INTERNATIONAL STANDARD

ISO/IEC 9506-4

First edition
1992-12-15

Industrial automation systems — Manufacturing Message Specification —

Part 4:
Companion standard for numerical control

Systèmes d'automatisation industrielle — Spécification de messagerie industrielle —

Partie 4: Norme d'accompagnement pour la commande numérique

Reference number
ISO/IEC 9506-4:1992(E)

Table of Contents

List of Figures

List of Tables

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO/IEC 9506-4 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Sub-Committee SC 1, *Physical device control*.

ISO/IEC 9506 consists of the following parts, under the general title *Industrial automation systems — Manufacturing Message Specification:*

— *Part 1: Service definition*

— *Part 2: Protocol specification*

— *Part 3: Companion standard for robotics*

— *Part 4: Companion standard for numerical control*

Annexes A to D form an integral part of this part of ISO/IEC 9506. Annex E is for information only.

Introduction

ISO/IEC 9506-1 defines the MMS services, a wide variety of services which are useful for the interworking of many types of manufacturing and process control devices when used in open communication systems conforming to the OSI Model (ISO 7498). However, these MMS services by themselves only provide a messaging environment for abstract, generic types of such control devices.

In order to convey device-specific semantics for real manufacturing equipment such as NC machines, robots, programmable controllers, process control systems and so forth, many of the MMS services have optional parameters which have not been detailed in MMS. MMS also specifies that any of these parameters may only be used in the context of a so-called "companion standard" designed for specific devices by the appropriate, recognized standardizing organization.

This part of ISO/IEC 9506 is intended to be such a companion standard to ISO/IEC 9506-1 and ISO/IEC 9506-2. Specifically, it defines the numerical control semantics to the manufacturing message specification. It should be used when numerically controlled manufacturing systems or devices are connected to a communication network conforming to the OSI Model and employing the MMS service and protocol.

Part 4 of ISO/IEC 9506 hereinafter shall be variously referenced as "this standard", "this International Standard", "this part of ISO/IEC 9506", or simply "ISO/IEC 9506-4".

The definitions and specifications of the numerical control specific semantics in this part of ISO/IEC 9506 follow the requirements and guidelines of annex A of ISO/IEC 9506-1.

NOTE 1 This part of ISO/IEC 9506 does not constrain the mapping from real to virtual numerically controlled manufacturing devices, nor does it specify individual implementations or products. This standard also recognizes that safe operation of NC devices is paramount, and that any operations described or defined herein are permissible only if complying with the relevant safety standard(s) and regulations.

This page intentionally left blank

Industrial automation systems — Manufacturing Message Specification —

Part 4:
Companion standard for numerical control

1 Scope

This part of ISO/IEC 9506 extends the concepts and principles defined in ISO/IEC 9506-1 and ISO/IEC 9506-2.

This part of ISO/IEC 9506, as a companion standard to Manufacturing Message Specification, describes the semantics of numerically controlled manufacturing devices and equipment. Specifically, this part of ISO/IEC 9506

a) describes the use of Manufacturing Message Specification services in NC specific applications,

b) describes the model of an NC device in terms of its application specific functions and how these functions map onto the attributes of a virtual manufacturing device (VMD),

c) provides the NC specific syntax for those MMS services which are applicable to NC operations, and which also allow companion standard specific parameters,

d) defines "standardized" names for NC specific objects,

e) defines NC specific conformance in conjunction with the MMS service and parameter conformance building blocks (CBBs) required.

Furthermore, it should be noted that in addition to this part of ISO/IEC 9506, within ISO and IEC other companion standards have been developed or such work is in progress, in the application areas of robots and robotic systems, programmable control systems, and process control systems.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9506. At the time of publication, the editions indicated were valid. All standards are subject to revision, and 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 standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 841:1974, *Numerical control of machines – Axis and motion nomenclature.*

ISO 2806: –[1)], *Numerical control of machines – Vocabulary.*

1) To be published. (Revision of ISO 2806:1990.)

ISO 7498:1984, *Information processing systems – Open Systems Interconnection – Basic Reference Model.*

ISO/TR 8509:1987, *Information processing systems – Open Systems Interconnection – Service conventions.*

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

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

ISO/IEC 9506-2:1991, *Industrial automation systems – Manufacturing– Message Specification – Part 2: Protocol Specification.*

3 Definitions

For the purposes of this part of ISO/IEC 9506 the following definitions apply.

3.1 Alarm

The immediate indication or notification of the occurrence of a severe problem (malfunction) or abnormal condition within the NC system, usually causing an interruption of the operation, which should be resolved externally (such as through operator intervention).

3.2 Device

The whole or part of the machinery or electro-mechanical equipment employed in manufacturing or other processes. Devices may be machine tools, functional parts of machine tools, workpiece handling equipment, or any similar apparatus. In this context, a device shall be visible to an NC-CS user as a separate unit which may be controlled independently by the NC-CS user, such as started and stopped.

3.3 Numerical Control (NC)

Automatic control of a manufacturing or other process performed by one or more device(s), usually involving motion mechanisms, and making use of preprogrammed numeric data, which may be introduced while the operation is in progress.

3.4 Numerical Controller

An electronic control system designed to perform the numerical control (NC) functions of manufacturing or other processes. It usually consists of a computer, file store(s), both analogue and digital input and output interfaces, and one or more operator interface device(s).

