStatus-Request ::= NULL
CS-DefineEventEnrollment-Request ::= NULL
CS-DeleteEventEnrollment-Request ::= NULL
CS-AlterEventEnrollment-Request ::= NULL
CS-ReportEventEnrollmentStatus-Request ::= NULL
CS-GetEventEnrollmentAttributes-Request ::= NULL
CS-AcknowledgeEventNotification-Request ::= NULL
CS-GetAlarmSummary-Request ::= NULL
CS-GetAlarmEnrollmentSummary-Request ::= NULL
CS-ReadJournal-Request ::= NULL
CS-WriteJournal-Request ::= NULL
CS-InitializeJournal-Request ::= NULL
CS-ReportJournalStatus-Request ::= NULL
```

```
CS-CreateJournal-Request::= NULL
CS-DeleteJournal-Request::= NULL
CS-GetCapabilityList-Request::= NULL
CS-Status-Response::= NULL
CS-GetNameList-Response::= NULL
CS-Input-Response::= NULL
CS-Ouput-Response::= NULL
CS-InitiateDownloadSequence-Response::= NULL
CS-DownloadSegment-Response::= NULL
CS-TerminateDownloadSequence-Response::= NULL
CS-InitiateUploadSequence-Response::= NULL
CS-UploadSegment-Response::= NULL
CS-TerminateUploadSequence-Response::= NULL
CS-RequestDomainDownload-Response::= NULL
CS-RequestDomainUpload-Response::= NULL
CS-LoadDomainContent-Response::= NULL
CS-StoreDomainContent-Response::= NULL
CS-DeleteDomain-Response::= NULL
CS-GetDomainAttributes-Response::= NULL
CS-CreateProgramInvocation-Response::= NULL
CS-DeleteProgramInvocation-Response::= NULL
CS-Start-Response::= NULL
CS-Stop-Response::= NULL
CS-Resume-Response::= NULL
CS-Reset-Response::= NULL
CS-Kill-Response::= NULL
CS-GetProgramInvocationAttributes-Response::= SEQUENCE {
  ioState                                [0] IoState,
  programInvocationReference [1] IMPLICIT Identifier OPTIONAL
   -- Only provided if value of Independent attribute is FALSE
  }
CS-DefineEventCondition-Response::= NULL
CS-DeleteEventCondition-Response::= NULL
CS-GetEventConditionAttributes-Response::= NULL
CS-ReportEventConditionStatus-Response::= NULL
CS-AlterEventConditionMonitoring-Response::= NULL
CS-TriggerEvent-Response::= NULL
CS-DefineEventAction-Response::= NULL
CS-DeleteEventAction-Response::= NULL
CS-GetEventActionAttributes-Response::= NULL
CS-ReportEventActionStatus-Response::= NULL
CS-DefineEventEnrollment-Response::= NULL
CS-DeleteEventEnrollment-Response::= NULL
CS-AlterEventEnrollment-Response::= NULL
CS-ReportEventEnrollmentStatus-Response::= NULL
CS-GetEventEnrollmentAttributes-Response::= NULL
CS-AcknowledgeEventNotification-Response::= NULL
CS-GetAlarmSummary-Response::= NULL
CS-GetAlarmEnrollmentSummary-Response::= NULL
CS-ReadJournal-Response::= NULL
CS-WriteJournal-Response::= NULL
CS-InitializeJournal-Response::= NULL
CS-ReportJournalStatus-Response::= NULL
CS-CreateJournal-Response::= NULL
CS-DeleteJournal-Response::= NULL
CS-GetCapabilityList-Response::= NULL
CS-UnsolicitedStatus::= NULL
CS-EventNotification::= NULL
CS-EventNotification::= NULL
CS-Service-Error::= NULL
CS-GetDataExchangeAttributes-Request::= NULL
CS-GetDataExchangeAttributes-Response::= NULL
CS-ExchangeData-Request::= NULL
CS-ExchangeData-Response::= NULL
AdditionalService-Error::= NULL
AdditionalUnconfirmedService::= NULL
CsAdditionalObjectClasses::= NULL
```

```
CS-CreateJournal-Request ::= NULL
CS-DeleteJournal-Request ::= NULL
CS-GetCapabilityList-Request ::= NULL
CS-Status-Response ::= NULL
CS-GetNameList-Response ::= NULL
CS-Input-Response ::= NULL
CS-Output-Response ::= NULL
CS-InitiateDownloadSequence-Response ::= NULL
CS-DownloadSegment-Response ::= NULL
CS-TerminateDownloadSequence-Response ::= NULL
CS-InitiateUploadSequence-Response  ::= NULL
CS-UploadSegment-Response ::= NULL
CS-TerminateUploadSequence-Response ::= NULL
CS-RequestDomainDownload-Response ::= NULL
CS-RequestDomainUpload-Response ::= NULL
CS-LoadDomainContent-Response ::= NULL
CS-StoreDomainContent-Response ::= NULL
CS-DeleteDomain-Response ::= NULL
CS-GetDomainAttributes-Response ::= NULL
CS-CreateProgramInvocation-Response ::= NULL
CS-DeleteProgramInvocation-Response ::= NULL
CS-Start-Response ::= NULL
CS-Stop-Response ::= NULL
CS-Resume-Response ::= NULL
CS-Reset-Response ::= NULL
CS-Kill-Response ::= NULL
CS-GetProgramInvocationAttributes-Response ::= SEQUENCE {
   ioState                    [0] IoState,
   programInvocationReference[1] IMPLICIT Identifier OPTIONAL
      -- Only provided if value of Independent attribute is FALSE
   }
CS-DefineEventCondition-Response ::= NULL
CS-DeleteEventCondition-Response ::= NULL
CS-GetEventConditionAttributes-Response ::= NULL
CS-ReportEventConditionStatus-Response ::= NULL
CS-AlterEventConditionMonitoring-Response ::= NULL
CS-TriggerEvent-Response ::= NULL
CS-DefineEventAction-Response ::= NULL
CS-DeleteEventAction-Response ::= NULL
CS-GetEventActionAttributes-Response ::= NULL
CS-ReportEventActionStatus-Response ::= NULL
CS-DefineEventEnrollment-Response ::= NULL
CS-DeleteEventEnrollment-Response ::= NULL
CS-AlterEventEnrollment-Response ::= NULL
CS-ReportEventEnrollmentStatus-Response ::= NULL
CS-GetEventEnrollmentAttributes-Response ::= NULL
CS-AcknowledgeEventNotification-Response ::= NULL
CS-GetAlarmSummary-Response ::= NULL
CS-GetAlarmEnrollmentSummary-Response ::= NULL
CS-ReadJournal-Response ::= NULL
CS-WriteJournal-Response ::= NULL
CS-InitializeJournal-Response ::= NULL
CS-ReportJournalStatus-Response ::= NULL
CS-CreateJournal-Response ::= NULL
CS-DeleteJournal-Response ::= NULL
CS-GetCapabilityList-Response ::= NULL
CS-UnsolicitedStatus ::= NULL
CS-EventNotification ::= NULL
CS-Service-Error ::= NULL
CS-GetDataExchangeAttributes-Request ::= NULL
CS-GetDataExchangeAttributes-Response ::= NULL
CS-ExchangeData-Request ::= NULL
CS-ExchangeData-Response ::= NULL
AdditionalService-Error ::= NULL
AdditionalUnconfirmedService ::= NULL
CsAdditionalObjectClasses ::= NULL
```

```
EN-Additional-Detail::= NULL
EE-Additional-Detail::= NULL
JOU-Additional-Detail::= NULL
--------------------------------------------------------------------
```

## 8 Noms normalisés

Cet article précise les noms normalisés définis dans cette norme.

### 8.1 Objets Domaines – AUCUN

### 8.2 Objets Invocation de Programme – AUCUN

### 8.3 Objets Variable Nommée

Il existe trois objets Variable Nommée normalisés:

a) P_DDATE

b) P_PCSTATE

c) P_PCSTATUS

P_DDATE est une variable spécifique au domaine. Elle contient la date de la modification la plus récente du programme d'application qui est contenu dans le domaine. Elle est générée au niveau local.

P_PCSTATE est une chaîne binaire qui contient l'information sur l'état de santé et la valeur d'état du PC. Le paragraphe 5.1.4. contient une description sémantique de l'état et de la santé du PC. Cette variable est fournie aux clients qui souhaitent obtenir l'état en même temps que les données. La méthodologie pour maintenir cette variable à jour doit être précisée par le concepteur. Les méthodologies possibles comprennent ce qui suit:

a) la périodicité;

b) dès que la variable est lue;

c) une fois par temps de cycle de programme.

P_PCSTATUS est une variable structurée qui contient l'information relative à la santé et à l'état du PC et de ses sous-systèmes. Cette variable est fournie aux clients qui souhaitent obtenir des informations complémentaires sur l'état. La méthode pour maintenir à jour cette variable doit être définie par le concepteur dans la PICS.

Les définitions formelles de chacune de ces variables sont données ci-après.

#### 8.3.1 P_DDATE

Un ensemble d'objets Variable Nommée spécifiques au domaine doit être défini pour rassembler l'information en provenance de domaines existants dans le VMD.