3.5 NC-CS (NC Companion Standard)

This part of ISO/IEC 9506.

3.6 NC-CS user

The part of an application process which conceptually invokes the NC Companion Standard. In the context of this part of ISO/IEC 9506, this refers to the external, or "host", processor which communicates with, and exerts some measure of remote control over the NC system.

3.7 NC system

A workstation which consists of a numerical controller and one or more devices.

3.8 Operator Interface Device

An operator interface device provides the means for the NC-CS user to communicate with the NC system operator. Conversely, the same interface may be used by the NC system operator to control the NC system in the LOCAL CONTROL mode.

3.9 Offset

Term which may refer to such general concepts as zero offset, axes value transformation, coordinate transformation, or geometry transformation.

3.10 Axis

Refers to a real, guiding (linear or rotary) axis of a machine, which also reflects one dimension in the work space of the machine.

3.11 Coordinate Axes

See Coordinate System.

3.12 Zero Offset

The relationship (in one dimension) between the origins of a real machine axis and the corresponding part program coordinate.

3.13 Axis Value Transformation

(One dimensional) transformation (sequence of such primitives as translation, mirroring, and scaling) of a point on a real machine axis in terms of the corresponding part coordinate.

3.14 Coordinate System

Defined in ISO 841

NOTE - The Coordinate System may also be uniquely defined by three orthogonal coordinate axes. Always right-handed

3.15 Actual Coordinate System

The actual coordinate system used by the NC.

3.16 Coordinate Transformation

Expression of the spatial relationship (location and orientation) between two independent coordinate systems.

3.17 Geometry Transformation

The application of a sequence of primitive transformations (translation, rotation, scaling, mirroring) to a point in space in terms of the actual coordinate system.

In addition to the definitions in this clause, the definitions of clause 3 of ISO/IEC 9506, part 1 apply, which in addition to MMS-specific definitions lists a number of terms defined in ISO 7498, the OSI Basic Reference Model, in ISO/TR 8509, the OSI Service Conventions, and in ISO 8824, the Abstract Syntax Notation One Specification. All of these definitions apply to this standard by reference.

NC specific definitions may also be found in ISO 2806.

4 Abbreviations

The following abbreviations are used in this part of ISO/IEC 9506:

ASN.1	Abstract Syntax Notation One
CBB	Conformance Building Block
Cnf	confirm
CS	Companion Standard, often used with NC: NC-CS
Ind	indication
M	mandatory parameter
MMS	Manufacturing Message Specification
NC	Numerical Control or Numerical Controller
OSI	Open Systems Interconnection
PICS	Protocol Implementation Conformance Statement Proforma
Req	request
Rsp	response
VMD	Virtual Manufacturing Device

5 NC Application Description

This chapter describes the model and functionality of an NC system in a manufacturing communication environment.

5.1 NC Specific Model

An NC system is a workstation, where one or more devices are controlled by a Numerical Controller (NC). Such devices may consist of any mechanical or electro-mechanical equipment depending on their individual application. An NC system may be connected to other stations through one or more communication channel(s).

In a factory, several NC-CS users may communicate with a single NC system, as shown in Figure 1.

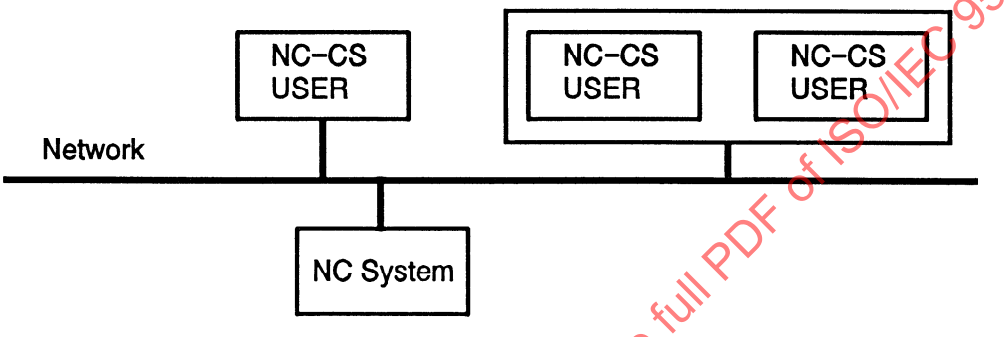

Figure 1: NC System in the Manufacturing Environment

An NC system may be described by models of its component parts. Figure 2 models the component parts. These component parts may describe aspects of the numerical controller, the device, or both.

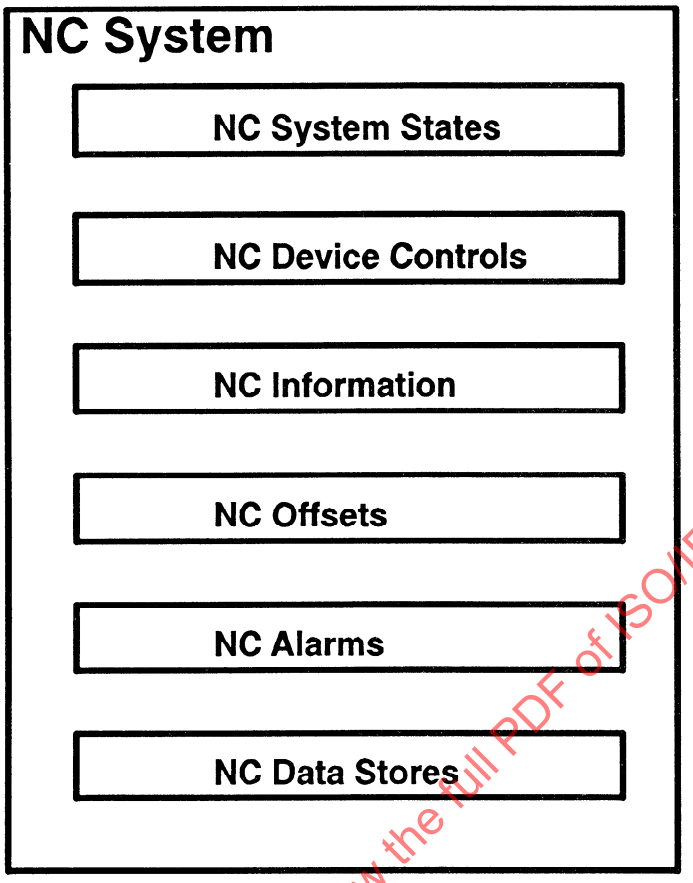

Figure 2: NC System Components

The models in this section describe the elements of the NC application area from a communication point of view. They include the following models:

- the NC system state model,
- the NC device control model,
- the NC information model,
- the NC offset model,
- the NC alarm model,
- the NC data store model.

5.1.1 NC System State Model

The NC System State Model describes the global state of an NC system, as visible to an NC-CS user. Also, it represents a mechanism which allows coordination between local users and NC-CS users (remote users), as well as the coordination between multiple NC-CS users.

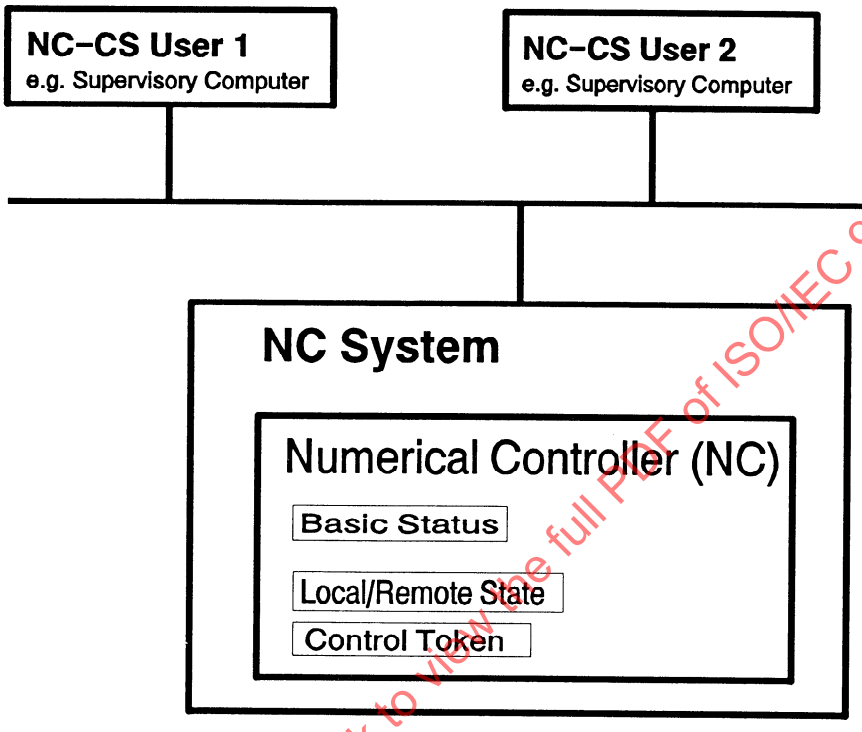

Figure 3: NC System State Model

5.1.1.1 Basic Status

The Basic Status describes the overall working condition of the NC system.

5.1.1.1.1 State Descriptions

a) NORMAL PRODUCTION - The NC system is fully operable.

b) LOCAL-ACTION REQUIRED - A component of the NC system has failed or is not operable

5.1.1.2 Local/Remote State

In the factory environment an NC system may be controlled by an NC-CS user (remotely) or locally, as shown in figure 4.

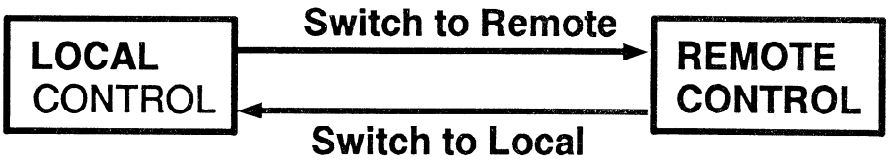

Figure 4: Local/Remote State

5.1.1.2.1 State Descriptions

a) LOCAL CONTROL - In this state, the NC operation is under control of a local user. For a transition to the REMOTE CONTROL mode, local action by the operator is required. ISO/IEC 9506 part 4 does not include any states or operations under LOCAL CONTROL.

NOTE - Even in LOCAL CONTROL some remote operations may be possible. For example, an NC-CS user may inform the NC system about the next part to be made. The choice of remote operations which may be allowed in LOCAL CONTROL is implementation dependent, and should be closely related to security. This choice may even be changed as a result of operator decisions.

b) REMOTE CONTROL - In this state, the NC system is controlled by a remote user. Any operator intervention which modifies the state of the controlling process, or which results in a safety violation shall force the transition to LOCAL CONTROL.