```
Object: Named Variable
Key Attribute: Variable Name = Domain-specific {
                  domainID,
                  itemID   "P_DDATE" }
     Attribute: Reference to Access Control List = 'M_ReadOnly'
     Attribute: Type Description = binary-time TRUE
                  -- contains date and time
     Attribute: Access Method = non-public
```

```
EN-Additional-Detail ::= NULL
EE-Additional-Detail ::= NULL
JOU-Additional-Detail ::= NULL
```

---

## 8 Standardized names

This clause specifies the standardized names defined in this standard.

### 8.1 Domain Objects – NONE

### 8.2 Program Invocation Objects – NONE

### 8.3 Named Variable Objects

There are three standardized named variable objects:

a) P_DDATE

b) P_PCSTATE

c) P_PCSTATUS

P_DDATE is a domain specific variable. It contains the date of the most recent modification of the application program that is contained in the domain. It is generated by local means.

P_PCSTATE is a bit string that contains health and state information about the PC. Subclause 5.1.4 contains a semantic description of the PC state and health. This variable is provided for those clients that wish to obtain status at the same time data is acquired. The methodology of maintaining this variable is to be specified by the implementer. Some possible methodologies include the following:

a) Periodic time basis,

b) Whenever this variable is read,

c) Once per program scan.

P_PCSTATUS is a structured variable that contains health and status information about the PC and its subsystems. This variable is provided for those clients which wish to obtain more detailed status information. The method of maintaining this variable shall be specified by the implementer in the PICS.

The formal definitions of each of these follows.

#### 8.3.1 P_DDATE

A set of domain-specific Named Variable objects shall be defined to achieve information from existing Domains in the VMD.

```
Object: Named Variable
Key Attribute: Variable Name = domain-specific {
                  domainID ,
                  itemID   "P_DDATE" }
    Attribute: Reference to Access Control List = 'M_ReadOnly'
    Attribute: Type Description = binary-time TRUE
                   -- contains date and time
    Attribute: Access Method = non-public
```

### 8.3.2 P_PCSTATE

Un objet Variable Nommée spécifique au VMD doit contenir le statut récapitulatif du PC.

```
Object: Named Variable
Key Attribute: Variable Name = vmd-specific "P_PCSTATE"
     Attribute: Reference to Access Control List = 'M_ReadOnly'
     Attribute: Type Description = bit-string 16
--        good              (0),
--        warning           (1),
--        bad               (2),
--        running           (3),
--        localControl      (4),
--        noOutputsDisabled  (5),
--        noInputsDisabled (6),
--        forced            (7),
--        appPresent        (8),
--        ioFault           (9),
--        puFault           (10),
--        powFault          (11),
--        memFault          (12),
--        comFault          (13),
--        implementerFault (14)
     Attribute: Access Method = non-public
```

La sémantique des bits de la variable P_PCSTATE résulte de l'information sur l'état de santé et les valeurs d'état des sous-systèmes du PC (voir 5.1.4):

– good – S'il est VRAI, cet attribut indique que le système PC et tous ses sous-systèmes sont en état de santé. Cet attribut est VRAI si et seulement si tous les sous-systèmes du PC indiquent un état de santé GOOD;

– warning – S'il est VRAI, cet attribut indique qu'au moins un sous-système du PC indique un état de santé WARNING et qu'aucun d'entre eux n'a un état de santé BAD;

– bad – S'il est VRAI, cet attribut indique qu'au moins un sous-système indique un état de santé BAD;

– running – S'il est VRAI, cet attribut indique qu'au moins une partie de l'application utilisateur a été chargée et qu'elle est sous le contrôle du PC;

– localControl – S'il est VRAI, cet attribut indique que la commande locale de substitution est active. Si elle l'est, la possibilité de contrôler le PC et ses sous-systèmes à partir du réseau peut être limitée. Cela peut par exemple être fortement lié à l'utilisation d'une clé de commutation locale;

– noOutputsDisabled – S'il est VRAI, cet attribut indique que le PC peut modifier l'état physique de toutes les sorties en exécutant le programme d'application ou par d'autres moyens. S'il n'est pas VRAI, l'état physique de certaines sorties n'est pas modifiable (leur état logique pouvant l'être). Cet attribut est généralement utilisé lors de la mise au point des programmes d'application des programmes d'application dans le PC;

– noInputsDisabled – S'il est VRAI, cet attribut indique que le PC peut accéder à l'état physique de toutes les entrées en exécutant le programme d'application ou par un d'autres moyens. S'il n'est pas VRAI, il n'est pas possible d'accéder à l'état physique de certaines entrées. Cet attribut est généralement utilisé lors de la mise au point des programmes d'application dans lesquels les entrées peuvent être simulées;

– forced – S'il est VRAI, cet attribut indique qu'un moins un point d'entrées-sorties ou une variable associé(e) au PC a été forcé(e). Lorsqu'une variable est forcée, le programme d'application doit utiliser la valeur spécifiée par le PADT et non la valeur provenant de l'exécution normale du programme;

– appPresent – S'il est VRAI, cet attribut indique qu'il existe au moins une application utilisateur dans le PC;

### 8.3.2 P_PCSTATE

A VMD-specific Named Variable object shall contain the summary status of the PC.

```
Object: Named Variable
Key Attribute: Variable Name = vmd-specific "P_PCSTATE"
    Attribute: Reference to Access Control List = ‘M_ReadOnly’
    Attribute: Type Description = bit-string 16
--      good              (0),
--      warning           (1),
--      bad               (2),
--      running           (3),
--      localControl      (4),
--      noOutputsDisabled (5),
--      noInputsDisabled  (6),
--      forced            (7),
--      appPresent        (8),
--      ioFault           (9),
--      puFault           (10),
--      powFault          (11),
--      memFault          (12),
--      comFault          (13),
--      implementerFault  (14)
    Attribute: Access Method = non-public
```

The semantic of the bits of the variable P_PCSTATE is derived from the health and state information of the PC subsystems, see 5.1.4:

– good – If TRUE, this attribute indicates the health of the PC system and of all its subsystems. This attribute is TRUE if and only if all subsystems of the PC indicate GOOD health condition;

– warning – If TRUE, this attribute indicates that at least one subsystem of the PC indicates a WARNING health condition and no subsystem indicates a "BAD" health condition;

– bad – If TRUE, this attribute indicates that at least one subsystem indicates a BAD health condition;

– 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;

– localControl – If TRUE, this attribute indicates that 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;

– noOutputsDisabled – 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 is not affected (logical state may be affected). This is typically used in the debugging of application programs in the PC;

– noInputsDisabled – 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 debugging of application programs where the inputs can be simulated;

– forced – If TRUE, this attribute indicates that at least one I/O point or variable associated with the PC has been forced. When a variable is forced, the application program shall use the value specified by the PADT instead of the value generated by the normal program execution;

– appPresent – If TRUE, this attribute indicates that at least one user application is present in the PC;

– puFault – S'il est VRAI, cet attribut indique un état de santé WARNING ou BAD provoqué par un sous-système d'unité de traitement;

– powFault – S'il est VRAI, cet attribut indique un état de santé WARNING ou BAD provoqué par un sous-système d'alimentation;

– memFault – S'il est VRAI, cet attribut indique un état de santé WARNING ou BAD provoqué par un sous-système de mémorisation;

– comFault – S'il est VRAI, cet attribut indique un état de santé WARNING ou BAD provoqué par un sous-système de communication;

– implementerFault – S'il est VRAI, cet attribut indique un état de santé WARNING ou BAD provoqué par un sous-système défini par le concepteur.

### 8.3.3 P_PCSTATUS

L'objet prédéfini Variable Nommée P_PCSTATUS doit contenir le récapitulatif d'état du PC et les valeurs d'état de ses sous-systèmes. Dans la spécification ci-dessous, p_NOS est le nombre de sous-systèmes.

L'objet Variable Nommée pour le Statut de l'Equipement doit avoir les attributs suivants:

```
Object: Named Variable
Key Attribute: Variable Name = vmd-specific "P_PCSTATUS"
     Attribute: Reference to Access Control List = 'M_ReadOnly'
     Attribute: Type Description = array {
                    numberOfElements    , -- p_NOS + 1
                    elementType structure {
                    components {{
                      componentName    "TYPE"
                      componentType    integer 8
                        summary                (0),
                        io                     (1),
                        pu                     (2),
                        power                  (3),
                        memory                 (4),
                        communication          (5),
                        implementer            (6)
                      },
                      {
                      componentName     "NAME",
                      componentType     visible-string
                      },
                      {
                      componentName     "STATE",
                      componentType     bit-string 16
                      },                              {
                      componentName     "SPECIFIC"
                      componentType     bit-string -- p_BIT
                      }
                    }
                  }
     Attribute: Access Method = non-public
```

L'élément ayant le numéro 0 doit contenir le statut récapitulatif, les autres éléments ayant une valeur supérieure doivent contenir l'état d'un sous-système. Le composant NAME doit contenir le nom du sous-système. Le composant TYPE doit contenir le type du sous-système. Le composant STATE doit contenir la valeur d'état du sous-système sous la forme d'une chaîne binaire dans le même ordre que celui spécifié dans chaque type de sous-système.

Les positions de bits pour chacune des valeurs d'état correspondant à chaque type de sous-système suivant sont définies ci-après:

– puFault – If TRUE, this attribute indicates WARNING or BAD health condition is caused by a Processing Unit subsystem;

– powFault – If TRUE, this attribute indicates WARNING or BAD health condition is caused by a Power Supply subsystem;

– memFault – If TRUE, this attribute indicates WARNING or BAD health condition is caused by a Memory subsystem;

– comFault – If TRUE, this attribute indicates WARNING or BAD health condition is caused by a Communication subsystem;

– implementerFault – If TRUE, this attribute indicates WARNING or BAD health condition is caused by an implementer specified subsystem.

### 8.3.3 P_PCSTATUS

The pre-defined Named Variable object P_PCSTATUS shall contain the status summary of the PC and the state information of the PC subsystems. In the specification below, p_NOS is the number of subsystems.

The Named Variable object for the Device Status shall have the following attributes:

```
Object: Named Variable
Key Attribute: Variable Name = vmd-specific "P_PCSTATUS"
    Attribute: Reference to Access Control List = 'M_ReadOnly'
    Attribute: Type Description = array {
                  numberOfElements     , -- p_NOS + 1
                  elementType structure {
                    components { {
                      componentName  "TYPE",
                      componentType  integer 8
--                      summary                (0),
--                      io                     (1),
--                      pu                     (2),
--                      power                  (3),
--                      memory                 (4),
--                      communication          (5),
--                      implementer            (6)
                      },
                      {
                      componentName  "NAME",
                      componentType  visible-string
                      },
                      {
                      componentName  "STATE",
                      componentType  bit-string 16
                      },
                      {
                      componentName  "SPECIFIC",
                      componentType  bit-string -- p_BIT
                      }
                    }
                  }
     Attribute: Access Method = non-public
```

The element with the number 0 shall contain the status summary, each element with a higher number shall contain the state of one subsystem. The component NAME shall contain the name of the subsystem. The component TYPE shall contain the type of the subsystem. The component STATE shall contain the state information of the subsystem as a bitstring in the same order as specified for each subsystem type.

The bit positions for each of the "state" values for each of the following subsystem types is given:

a) récapitulatif

```
good              (0),
warning           (1),
bad               (2),
running           (3),
localControl      (4),
noOutputsDisabled (5),
noInputsDisabled  (6),
forced            (7),
appPresent        (8),
ioFault           (9),
puFault           (10),
powFault          (11),
memFault          (12),
comFault          (13),
implementerFault  (14)
```

b) entrées-sorties

```
good              (0),
warning           (1),
bad               (2),
noOutputsDisabled (3),
noInputsDisabled  (4),
ioForced          (5)
```

c) unité de traitement

```
good              (0),
warning           (1),
bad               (2),
running           (3),
localControl      (4),
noOutputsDisabled (5),
noInputsDisabled  (6),
appPresent        (7),
forced            (8)
```

d) alimentation

```
good              (0),
warning           (1),
bad               (2),
inUse             (3),
mainsOperating    (4),
mainsLow          (5),
batteryOperating  (6),
batteryLow        (7),
protectionTripped (8),
```

e) mémoire

```
good              (0),
warning           (1),
bad               (2),
protected         (3)
```

f) communication

```
good              (0),
warning           (1),
bad               (2),
inUse             (3),
localError        (4),
remoteError       (5)
```

a) summary

```
good                (0),
warning             (1),
bad                 (2),
running             (3),
localControl        (4),
noOutputsDisabled   (5),
noInputsDisabled    (6),
forced              (7),
appPresent          (8),
ioFault             (9),
puFault             (10),
powFault            (11),
memFault            (12),
comFault            (13),
implementerFault    (14)
```

b) io

```
good                (0),
warning             (1),
bad                 (2),
noOutputsDisabled   (3),
noInputsDisabled    (4),
ioForced            (5)
```

c) pu

```
good                (0),
warning             (1),
bad                 (2),
running             (3),
localControl        (4),
noOutputsDisabled   (5),
noInputsDisabled    (6),
appPresent          (7),
forced              (8)
```

d) power

```
good                (0),
warning             (1),
bad                 (2),
inUse               (3),
mainsOperating      (4),
mainsLow            (5),
batteryOperating    (6),
batteryLow          (7),
protectionTripped   (8),
```

e) memory

```
good                (0),
warning             (1),
bad                 (2),
protected           (3)
```

f) communication

```
good                (0),
warning             (1),
bad                 (2),
inUse               (3),
localError          (4),
remoteError         (5)
```

g) concepteur

```
good                (0),
warning             (1),
bad                 (2)
```

Le concepteur doit décrire l'objet Variable Nommée dans la PICS:

a) les types de sous-systèmes pris en charge;

b) le nombre de sous-systèmes du PC (variable p_NOS).

c) la sémantique des bits dans le composant STATE pour le sous-système spécifique au concepteur;

d) la longueur de la chaîne binaire du composant SPECIFIC (variable p_BIT);

e) la sémantique du composant SPECIFIC.

### 8.4 Objets Accès Éclaté – AUCUN

### 8.5 Objets Liste des Variables Nommées – AUCUN

### 8.6 Objets Type Nommé – AUCUN

### 8.7 Objets Sémaphore – AUCUN

### 8.8 Objets Station d'opérateur – AUCUN

### 8.9 Objets Condition Événementielle – AUCUN

### 8.10 Objets Action Evénementielle – AUCUN

### 8.11 Objets Enveloppe Evénementielle – AUCUN

### 8.12 Objets Journal – AUCUN

### 8.13 Objets spécifiques à l'application – AUCUN

## 9 Conformité

### 9.1 Descriptions des classes de conformité

Les classes de conformité de cette norme d'accompagnement sont conçues pour prendre en charge différents types de PC, à partir de petits systèmes dotés de quelques fonctions de communication jusqu'à des systèmes plus complexes qui comptent de nombreuses fonctions de communication.

Les classes de conformité suivantes sont définies pour la norme d'accompagnement des Automates Programmables. Les classes de conformité sont hiérarchiques, ce qui signifie que chaque classe comprend tous les services de la classe immédiatement inférieure. Un aperçu des classes de conformité est donné dans le tableau 9 qui indique également le type de fonction PC que cette classe apporte.

**Tableau 9 – Classes de conformité CSPC**

| Classe de conformité PC | Fonctions PC ajoutées à cette classe de conformité |
|---|---|
| 1 | Acquisition de données, commandes paramétrées |
| 2 | Gestion de programmes |
| 3 | Acquisition de données non sollicitées |
| 4 | Acquisition de données configurées, alarmes et sémaphores |
| 5 | Instanciation d'objets MMS distants |

g) implementer