5.1.1.2.2 State Transitions

a) Switch to Remote - This transition shall only be the result of local action and shall enable remote control of the NC system. While this transition may also depend on the state of the Controlling Process, the exact conditions for this shall be at the discretion of the system implementer.

b) Switch to Local - This transition shall change the NC system from REMOTE CONTROL to LOCAL CONTROL. It may take place during any of the Device states and Controlling Process states, and the resultant states of the Device(s) and/or Controlling Process(es) shall be implementation dependent.

5.1.1.3 Control Token

An NC system may be controlled by several NC-CS users. Such a multiple client configuration requires a mechanism for taking and relinquishing control of the NC system. Without this capability, there is no means for preventing two or more NC-CS users from attempting to control the NC system simultaneously.

5.1.2 The NC Device Control Model

The NC Device Control Model may be viewed as a number of interacting components, such as one or more Devices, a Numerical Controller, an Operator Interface Device, and one or more Controlling Processes, as illustrated in figures 5 through 7.

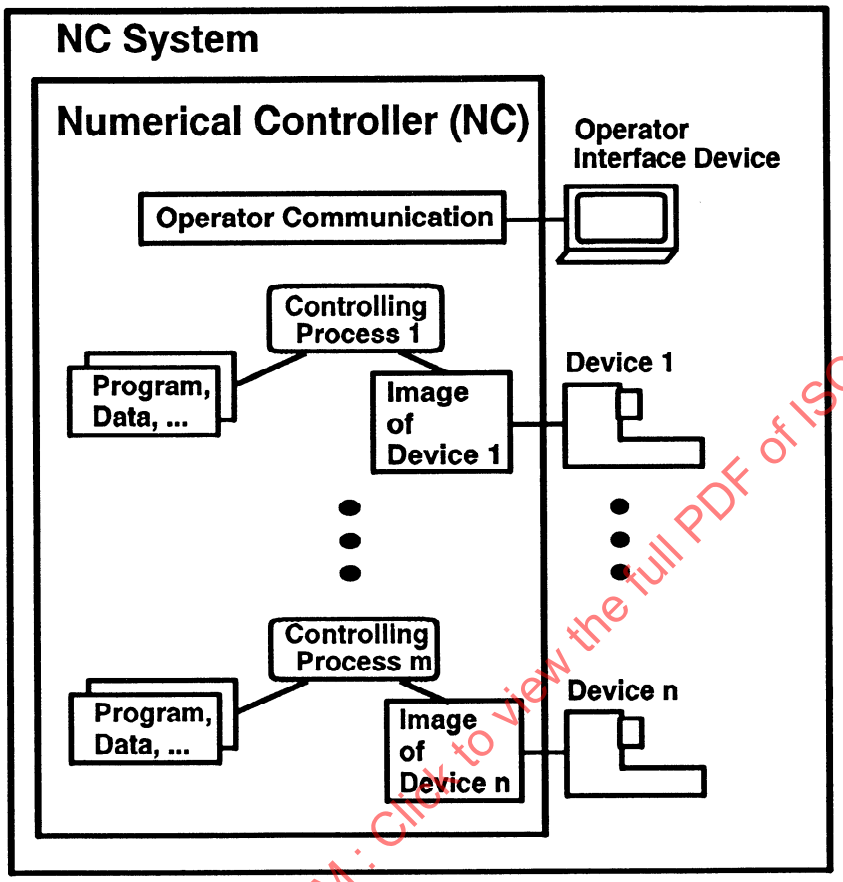

Figure 5: NC Device Control Model

5.1.2.1 Devices

Devices represent those mechanical or electro-mechanical parts of an NC system which are visible to an NC-CS user as a separate entity which may be controlled separately, such as started or stopped.

NOTE - The manner in which devices are delineated within a system shall be arbitrary. While one implementation may consider as devices such overall entities as a milling machine or tool warehouse, another one may subdivide a system into smaller units, such as spindle, rotary table, or tool changer.

5.1.2.2 Image of Device

The Numerical Controller contains an image of every Device. The image fits with some mechanics and some resources of the Device configured when the NC System is set up. The Device Image includes for example the actual state of the Device or the number of axis.

5.1.2.3 Controlling Process

Controlling Processes represent the execution of the user programs, which are linked to part programs and data such as tool corrections and fixture offsets. They may also be linked to execution information, such as the current execution state, the line of program currently being processed, and so on. Controlling Processes are linked to devices, and they may be created dynamically by the NC-CS user.

5.1.2.4 Operator Interface Device

An operator interface device provides the means for the NC-CS user to communicate with the NC system operator. Physically, the same interface may also be used by the NC system operator to control locally the NC system.

5.1.2.5 Single Device NC System

The general model of an NC system may consist of one or more devices and one or more Controlling Processes. A simple numerical controller, however, may support only a single device and a single Controlling Process.

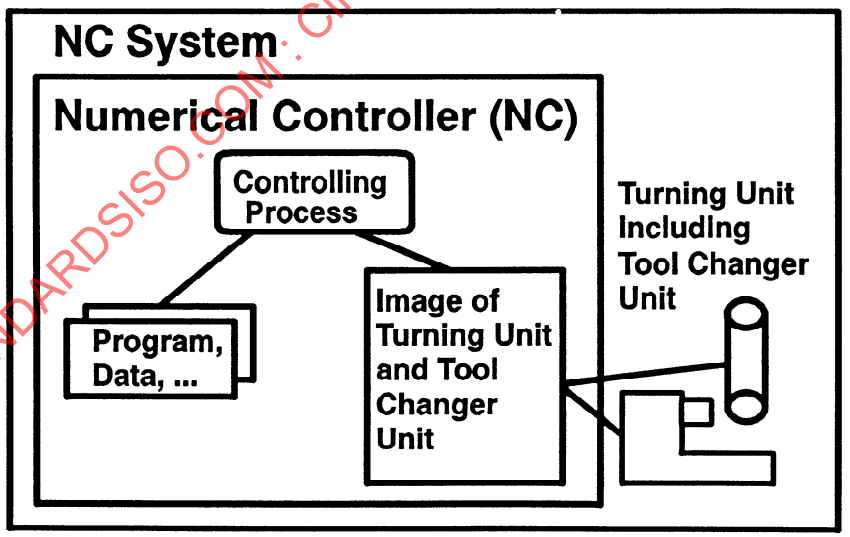

Figure 6: NC Device Control Model (Single Device)

For example figure 6 shows an NC system consisting of a single device and a single Controlling Process. The device image represents a turning unit and a tool changer. The Controlling Process links programs and data to the device.

5.1.2.6 Multiple Device NC System

Other numerical controllers may be able to control several, usually interacting devices. An example of such a system is shown in figure 7, which models three separate devices, two turning units, and a tool warehouse. The NC-CS user may act on each of these devices independently.

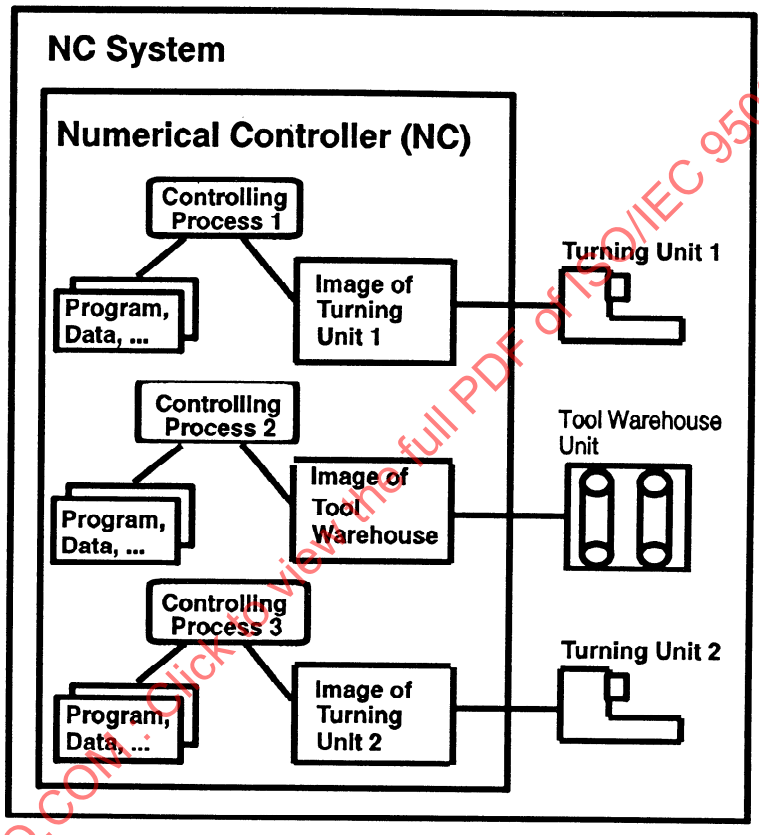

Figure 7: NC Device Control Model (3 independent Devices)

The operation of the NC device is described by the Device State Model and the Controlling Process State Model.

5.1.2.7 The Device State Model

The Device State Model describes the state of the device. The image of the device includes the Device State.

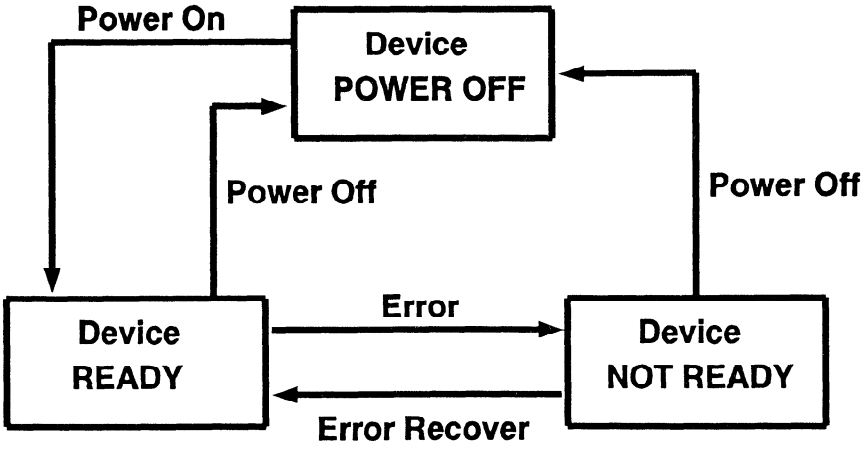

Figure 8: NC Device State Model

5.1.2.7.1 State Descriptions

a) POWER OFF - In this state, all device functions are inactive except for those pertaining to safety (e.g. brakes, clamps). No power should be applied to the device.

b) READY - In this state, power is applied to the device, that is, axes drives, spindle drives, hydraulic pumps, etc.

c) NOT READY - This state may be caused by an error or a stop condition. If caused by an error condition, only an action which acknowledges and removes the error condition may cause an exit from this state. It is assumed that the NC provides sufficient information for proper diagnosis.