```
good                (0),
warning             (1),
bad                 (2)
```

The implementer shall describe for this Named Variable object in the PICS:

a) the supported subsystem types;
b) the number of the subsystems of the PC (variable p_NOS);
c) the semantic of the bits in the component STATE for the implementer-specific subsystem;
d) the length of the bitstring of the component SPECIFIC (variable p_BIT);
e) the semantic of the component SPECIFIC.

### 8.4 Scattered Access Objects – NONE

### 8.5 Named Variable List Objects – NONE

### 8.6 Named Type Objects – NONE

### 8.7 Semaphore Objects – NONE

### 8.8 Operator Station Objects – NONE

### 8.9 Event Condition Objects – NONE

### 8.10 Event Action Objects – NONE

### 8.11 Event Enrollment Objects – NONE

### 8.12 Journal Objects – NONE

### 8.13 Application-specific Objects – NONE

## 9 Conformance

### 9.1 Conformance Class Descriptions

The conformance classes of this Companion Standard are designed to support different types of PCs, beginning from small ones with only a few communication functions up to complex ones which may support many communication functions.

The following conformance classes are defined for the Companion Standard for Programmable Controllers. The conformance classes are hierarchical; that is, each class includes all of the services of the lower numbered class. An overview of the conformance classes is contained in table 9 indicating the type of PC function added by that class.

**Table 9 – CSPC Conformance Classes**

| PC Conformance Class | Added PC Functions to that PC Conformance Class |
|---|---|
| 1 | Data Acquisition, Parametric Control |
| 2 | Program Management |
| 3 | Unsolicited Data Acquisition |
| 4 | Configured Data Acquisition, Alarming, Semaphores |
| 5 | Remote MMS Object Instantiation |

Les services MMS requis, tels qu'ils ont été définis dans l'ISO/CEI 9506, sont donnés pour chacune des classes de conformité PC dans le tableau 10. Il en est de même pour les paramètres MMS, décrits dans l'ISO/CEI 9506, qui sont attribués à chaque classe de conformité PC dans le tableau 11. Les noms prédéfinis requis sont définis dans le tableau 13.

Dans les services décrits ci-après, le terme «Serveur» indique que le rôle d'appelant est pris en charge et le terme «Client» indique que le rôle d'appelé est pris en charge. Pour tous les autres services, les termes «Serveur» indiquent que le rôle d'appelé est pris en charge et «Client» indique que le rôle d'appelant est pris en charge.

a) DownloadSegment

b) TerminateDownloadSegment

c) RequestDomainDownload

d) RequestDomainUpload

e) UnsolicitedStatus

f) InformationReport

g) EventNotification

Le «S» dans les tableaux ci-après indique que le service est requis par le PC en tant que Serveur. Un «C» indique qu'il est requis en tant que Client. Le «X» indique qu'il est requis en tant que Serveur et en tant que Client. Un numéro indique que des informations complémentaires sont données à la suite du tableau dans une note précédée du même numéro.

### 9.1.1 Blocs élémentaires de conformité des services

Le tableau 10 résume les services MMS requis dans chaque classe de conformité de PC.

**Tableau 10 – Blocs élémentaires de conformité des services**

| Service | Paragraphe ISO/CEI 9506-1 | CBB | Classe de conformité PC | | | | |
|---|---|---|---|---|---|---|---|
| | | Bit | 1 | 2 | 3 | 4 | 5 |
| Initiate | 8.2 | | S | S | X | X | X |
| Conclude | 8.3 | 83 | S | S | X | X | X |
| Cancel | 8.5 | 84 | | | X | X | X |
| Status | 9.2 | 0 | S | S | X | X | X |
| UnsolicitedStatus | 9.3 | 78 | | | X | X | X |
| GetNameList | 9.4 | 1 | | S | S | S | S |
| Identify | 9.5 | 2 | S | S | S | S | S |
| Rename | 9.6 | 3 | | | | | |
| GetCapabilityList | 9.7 | 71 | | | | | |
| InitiateDownloadSequence | 10.2 | 26 | | S | S | S | S |
| DownloadSegment | 10.3 | 27 | | S | S | S | S |
| TerminateDownloadSequence | 10.4 | 28 | | S | S | S | S |
| InitiateUploadSequence | 10.5 | 29 | | S | S | S | S |
| UploadSegment | 10.6 | 30 | | S | S | S | S |
| TerminateUploadSequence | 10.7 | 31 | | S | S | S | S |
| RequestDomainDownload | 10.8 | 32 | | | | | |
| RequestDomainUpload | 10.9 | 33 | | | | | |
| LoadDomainContent | 10.10 | 34 | | | | | |
| StoreDomainContent | 10.11 | 35 | | | | | |
| DeleteDomain | 10.12 | 36 | | S | S | S | S |
| GetDomainAttributes | 10.13 | 37 | | S | S | S | S |

The required MMS services, as defined in ISO/IEC 9506, are given for each PC conformance class in table 10. Similarly, the required MMS parameters, as defined in ISO/IEC 9506, are given for each PC conformance class in table 11. The required predefined names are given in table 13.

For the following services, the Server indicates that the requester role is supported and Client indicates that the responder role is supported. For all other services, the Server indicates that the responder role is supported and Client indicates that the requester role is supported.

a) DownloadSegment
b) TerminateDownloadSequence
c) RequestDomainDownload
d) RequestDomainUpload
e) UnsolicitedStatus
f) InformationReport
g) EventNotification

An "S" in each table indicates that the service is required for a PC as a Server. A "C" indicates that it is required as a Client. An "X" indicates that it is required both as a Server and a Client. A number indicates that more information can be found in the note with corresponding number below the table.

### 9.1.1 Service Conformance Building Blocks

Table 10 summarizes the MMS services which are required in each PC conformance class.

**Table 10 – Service Conformance Building Blocks**

| Service | ISO/IEC 9506-1 | CBB | PC Conformance Class | | | | |
|---|---|---|---|---|---|---|---|
| | Subclause | Bit | 1 | 2 | 3 | 4 | 5 |
| Initiate | 8.2 | | S | S | X | X | X |
| Conclude | 8.3 | 83 | S | S | X | X | X |
| Cancel | 8.5 | 84 | | | X | X | X |
| Status | 9.2 | 0 | S | S | X | X | X |
| UnsolicitedStatus | 9.3 | 78 | | | X | X | X |
| GetNameList | 9.4 | 1 | | S | S | S | S |
| Identify | 9.5 | 2 | S | S | S | S | S |
| Rename | 9.6 | 3 | | | | | |
| GetCapabilityList | 9.7 | 71 | | | | | |
| InitiateDownloadSequence | 10.2 | 26 | | S | S | S | S |
| DownloadSegment | 10.3 | 27 | | S | S | S | S |
| TerminateDownloadSequence | 10.4 | 28 | | S | S | S | S |
| InitiateUploadSequence | 10.5 | 29 | | S | S | S | S |
| UploadSegment | 10.6 | 30 | | S | S | S | S |
| TerminateUploadSequence | 10.7 | 31 | | S | S | S | S |
| RequestDomainDownload | 10.8 | 32 | | | | | |
| RequestDomainUpload | 10.9 | 33 | | | | | |
| LoadDomainContent | 10.10 | 34 | | | | | |
| StoreDomainContent | 10.11 | 35 | | | | | |
| DeleteDomain | 10.12 | 36 | | S | S | S | S |
| GetDomainAttributes | 10.13 | 37 | | S | S | S | S |

| Service | Paragraphe ISO/CEI 9506-1 | CBB | Classe de conformité PC | | | | |
|---|---|---|---|---|---|---|---|
| | | Bit | 1 | 2 | 3 | 4 | 5 |
| CreateProgramInvocation | 11.2 | 38 | | | | | S |
| DeleteProgramInvocation | 11.3 | 39 | | | | | S |
| Start | 11.4 | 40 | | S | S | S | S |
| Stop | 11.5 | 41 | | S | S | S | S |
| Resume | 11.6 | 42 | | S | S | S | S |
| Reset | 11.7 | 43 | | S | S | S | S |
| Kill | 11.8 | 44 | | | | | |
| GetProgramInvocationAttributes | 11.9 | 45 | | | | | S |
| Read | 12.6 | 4 | S | S | X | X | X |
| Write | 12.7 | 5 | S | S | X | X | X |
| InformationReport | 12.8 | 79 | | | X | X | X |
| GetVariableAccessAttributes | 12.9 | 6 | | | | | S |
| DefineNamedVariable | 12.10 | 7 | | | | | |
| DefineScatteredAccess | 12.11 | 8 | | | | | |
| GetScatteredAccessAttributes | 12.12 | 9 | | | | | |
| DeleteVariableAccess | 12.13 | 10 | | | | | |
| DefineNamedVariableList | 12.14 | 11 | | | | | S |
| GetNamedVariableListAttributes | 12.15 | 12 | | | | | S |
| DeleteNamedVariableList | 12.16 | 13 | | | | | S |
| DefineNamedType | 12.17 | 14 | | | | | |
| GetNamedTypeAttributes | 12.18 | 15 | | | | | |
| DeleteNamedType | 12.19 | 16 | | | | | |
| TakeControl | 13.2 | 19 | | | | S | S |
| RelinquishControl | 13.3 | 20 | | | | S | S |
| DefineSemaphore | 13.4 | 21 | | | | | S |
| DeleteSemaphore | 13.5 | 22 | | | | | S |
| ReportSemaphoreStatus | 13.6 | 23 | | | | S | S |
| ReportPoolSemaphoreStatus | 13.7 | 24 | | | | | |
| ReportSemaphoreEntryStatus | 13.8 | 25 | | | | S | S |
| AttachToSemaphore | 13.9 | 82 | | | | | |
| Input | 14.2 | 17 | | | | | |
| Output | 14.3 | 18 | | | | | |
| DefineEventCondition | 15.2 | 47 | | | | | S |
| DeleteEventCondition | 15.3 | 48 | | | | | S |
| GetEventconditionAttributes | 15.4 | 49 | | | | | S |
| ReportEventConditionStatus | 15.5 | 50 | | | | S | S |
| AlterEventConditionMonitoring | 15.6 | 51 | | | | S | S |
| TriggerEvent | 15.7 | 52 | | | | | |
| DefineEventAction | 15.8 | 53 | | | | | S |
| DeleteEventAction | 15.9 | 54 | | | | | S |
| GetEventActionAttributes | 15.10 | 55 | | | | | S |
| ReportEventActionStatus | 15.11 | 56 | | | | S | S |
| DefineEventEnrollment | 15.12 | 57 | | | | | S |
| DeleteEventEnrollment | 15.13 | 58 | | | | | S |
| GetEventEnrollmentAttributes | 15.14 | 61 | | | | | S |
| ReportEventEnrollmentStatus | 15.15 | 60 | | | | S | S |

| Service | ISO/IEC 9506-1 Subclause | CBB Bit | PC Conformance Class 7 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| CreateProgramInvocation | 11.2 | 38 | | | | | S |
| DeleteProgramInvocation | 11.3 | 39 | | | | | S |
| Start | 11.4 | 40 | | S | S | S | S |
| Stop | 11.5 | 41 | | S | S | S | S |
| Resume | 11.6 | 42 | | S | S | S | S |
| Reset | 11.7 | 43 | | S | S | S | S |
| Kill | 11.8 | 44 | | | | | |
| GetProgramInvocationAttributes | 11.9 | 45 | | | | | S |
| Read | 12.6 | 4 | S | S | X | X | X |
| Write | 12.7 | 5 | S | S | X | X | X |
| InformationReport | 12.8 | 79 | | | X | X | X |
| GetVariableAccessAttributes | 12.9 | 6 | | | | | S |
| DefineNamedVariable | 12.10 | 7 | | | | | |
| DefineScatteredAccess | 12.11 | 8 | | | | | |
| GetScatteredAccessAttributes | 12.12 | 9 | | | | | |
| DeleteVariableAccess | 12.13 | 10 | | | | | |
| DefineNamedVariableList | 12.14 | 11 | | | | | S |
| GetNamedVariableListAttributes | 12.15 | 12 | | | | | S |
| DeleteNamedVariableList | 12.16 | 13 | | | | | S |
| DefineNamedType | 12.17 | 14 | | | | | |
| GetNamedTypeAttributes | 12.18 | 15 | | | | | |
| DeleteNamedType | 12.19 | 16 | | | | | |
| TakeControl | 13.2 | 19 | | | | S | S |
| RelinquishControl | 13.3 | 20 | | | | S | S |
| DefineSemaphore | 13.4 | 21 | | | | | S |
| DeleteSemaphore | 13.5 | 22 | | | | | S |
| ReportSemaphoreStatus | 13.6 | 23 | | | | S | S |
| ReportPoolSemaphoreStatus | 13.7 | 24 | | | | | |
| ReportSemaphoreEntryStatus | 13.8 | 25 | | | | S | S |
| AttachToSemaphore | 13.9 | 82 | | | | | |
| Input | 14.2 | 17 | | | | | |
| Output | 14.3 | 18 | | | | | |
| DefineEventCondition | 15.2 | 47 | | | | | S |
| DeleteEventCondition | 15.3 | 48 | | | | | S |
| GetEventConditionAttributes | 15.4 | 49 | | | | | S |
| ReportEventConditionStatus | 15.5 | 50 | | | | S | S |
| AlterEventConditionMonitoring | 15.6 | 51 | | | | S | S |
| TriggerEvent | 15.7 | 52 | | | | | |
| DefineEventAction | 15.8 | 53 | | | | | S |
| DeleteEventAction | 15.9 | 54 | | | | | S |
| GetEventActionAttributes | 15.10 | 55 | | | | | S |
| ReportEventActionStatus | 15.11 | 56 | | | | S | S |
| DefineEventEnrollment | 15.12 | 57 | | | | | S |
| DeleteEventEnrollment | 15.13 | 58 | | | | | S |
| GetEventEnrollmentAttributes | 15.14 | 61 | | | | | S |
| ReportEventEnrollmentStatus | 15.15 | 60 | | | | S | S |

| Service | Paragraphe ISO/CEI 9506-1 | CBB | Classe de conformité PC | | | | |
|---|---|---|---|---|---|---|---|
| | | Bit | 1 | 2 | 3 | 4 | 5 |
| AlterEventEnrollment | 15.16 | 59 | | | | | S |
| EventNotification | 15.17 | 80 | | | | S | S |
| AcknowledgeEventNotification | 15.18 | 62 | | | | S | S |
| GetAlarmSummary | 15.19 | 63 | | | | S | S |
| GetAlarmEnrollmentSummary | 15.20 | 64 | | | | | S |
| AttachtoEventCondition | 15.21 | 81 | | | | | S |
| ReadJournal | 16.2 | 65 | | | | | |
| WriteJournal | 16.3 | 66 | | | | | |
| InitializeJournal | 16.4 | 67 | | | | | |
| ReportJournalStatus | 16.5 | 68 | | | | | |
| CreateJournal | 16.6 | 69 | | | | | |
| DeleteJournal | 16.7 | 70 | | | | | |
| ObtainFile | B.1 | 46 | | | | | |
| FileOpen | C.3 | 72 | | | | | |
| FileRead | C.4 | 73 | | | | | |
| FileClose | C.5 | 74 | | | | | |
| FileRename | C.6 | 75 | | | | | |
| FileDelete | C.7 | 76 | | | | | |
| FileDirectory | C.8 | 77 | | | | | |
| ExchangeData | 20.2, AM1 | 86 | | | | | X |
| GetDataExchangeAttributes | 20.3, AM1 | 85 | | | | | S |

### 9.1.2 Blocs élémentaires de conformité des paramètres

Le tableau 11 décrit les blocs élémentaires de conformité des paramètres. Un «X» indique que le paramètre est requis.

**Tableau 11 – Blocs élémentaires de conformité des paramètres**

| Paramètre CBB | CBB | Classe de conformité de PC | | | | |
|---|---|---|---|---|---|---|
| | Bit | 1 | 2 | 3 | 4 | 5 |
| STR1 | 0 | | | X | X | X |
| STR2 | 1 | | | X | X | X |
| VNAM | 2 | | | X | X | X |
| VADR | 4 | 1) | 1) | 1) | 1) | 1) |
| VALT | 3 | | | | X | X |
| VSCA | 5 | | | | | |
| TPY | 6 | | | | | |
| VLIS | 7 | | | | | |
| REAL | 8 | 2) | 2) | 2) | 2) | 2) |
| AKEC | 9 | | | | | |
| CEI | 10 | | | | | |

NOTE – Le concepteur doit spécifier lequel, du VNAM ou du VADR, ou encore si les deux à la fois, sont supportés dans les classes de conformité 1 et 2.

1) Si le concepteur supporte les variables directement représentées qui sont visibles par la MMS, VADR est obligatoire avec l'Adresse Symbolique.

2) Si le concepteur supporte des variables de type Réel, ce type de données doit être visible par la MMS.

| Service | ISO/IEC 9506-1 Subclause | CBB Bit | PC Conformance Class 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| AlterEventEnrollment | 15.16 | 59 | | | | | S |
| EventNotification | 15.17 | 80 | | | | S | S |
| AcknowledgeEventNotification | 15.18 | 62 | | | | S | S |
| GetAlarmSummary | 15.19 | 63 | | | | S | S |
| GetAlarmEnrollmentSummary | 15.20 | 64 | | | | | S |
| AttachtoEventCondition | 15.21 | 81 | | | | | S |
| ReadJournal | 16.2 | 65 | | | | | |
| WriteJournal | 16.3 | 66 | | | | | |
| InitializeJournal | 16.4 | 67 | | | | | |
| ReportJournalStatus | 16.5 | 68 | | | | | |
| CreateJournal | 16.6 | 69 | | | | | |
| DeleteJournal | 16.7 | 70 | | | | | |
| ObtainFile | B.1 | 46 | | | | | |
| FileOpen | C.3 | 72 | | | | | |
| FileRead | C.4 | 73 | | | | | |
| FileClose | C.5 | 74 | | | | | |
| FileRename | C.6 | 75 | | | | | |
| FileDelete | C.7 | 76 | | | | | |
| FileDirectory | C.8 | 77 | | | | | |
| ExchangeData | 20.2, AM1 | 86 | | | | | X |
| GetDataExchangeAttributes | 20.3, AM1 | 85 | | | | | S |

### 9.1.2 Parameter Conformance Building Blocks

Table 11 describes the parameter conformance building blocks. An "X" indicates that it is required.

**Table 11 – Parameter Conformance Building Blocks**

| Parameter CBB | CBB Bit | PC Conformance Class 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| STR1 | 0 | | | X | X | X |
| STR2 | 1 | | | X | X | X |
| VNAM | 2 | | | X | X | X |
| VADR | 4 | 1) | 1) | 1) | 1) | 1) |
| VALT | 3 | | | | X | X |
| VSCA | 5 | | | | | |
| TPY | 6 | | | | | |
| VLIS | 7 | | | | | |
| REAL | 8 | 2) | 2) | 2) | 2) | 2) |
| AKEC | 9 | | | | | |
| CEI | 10 | | | | | |

NOTE – The implementer shall specify which of VNAM or VADR or both is supported in Conformance Class 1 and 2.

1) If the implementer supports directly represented variables which are visible to MMS, VADR is required with Symbolic Address.

2) If the implementer supports variables with data type Real, the data type REAL shall be visible via MMS.

### 9.1.3 Autres paramètres spécifiés à l'initialisation

Le tableau 12 décrit les prescriptions concernant les autres paramètres spécifiés à l'initialisation.

**Tableau 12 – Autres paramètres spécifiés de déclenchement**

| Paramètre | Classe de conformité du PC | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Nest | 0 | 0 | 1 | 2 | 2 |

Le numéro en regard de la cellule NEST indique le niveau d'imbrication minimal requis.

## 9.2 Restrictions aux paramètres optionnels de MMS – AUCUNE

## 9.3 Conformité aux objets normalisés

Le tableau 13 indique les conditions de conformité pour les objets dont le nom est normalisé dans la présente norme d'accompagnement.

**Tableau 13 – Conformité aux objets nommés normalisés**

| Nom | Classe de conformité du PC | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| P_PCSTATUS | | | | S | S |
| P_PCSTATE | [1] | [1] | S | S | S |
| P_DDATE | | | S | S | S |
| [1] Le support est obligatoire seulement si VNAM est aussi supporté. | | | | | |

## 9.4 Additions à la PICS MMS

La déclaration pro forma de conformité de réalisation (PICS) suivante est définie pour les Services de Messageries des Automates Programmables. Le concepteur doit spécifier les éléments suivants pour toutes les classes de conformité.

NOTE – Ces éléments s'ajoutent aux éléments spécifiés en 18.5 de l'ISO/CEI 9506-2.

### 9.1.3 Other Initiate Specified Parameters

Table 12 describes requirements on other Initiate specified parameters.

**Table 12 – Other Initiate Specified Parameters**

| Parameter | PC Conformance Class | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Nest | 0 | 0 | 1 | 2 | 2 |

The number next to "NEST" is the minimum required nesting level.

## 9.2 Restrictions on MMS Optional Parameters – NONE

## 9.3 Conformance to Standardized Objects

Table 13 indicates the conformance requirements for the standardized named objects of this companion standard.

**Table 13 – Conformance to Standardized Named Objects**

| Name | PC Conformance Class | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| P_PCSTATUS | | | | S | S |
| P_PCSTATE | [1] | [1] | S | S | S |
| P_DDATE | | | S | S | S |
| [1] Support is required only if VNAM is also supported. | | | | | |

## 9.4 Additions to the MMS PICS

The following Protocol Implementation Conformance Statement (PICS) is defined for Programmable Controller Messaging Services. The implementer shall specify the following items for all conformance classes.

NOTE – These items are in addition to the items specified in 18.5 of ISO/IEC 9506-2.

**Tableau 14 – PICS pour toutes les classes de conformité**

| | | |
|---|---|---|
| 1 | Le type et le numéro de chaque type de sous-système qui fournit un statut spécifique à l'implémentation | 5.1.4<br>8.3.3 |
| 2 | Pour chaque sous-système: définition de l'anomalie grave et de l'anomalie mineure | 5.1.4 |
| 3 | Pour chaque sous-système: sémantique des états définis par le concepteur et la longueur de la CHAÎNE BINAIRE | 5.1.4<br>8.3.3 |
| 4 | Les conditions indiquées par les valeurs d'état de santé de l'Unité de Traitement | 5.1.4.2.2 |
| 5 | Pour chaque sous-système d'interface de communication, spécification du nombre d'erreurs «acceptable» | 5.1.4.5.2 |
| 6 | Correspondance des sous-systèmes spécifiques au concepteur avec la réalisation | 5.1.4.6 |
| 7 | Les valeurs d'I/O State admises pour chaque service d'invocation de programme | 5.2.7.1<br>6.3.3 |
| 8 | La définition de la valeur d'I/O State spécifiée par le concepteur | 5.2.7.1<br>6.3.3 |
| 9 | Le mécanisme de la valeur d'I/O State spécifiée par l'utilisateur | 5.2.7.1<br>6.3.3 |
| 10 | Ce que d'autres Clients peuvent faire sur le PC lorsqu'un Client est en train de le télécharger | 5.2.7.2 |
| 11 | La correspondance des objets de la norme d'accompagnement du PC avec la réalisation | 6 |
| 12 | Autres objets prédéfinis | 6 |
| 13 | La correspondance d'un VMD avec la réalisation (1 vers 1, 1 vers n, etc.) | 6 |
| 14 | Syntaxe et sémantique des attributs de domaines ListOfCapabilities | 6.2 |
| 15 | La prise en charge et les règles en relation avec les attributs Indépendants et Référence d'invocation de programme des Invocations de programme | 6.3.1 |
| 16 | Syntaxe et sémantique des variables représentées directement | 6.4.1 |
| 17 | Restriction d'accès aux variables non nommées | 6.4.1 |
| 18 | Méthode d'accès des variables nommées prédéfinies | 6.4.1 |
| 19 | Correspondance des valeurs de santé et d'état spécifiées dans le récapitulatif d'états du PC avec la réalisation | 8.3.2 |
| 20 | Mécanisme de mise à jour des variables nommées P_PCSTATE et P_PCSTATUS | 8.3.2 |
| 21 | Classe de conformité prise en charge | 9.1 |

**Table 14 – PICS for all conformance classes**

| | | |
|---|---|---|
| 1 | The type and number of each type of subsystem which provides status in the implementation | 5.1.4<br>8.3.3 |
| 2 | For each subsystem: definition of Major and Minor fault | 5.1.4 |
| 3 | For each subsystem: semantics of implementer specified states and the length of the BIT STRING | 5.1.4<br>8.3.3 |
| 4 | The conditions indicated by Processing Unit health values | 5.1.4.2.2 |
| 5 | For each communication interface subsystem, specify the number of errors which is “acceptable” | 5.1.4.5.2 |
| 6 | Mapping of the implementer specific subsystems onto the implementation | 5.1.4.6 |
| 7 | The supported values of I/O State for each Program Invocation service | 5.2.7.1<br>6.3.3 |
| 8 | The definition of Implementer Specified value of I/O State | 5.2.7.1<br>6.3.3 |
| 9 | The mechanism for User Specified value of I/O State | 5.2.7.1<br>6.3.3 |
| 10 | What other Clients can do to the PC while one Client is downloading it | 5.2.7.2 |
| 11 | The mapping of the PC Companion Standard Objects onto the implementation | 6 |
| 12 | Additional predefined objects | 6 |
| 13 | The mapping of VMD onto the implementation (1 to 1, 1 to many, etc.) | 6 |
| 14 | The syntax and semantics of ListOfCapabilities attributes of Domains | 6.2 |
| 15 | The support of, and rules related to the Independent and Program Invocation Reference attributes of Program Invocations | 6.3.1 |
| 16 | Syntax and semantics of the directly represented variables | 6.4.1 |
| 17 | Restrictions for accessing unnamed variables | 6.4.1 |
| 18 | Access Method for predefined named variables | 6.4.1 |
| 19 | The mapping of health and state specified in the PC summary status onto the implementation | 8.3.2 |
| 20 | The mechanism for maintaining P_PCSTATE and P_PCSTATUS named variables | 8.3.2 |
| 21 | The supported conformance class | 9.1 |

# Annexe A
(informative)

# Exemples

Cette annexe vise à donner quelques exemples d'utilisation des classes de conformité CSPC avec les messages MMS réels.

Les messages MMS figurant dans cette annexe utilisent la «notation normalisée pour les valeurs de données», comme dans l'annexe B (ISO/CEI 8825) de l'ASN.1, règles élémentaires de codage (mentionnées sous le nom de «Notation Normalisée»). Toutes les valeurs utilisées dans la Notation Normalisée sont des décimaux, à l'exception des chaînes de format 'xxxx'H qui sont des hexadécimaux.

## A.1 Equipement de surveillance simple

Cet exemple décrit un PC simple qui ne prend en charge que la classe de conformité CSPC 1. Le PC est utilisé pour compter les pièces qui arrivent sur deux chaînes de montage distinctes.

Un mécanisme local permet au programme d'application de définir le nom des variables visibles par MMS. Ce mécanisme local est utilisé pour définir les variables «CB1COUNT» et «CB2COUNT», qui sont respectivement le nombre de pièces d'assemblage sur les chaînes de montage 1 et 2.

### A.1.1 Le Contrôleur de Cellules établit une association

Afin d'établir une association avec le PC, le Contrôleur de Cellules envoie une PDU de demande d'initialisation dans le champ Données Utilisateur de l'appel de demande A_ASSOCIATE. Cette demande d'initialisation omet les champs optionnels «localDetailCalling» et «proposedDataStructureNestingLevel», et prend leur valeur par défaut. Le paramètre «serviceSupportedCalling» affiche que l'utilisateur MMS dans le Contrôleur de Cellules peut prendre en charge les indications de services suivants:

a) Download Segment;

b) Request Domain Download;

c) Request Domain Upload;

d) Unsolicited Status;

e) Conclude;

f) Exchange Data.

Notation normalisée:

```
{initiate-RequestPDU
  {proposedMaxServOutstandingCalling        1,
   proposedMaxServOutstandingCalled         1,
   initRequestDetail
  {proposedVersionNumber                    {0},
   proposedParameterCBB                     'E800'H
   serviceSupportedCalling          '00000010C0000000000212'H
  }
 }
}
```

# Annex A
(informative)

# Examples

The purpose of this annex is to give some examples of the use of CSPC conformance classes with actual MMS messages.

The MMS messages shown in this annex are shown in the "standard notation for data values" as used in Annex B (ISO/IEC 8825) of the ASN.1, basic encoding rules (referred to as "Standard Notation"). All values used in Standard Notation are decimal, except for strings of the form 'xxxx'H which indicate hexadecimal.

## A.1 Simple Monitoring Device

This example describes a simple PC that only supports CSPC conformance class 1. This PC is used to count parts that go by on two distinct conveyer belts.

There is a local mechanism that allows the application program to define variable names that can be visible via MMS. This local mechanism has been used to define the variables "CB1COUNT" and "CB2COUNT", which represent the assembly part counts for conveyer belt 1 and conveyer belt 2, respectively.

### A.1.1 Cell Controller Establishes an Association

In order to establish an association with the PC, the Cell Controller sends an Initiate request PDU in the User Data field of an A_ASSOCIATE request call. This Initiate request omits the optional fields "localDetailCalling" and "proposedDataStructureNestingLevel", allowing them to default. Also, the "serviceSupportedCalling" parameter is showing that MMS user in the Cell Controller can support the following service indications:

a) Download Segment;

b) Request Domain Download;

c) Request Domain Upload;

d) Unsolicited Status;

e) Conclude;

f) Exchange Data.

Standard Notation:

```
{initiate-RequestPDU
 {proposedMaxServOutstandingCalling     1,
  proposedMaxServOutstandingCalled      1,
  initRequestDetail
  {proposedVersionNumber                {0},
   proposedParameterCBB                 'E800'H
   servicesSupportedCalling       '00000010C0000000000212'H
  }
 }
}
```

Le PC obtient l'indication d'initialisation dans le champ Données Utilisateur de l'appel d'indication A_ASSOCIATE. Le fournisseur MMS dans le PC détermine le paramètre «serviceSupportedCalling» en fonction des services qui peuvent être pris en charge à la fois par l'utilisateur MMS dans le Contrôleur de Cellules et le fournisseur MMS dans le PC. Supposons que le fournisseur MMS sur le PC peut envoyer les PDU de demande de service Unsolicitated status, Information Report et Conclude; le paramètre «serviceSupportedCalling» dans l'indication de déclenchement prend alors la valeur suivante:

```
'00000000000000000210'H
```

Elle signifie que l'utilisateur MMS sur le PC peut envoyer les PDU de demande de service Unsolicitated status et Conclude.

Le PC accepte l'association, avec une PDU de réponse d'initialisation qui est contenue dans le champ Données Utilisateur d'une réponse A_ASSOCIATE. La réponse Déclenchement omet les champs optionnels «localDetailCalled» et «negotiatedDataStructureNestingLevel», en leur attribuant leur valeur par défaut. Le paramètre «serviceSupportedCalled» affiche que l'utilisateur MMS (VMD) sur le PC peut admettre la classe de conformité CSPC 1.

Notation normalisée:

```
{initiate-ResponsePDU
  {negotiatedMaxServOutstandingCalling 1,
   negotiatedMaxServOutstandingCalled  1,
   InitResponseDetail
   {negotiatedVersionNumber            {0},
    negotiatedParameterCBB             '2800'H
    servicesSupportedCalled            'AC00000000000000000010'H
  }
 }
}
```

Le Contrôleur de Cellules obtient la confirmation d'initialisation dans le champ Données Utilisateur de l'appel de confirmation A_ASSOCIATE. Le fournisseur MMS sur le Contrôleur de Cellules définit le paramètre «serviceSupportedCalled» en fonction des services qui peuvent être supportés à la fois par l'utilisateur MMS sur le PC (VMD) et par le fournisseur MMS sur le Contrôleur de Cellules. Supposons que le fournisseur MMS sur le Contrôleur de Cellules envoie une PDU de demande de services entrant dans la classe de conformité CSPC 4, le paramètre «serviceSupportedCalling» dans l'indication de déclenchement devient alors le suivant:

```
'AC00000000000000000010'H
```

Cette valeur signifie que l'utilisateur MMS sur le PC peut envoyer une PDU de demande de service entrant dans la classe de conformité CSPC 1.

### A.1.2 Le Contrôleur de Cellules obtient l'état du PC

Afin de s'assurer que le PC est en état de fonctionnement avant de lancer un projet important, le Contrôleur de Cellules obtient l'état du PC. Pour ce faire, la variable nommée normalisée «P_PCSTATE» est lue.

The PC gets the Initiate indication in the User Data field of the A_ASSOCIATE indication call. The MMS provider in the PC determines the "serviceSupportedCalling" parameter according to the services which can be supported both by the MMS user in the Cell Controller and the MMS provider in the PC. Suppose that the MMS provider in the PC can send the Unsolicited status, Information Report and Conclude service request PDU, then "serviceSupportedCalling" parameter in the Initiate indication becomes

```
'00000000000000000210'H
```

This means MMS user in the PC can send Unsolicited status and Conclude service request PDU.

The PC accepts the association, with an Initiate response PDU that is carried in the User Data field of an A_ASSOCIATE response. This Initiate response omits the optional fields "localDetailCalled" and "negotiatedDataStructureNestingLevel", allowing them to default. Also, the "serviceSupportedCalled" parameter is showing that MMS user(VMD) in the PC can support the CSPC conformance class 1.

Standard Notation:

```
{initiate-ResponsePDU
 {negotiatedMaxServOutstandingCalling  1,
  negotiatedMaxServOutstandingCalled   1,
  InitResponseDetail
  {negotiatedVersionNumber              {0},
   negotiatedParameterCBB               '2800'H
   servicesSupportedCalled              'AC000000000000000000010'H
  }
 }
}
```

The Cell Controller gets the Initiate confirm in the User Data field of the A_ASSOCIATE confirm call. The MMS provider in the Cell Controller determines the "serviceSupportedCalled" parameter according to the services which can be supported both by the MMS user in the PC (VMD) and the MMS provider in the Cell Controller. Suppose that the MMS provider in the Cell controller can send the services request PDU which conform to CSPC conformance class 4, then "serviceSupportedCalling" parameter in the Initiate indication becomes

```
'AC000000000000000000010'H
```

This means MMS user in the PC can send service request PDU which conform to CSPC conformance class 1.

### A.1.2 Cell Controller Gets Status of the PC

In order to make sure that the PC is in good working order before a major project is begun, the Cell Controller gets the status of PC. This is performed by reading the standardized named variable "P_PCSTATE".

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                  17,
  confirmedServiceRequest
  {read
   {specificationWithResult FALSE
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name       {vmd-specific  "P_PCSTATE"}}
     }
    }
   }
  }
 }
}
```

Ayant reçu cette demande, le PC fournit les valeurs courantes de P_PCSTATE. La valeur de P_PCSTATE est «good», «running»,« noOutputDisabled» et «noInputDisabled».

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                  17,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success  {bitstring  '9600'H}}
   }
  }
 }
}
```

### A.1.3 Le Contrôleur de Cellules écrit des variables

Comme il a été précisé dans le début de l'exemple, le Contrôleur de Cellules sait que les variables «CB1COUNT» et «CB2COUNT» contiennent les nombres sous forme d'entiers qu'il souhaite lire du PC. Le Contrôleur de Cellules connaît les types de données de ces variables prédéfinies.

Le Contrôleur de Cellules souhaite initialiser les compteurs à zéro pour être sûr que seules les pièces de la période en cours seront enregistrées. Pour ce faire, le Contrôleur de Cellules envoie une requête d'écriture pour chaque variable de comptage.

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                  18,
  confirmedServiceRequest
  {write
   variableAccessSpecification
   listOfVariable
 {variableSpecification
   {name {vmd-specific  "CB1COUNT"} }
   }
   {variableSpecification
    {name  {vmd-specific  "CB2COUNT"} }
   }
  }
  listOfData
```

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                    17,
  confirmedServiceRequest
  {read
   {specificationWithResult  FALSE
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name   {vmd-specific  "P_PCSTATE"}}
     }
    }
   }
  }
 }
}
```

With the request, the PC provides the current values of P_PCSTATE. The value of P_PCSTATE represents "good", "running", "noOutputDisabled" and "noInputDisabled".

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                    17,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success              {bitstring  '9600'H}}
   }
  }
 }
}
```

### A.1.3 Cell Controller Writes to Variables

As stated at the beginning of this example, the Cell Controller knows that variables CB1COUNT and CB2COUNT contain the integer counts it wants to see from this PC. The Cell Controller knows data types of these predefined variables.

The Cell Controller wishes to set the counters to zero to make sure that only parts from the current time on will be registered. To do this the Cell Controller sends a Write request for each counter variable.

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                    18,
  confirmedServiceRequest
  {write
   {variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name  {vmd-specific  "CB1COUNT"} }
     }
     {variableSpecification
      {name  {vmd-specific  "CB2COUNT"} }
     }
    }
    listOfData
```

```
    { {00}, {00} }
   }
  }
 }
}
```

Le PC indique au Contrôleur de Cellules qu'il a bien exécuté la requête d'écriture.

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                 18,
  confirmedServiceResponse
  {write
   {success,
    success }
  }
 }
}
```

### A.1.4 Le Contrôleur de Cellules lit les variables

Tout au long de cet important projet surveillé par le Contrôleur de Cellules de façon régulière, le Contrôleur de Cellules va vérifier les deux variables de comptage depuis le PC.

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                 19,
  confirmedServiceRequest
  {read
   {SpecificationWithResult FALSE,
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name  {vmd-specific "CB1COUNT"} }
     }
     {variableSpecification
      {name  {vmd-specific "CB2COUNT"} }
     }
    }
   }
  }
 }
}
```

A chaque requête, le PC fournit les valeurs courantes de CB1COUNT (dénommée «xx» dans l'exemple) et de CB2COUNT (dénommé ici «yyyy»).

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                 19,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success     {integer xx} },
    {success     {integer yyyy} } }
  }
 }
}
```

```
      { {00}, {00} }
    }
   }
  }
}
```

The PC indicates to the Cell Controller that the Write request was honored.

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                    18,
  confirmedServiceResponse
  {write
   {success,
    success }
  }
 }
}
```

### A.1.4 Cell Controller Reads From Variables

Periodically throughout the major project being supervised by the Cell Controller, the Cell Controller will read the two counter variables from the PC.

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                    19,
  confirmedServiceRequest
  {read
   {SpecificationWithResult   FALSE,
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name  {vmd-specific "CB1COUNT"} }
     }
     {variableSpecification
      {name  {vmd-specific "CB2COUNT"} }
     }
    }
   }
  }
 }
}
```

With each request, the PC provides the current values of CB1COUNT (here represented as "xx") and CB2COUNT (here represented as "yyyy").

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                    19,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success  {integer xx} },
    {success  {integer yyyy} } }
  }
 }
}
```

## A.2 Un équipement «chargement et exécution»

Cet exemple décrit un PC relativement simple qui prend en charge la classe de conformité CSPC 2. Ce PC est utilisé pour contrôler plusieurs équipements servant à assembler des montres-bracelets qui sont constituées d'un système d'entraînement des aiguilles, d'un cadran, d'aiguilles et d'un boîtier. L'information d'assemblage de la montre à transmettre au PC doit être téléchargée dans un domaine. La différentiation de produits est téléchargée en tant que domaine complet. La différentiation de produits est, par exemple, la différence entre une montre de sport et une montre de collection. De plus, le PC dispose d'une invocation de programme qui correspond au domaine complet, et la création, le déclenchement ou l'interruption de l'invocation de programme est exécutée sur la demande du Contrôleur de Cellules.

### A.2.1 Le Contrôleur de Cellules établit une association

Afin d'établir une association avec le PC, le Contrôleur de Cellules envoie une PDU de demande d'initialisation dans le champ de données utilisateur d'un appel de requête A_ASSOCIATE. Cette requête d'initialisation omet les champs optionnels «localDetailCalling» et «proposedDataStructureNestingLevel» et prend leur valeur par défaut.

Notation normalisée:

```
{initiate-RequestPDU
 {proposedMaxServOutstandingCalling   1,
  proposedMAxServOutstandingCalled    1,
  initRequestDetail
  {proposedVersionNumber              {0},
   proposedParameterCBB               ‘2800’H,
   servicesSupportedCalling     ‘0000010C00000000000212’H,
  }
 }
}
```

Le PC obtient l'indication d'initialisation dans le champ Données Utilisateur de l'appel d'indication A_ASSOCIATE.

Le PC accepte l'association, à l'aide de la PDU réponse d'initialisation qui est contenue dans le champ Données Utilisateur d'une réponse A_ASSOCIATE. La réponse d'initialisation omet les champs optionnels «localDetailCalled» et «negotiateDataStructureNestingLevel» et prend leur valeur par défaut. Le paramètre «serviceSupportedCalled» affiche que l'utilisateur MMS (VMD) sur le PC peut admettre la classe de conformité CSPC 2.

Notation normalisée:

```
{initiate-ResponsePDU
 {negotiatedMaxServOutstandingCalling 1,
  negotiatedMaxServOutstandingCalled  1,
  initResponseDetail
  {negotiatedVersionNumber            {0},
   negotiatedParameterCBB             ‘2800’H,
   servicesSupportedCalled    ‘EC00003F0FF40000000010’H,
  }
 }
}
```

Le Contrôleur de Cellules obtient la confirmation d'initialisation dans le champ Données Utilisateur de l'appel de confirmation A_ASSOCIATE. Le fournisseur MMS sur le Contrôleur de Cellules définit le paramètre «serviceSupportedCalled» en fonction des services qui peuvent être pris en charge à la fois par l'utilisateur MMS sur le PC (VMD) et le fournisseur MMS sur le

## A.2 A "Load and Go" Device

This example describes a relatively simple PC that supports CSPC conformance class 2. This PC is used to control various devices used to assemble watches which consist of hands-driving unit, dial plate, hands and outer frame. The watch assembling information to be given to the PC will be downloaded into a domain. The product differentiation is downloaded as an entire domain. Product differentiation is, for example, the difference between sporty watch and old fashioned watch. In addition, the PC has one Program Invocation which corresponds to the entire domain, and the creation, starting or stopping of Program Invocation is performed by the request of the Cell controller.

### A.2.1 Cell Controller Establishes An Association

In order to establish as association with the PC, the Cell Controller sends an Initiate request PDU in the user data field of an A_ASSOCIATE request call. This Initiate request omits the optional fields "localDetailCalling" and "proposedDataStructureNestingLevel", allowing them to default.

Standard Notation:

```
{initiate-RequestPDU
 {proposedMaxServOutstandingCalling      1,
  proposedMaxServOutstandingCalled       1,
  initRequestDetail
  {proposedVersionNumber                 {0},
   proposedParameterCBB                  '2800'H,
   servicesSupportedCalling        '000000100000000000000212'H,
  }
 }
}
```

The PC gets the Initiate indication in the User Data field of the A_ASSOCIATE indication call.

The PC accepts the association, with an Initiate response PDU that is carried in the User Data field of an A_ASSOCIATE response. This Initiate response omits the optional fields "localDetailCalled" and "negotiatedDataStructureNestingLevel", allowing them to default. Also, the "serviceSupportedCalled" parameter is showing that MMS user (VMD) in the PC can support the CSPC conformance class 2.

Standard Notation:

```
{initiate-ResponsePDU
 {negotiatedMaxServOutstandingCalling    1,
  negotiatedMaxServOutstandingCalled     1,
  initResponseDetail
  {negotiatedVersionNumber               {0},
   negotiatedParameterCBB                '2800' H,
   servicesSupportedCalled         'EC00003F0FF40000000010'H,
  }
 }
}
```

The Cell Controller gets the Initiate confirm in the User Data field of the A_ASSOCIATE confirm call. The MMS provider in the Cell Controller determines the "serviceSupportedCalled" parameter according to the services which can be supported both by the MMS user in the PC (VMD) and the MMS provider in the Cell Controller. Suppose that the MMS provider in the Cell

Contrôleur de Cellules. Supposons que le fournisseur MMS sur le Contrôleur de Cellules peut envoyer la PDU de demande de services entrant dans la classe de conformité CSPC 4, alors le paramètre «serviceSupportedCalling» dans l'indication Déclenchement devient EC00003F0FF40000000010H. Cela signifie que l'utilisateur MMS sur le PC peut envoyer une PDU de demande de service entrant dans la classe de conformité CSPC 2.

### A.2.2 Le Contrôleur de Cellules obtient l'état du PC

Afin de s'assurer que le PC est en état de fonctionnement avant qu'un projet important ne soit lancé, le Contrôleur de Cellules obtient l'état du PC. Pour ce faire, il lit la variable nommée normalisée «P_PCSTATE».

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                    20,
  confirmedServiceRequest
  {read
   {specificationWithResult  FALSE
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name   {vmd-specific   "P_PCSTATE"}}
     }
    }
   }
  }
 }
}
```

Ayant reçu cette demande, le PC fournit les valeurs courantes de P_PCSTATE. La valeur de P_PCSTATE est «good», «noOutputDisabled» et «noInputDisabled».

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                    20,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success {bitstring      '8600'H}}
   }
  }
 }
}
```

### A.2.3 Le Contrôleur de Cellules définit les domaines

#### A.2.3.1 Création de domaine de programme

Le domaine de programme contient le code de programmation nécessaire pour que le PC pilote l'assemblage de la montre.

##### A.2.3.1.1 Entrée dans le mode Téléchargement

Le Contrôleur de Cellules demande la création d'un domaine qu'il appelle «ProgrammeMontre» en envoyant la PDU suivante au PC. Ce domaine contiendra tout le code de programmation et les données de l'application PC.

controller can send the services request PDU which conform to CSPC conformance class 4, then "serviceSupportedCalling" parameter in the Initiate indication becomes 'EC00003F0FF40000000010'H. This means MMS user in the PC can send service request PDU which conform to CSPC conformance class 2.

### A.2.2 Cell Controller Gets Status of the PC

In order to make sure that the PC is in good working order before a major project is begun, the Cell Controller gets the status of PC. This is performed by reading the standardized named variable "P_PCSTATE".

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                      20,
  confirmedServiceRequest
  {read
   {specificationWithResult    FALSE
    variableAccessSpecification
    {listOfVariable
     {variableSpecification
      {name   {vmd-specific  "P_PCSTATE"}}
     }
    }
   }
  }
 }
}
```

With the request, the PC provides the current values of P_PCSTATE. The value of P_PCSTATE represents "good", "noOutputDisabled" and "noInputDisabled".

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                      20,
  confirmedServiceResponse
  {read
   {listOfAccessResult
    {success              {bitstring   '8600'H}}
   }
  }
 }
}
```

### A.2.3 Cell Controller Sets Up The Domains

#### A.2.3.1 Create Program Domain

The program domain contains all the program code needed to direct a PC to assemble watch.

##### A.2.3.1.1 Enter Download Mode

The Cell Controller requests the creation of a domain it calls "WatchProgram" by sending the following PDU to the PC. This will be the domain to contain all of the PC application program code and data.

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                    21,
  confirmedServiceRequest
  {initiateDownloadSequence
   {domainName                "Watchprogram",
    listOfCapabilities        "MEMORY 5000 – 7FFF,
                                 CODE 0000 – 3FFF,
                                 DATA 4000 – 4FFF",
    sharable                  TRUE
   }
  }
 }
}
```

Après avoir envoyé la PDU de réponse suivante, le PC crée un nouveau domaine et lui attribue l'état CHARGEMENT EN COURS (techniquement parlant, il s'agit d'un état transitoire sur le diagramme des états entre l'état «NON EXISTANT» et CHARGEMENT EN COURS).

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                    21,
  confirmedServiceResponse
  {initiateDownloadSequence}
 }
}
```

#### A.2.3.1.2 Téléchargement du code de programmation

Le téléchargement du programme est lancé par le PC.

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                    22,
  confirmedServiceRequest
  {downloadSegment
   {domainName                "WatchProgram"
   }
  }
 }
}
```

Le PC passe l'état du domaine de CHARGEMENT EN COURS à TERMINÉ, une fois que cette PDU a été envoyée (les données de chargement sont dénommées «xx yy zz ...»).

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                    22,
  confirmedServiceResponse
  {downloadSegment
   {loadData
    {non-coded                xx yy zz ......,
    }
    moreFollows               FALSE
   }
  }
 }
}
```

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                                21,
  confirmedServiceRequest
  {initiateDownloadSequence
   {domainName                            "WatchProgram",
    listOfCapabilities                    "MEMORY 5000 – 7FFF,
                                            CODE 0000 – 3FFF,
                                            DATA 4000 – 4FFF",
    sharable                              TRUE
   }
  }
 }
}
```

After sending the following response PDU, the PC creates a new domain and puts it into the LOADING state (technically this is a transition on the state diagram from NON-EXISTENT to LOADING).

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                                21,
  confirmedServiceResponse
  {initiateDownloadSequence}
 }
}
```

#### A.2.3.1.2 Download Program Code

The downloading of the program is initiated by the PC.

Standard Notation:

```
{confirmed-RequestPDU
 {invokeID                                22,
  confirmedServiceRequest
  {downloadSegment
   {domainName                            "WatchProgram"
   }
  }
 }
}
```

The PC moves the state of the domain from LOADING to COMPLETE after this PDU has been sent. (The load data is represented by "xx yy zz ......".)

Standard Notation:

```
{confirmed-ResponsePDU
 {invokeID                                22,
  confirmedServiceResponse
  {downloadSegment
   {loadData
    {non-coded                            xx yy zz ......,
    }
    moreFollows                           FALSE
   }
  }
 }
}
```

#### A.2.3.1.3 Sortie du mode de téléchargement

Lorsque le téléchargement est achevé, le PC envoie la requête TerminateDownloadSequence pour sortir du mode de téléchargement.

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                    30,
  confirmedServiceRequest
  {terminateDownloadSequence
   {domainName                "WatchProgram",
    discard                   FALSE
   }
  }
 }
}
```

Le PC passe l'état du domaine de TERMINÉ à PRÊT une fois que la PDU a été envoyée.

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                    30,
  confirmedServiceResponse
  {terminateDownloadSequence}
 }
}
```

### A.2.4 Le PC commence l'assemblage des montres

#### A.2.4.1 Le Contrôleur de Cellules crée une invocation de programme

Une fois le téléchargement terminé, le Contrôleur de Cellules crée une invocation de programme qui correspond au domaine «ProgrammeMontre».

Notation normalisée:

```
{confirmed-RequestPDU
 {invokeID                    31,
  confirmedServiceRequest
  {createProgramInvocation
   {programInvocationName     "AssembleWatch"
    listOfDomainNames         {"WatchProgram"}
   }
  }
 }
}
```

Le PC crée l'invocation de programme désignée avec l'état de REPOS (techniquement parlant, il s'agit d'un état transitoire entre l'état NON EXISTANT et REPOS), passe l'état du domaine de PRÊT à UTILISÉ, et envoie la réponse suivante au Contrôleur de Cellules.

Notation normalisée:

```
{confirmed-ResponsePDU
 {invokeID                    31,
  confirmedServiceResponse
  {createProgramInvocation}
 }
}
```