5.1.2.7.2 State Transitions

a) Power On - This is associated with energizing the device, and represents the transition from POWER OFF to READY.

b) Power Off - This transition puts the device into the POWER OFF state, which shall halt all operations of the device.

c) Error - This transition puts the device from the READY state into the NOT READY state.

d) Error Recover - This transition returns the device to the READY state.

NOTE - The above device state model is simplified. In reality, such transitions as Power On and Power Off may be rather complex procedures.

5.1.2.8 The Controlling Process State Model

The Controlling Process state model provides the user with the information about the execution state of the user program. A Controlling Process may control one or more devices. For an NC-CS user it is necessary to be aware of the program, and the way the program acts on the devices.

Therefore, the Controlling Process state model describes the execution states of the user program, such as start, stop, and so on.

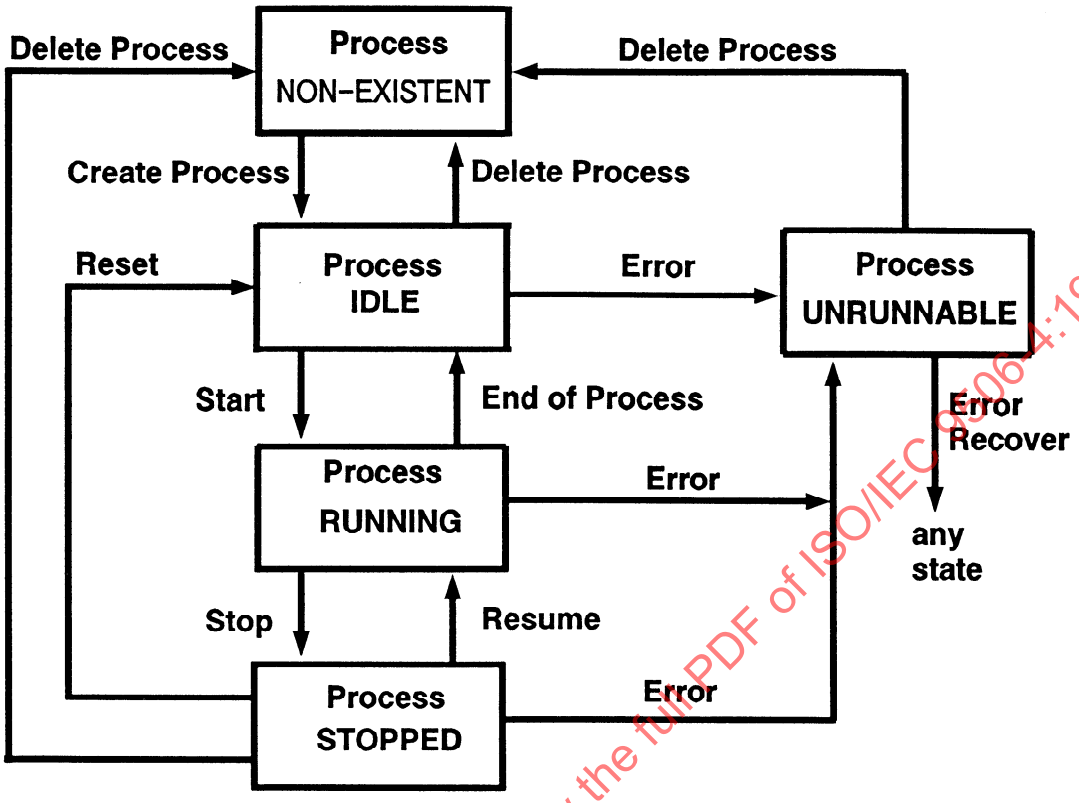

Figure 9: NC Controlling Process State Model

The Controlling Process may be created dynamically by the NC-CS user. Such creation usually entails the binding of a device to the Controlling Process, and the selection of user programs and data. To operate any device, at least one Controlling Process is needed. The binding of some Controlling Processes and their devices may be static.

5.1.2.8.1 State Descriptions

a) NON-EXISTENT - This is the state of a Controlling Process prior to its creation.

b) UNRUNNABLE - This state is the result of an error. Recovery from this state requires the explicit action of an operator.

c) IDLE - In this state the Controlling Process is ready to execute, that is, all required initialization have been completed.

d) RUNNING - In this state the Controlling Process is fully operational and executing user program commands.

e) STOPPED - This state may be the result of a stop command, or any other stop condition. From this state, after stopping, the user program may resume execution normally.

5.1.2.8.2 State Transitions

State transitions may be initiated from a number of sources, such as the NC-CS user, the operator, the user program, or the Device. Also, it is presumed that the manufacturer may define certain conditions which will inhibit these transitions, while providing sufficient information to properly diagnose the situation.

a) Create Process - This transition creates a Controlling Process by linking user programs, data, and devices.

b) Delete Process - This transition deletes a Controlling Process, by deleting all such links between programs, data, and devices.

c) Start - This transition starts the execution of the Controlling Process, assuming that all prerequisites have been fulfilled.

d) End of Process - This transition completes the execution of the Controlling Process. This transition usually corresponds to the end of the program.

e) Stop - This transition stops execution of the program. It may be caused by a stop command from the NC-CS user, and it puts the Controlling Process into the STOPPED state. Execution may be restarted by the Resume transition.

f) Resume - This transition restores the RUNNING state.

g) Error - This transition shall suspend the Controlling Process and cause it to be UNRUNNABLE. The cause of the error shall be removed before the process may be recovered from this state.

h) Error Recover - Removing the cause of a previous error by local or remote action shall result in this transition from the UNRUNNABLE state to any state.

i) Reset - This transition puts the Controlling Process into the IDLE state.

5.1.2.8.3 Correlation To The Device State

Since the states of a device and its associated Controlling Process states are interdependent they will influence each other. For instance, when a device is switched off or enters the NOT READY state, then there should be a Controlling Process state transition.

The following table describes the correlation between device states and Controlling Process states.

Table 1 - Correlation between Device States and Controlling Process States

Device State	Possible Controlling Process States
POWER OFF	NON-EXISTENT, UNRUNNABLE, IDLE
READY	any state
NOT READY	NON-EXISTENT, UNRUNNABLE, STOPPED, IDLE

5.1.3 NC Information Model

Basic Status, Device Status, and the Controlling Process Status provide the NC-CS user with general information about the state of the NC system. However, to provide the more detailed information about the NC system required for remote control by the NC-CS user, this part of ISO/IEC 9506 includes the additional types of information described here by the NC Information Model.

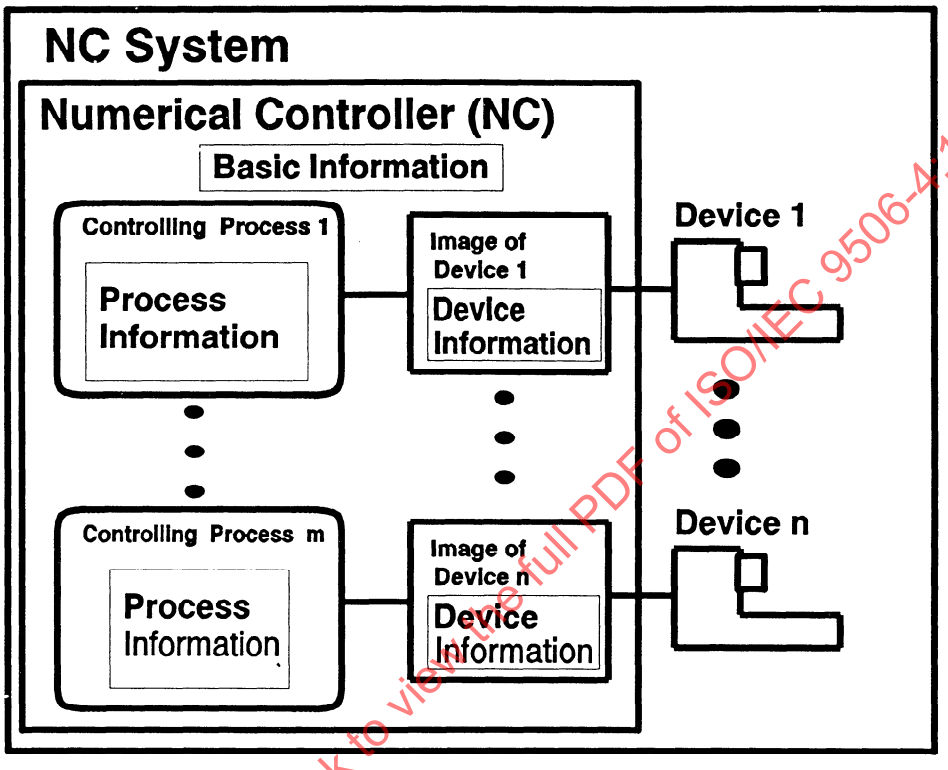

Figure 10: NC Information Model

5.1.3.1 Basic Information

This encompasses information about the Numerical Controller, such as controller voltage or controller error information.

5.1.3.2 Device Information

This encompasses information about the device, such as hydraulic pressure or device voltage.

5.1.3.3 Process Information

This encompasses information about the Controlling Process, e.g. the currently executing program block.

5.1.4 Offset Model

An important part of the operation of an NC system is the capability to handle axis transformation, zero offsets, and other coordinate transformations. In stand-alone, conventional NC systems most of this functionality is handled by the machine operator, as well as by user programs. The purpose of the Offset Model is to provide the NC-CS user with these functions.

The Offset Model shall provide the NC-CS user with all the geometric transformation capabilities required for remote control. Offsets represent the transformations of the location descriptions of points which are usually associated with the machined part or workpiece. The Offset Model distinguishes two types of such transformations, one dimensional and three dimensional, that is, in terms of one axis, or in terms of Cartesian coordinates. Both types further distinguish between (1) transformation of the reference origin, and (2) transformations of axis values or geometry. For better visualization of the Offset Model refer to figures 11 through 15.

For tutorial information on this, see (informative) Annex E.

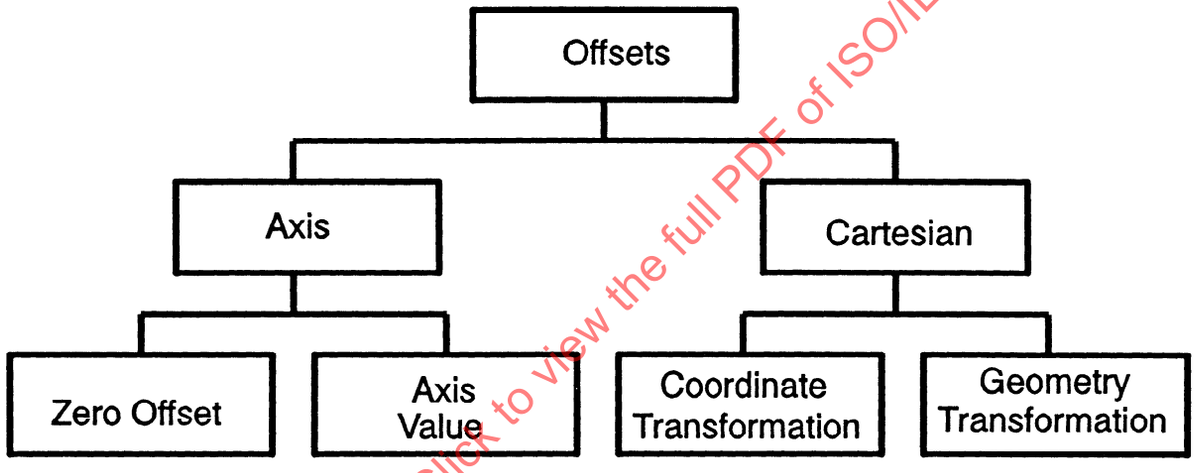

Figure 11: Offset Classification Hierarchy

5.1.4.1 Zero Offset

A zero offset shifts the virtual origin (zero point) of a physical machine axis (see figure 12). This virtual origin is the reference for the part program.

Zero offsets equally apply to linear and rotary axes.

virtual origin = real origin + zero offset 1 + zero offset 2 ... + zero offset n

The number of zero offsets supported in an NC is implementation specific, and shall be stated by the appropriate PICS entry.
For rotary axes the permissible range shall be specified (e.g. 0..360, -180..180, ...).

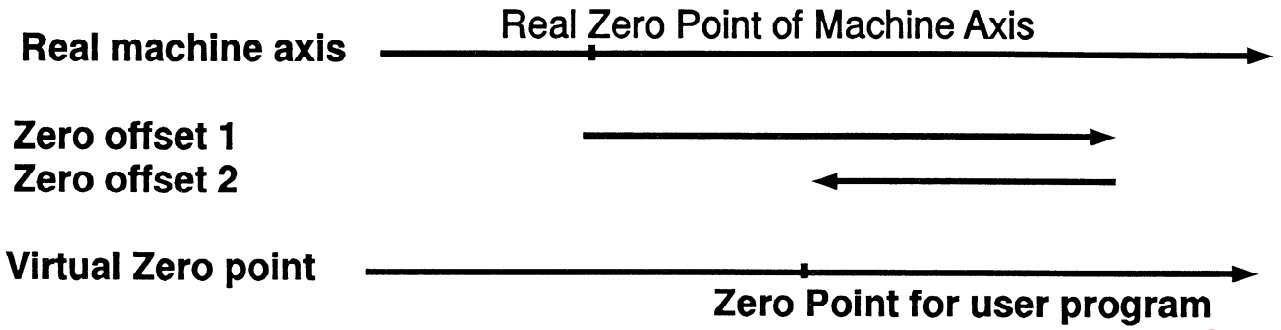

Figure 12: Zero Offset and Virtual Zero Point

5.1.4.2 Axis Value Transformation

All axis value transformations are with reference to the virtual origin of the respective axis. Rotary and linear axes value transformations are treated equally. In the end effect, axis value transformations change the location or geometry of the part or workpiece produced by the user program.

ISO/IEC 9505-4 defines the following primitive axis value transformations (see figure 13):

```
Translation:
  Translation of the point P0 with value x
          primitive T(x):     P1 = T(x) * P0     (matrix notation)
                              p1 = x + p0        (coordinate notation)
Mirroring:
  Mirroring of point P0 about the virtual origin.
          primitive M():      P1 = M() * P0      (matrix notation)
                              p1 = - p0          (coordinate notation)
Scaling:
  Scaling of point P0 relative to the virtual origin with the scale factor x.
          primitive S(x):     P1 = S(x) * P0     (matrix notation)
                              p1 = x * p0        (coordinate notation)
```

Scaling and mirroring by themselves, separately, are commutative. Otherwise, any sequence of primitive operations is order dependent. Following is a commonly used notation which reflects the order:

```
P1 = T(2) * M() * S(2) * T(1) * P0
                         ^first primitive
                   ^second primitive
             ^third primitive
       ^fourth primitive
```

NOTE 1 - This standard does not limit the number of primitive transformations, which shall be implementation specific.

NOTE 2 - The range specification for rotary axes shall be implementation specific (for instance, 0..360, -180..180, ...).

NOTE 3 - Complex axis value transformations may be expressed by a sequence of operations, such as mirroring about centre point 5, which may be achieved by the following expression:

P1 = T(5) * M() * T(-5) * P0

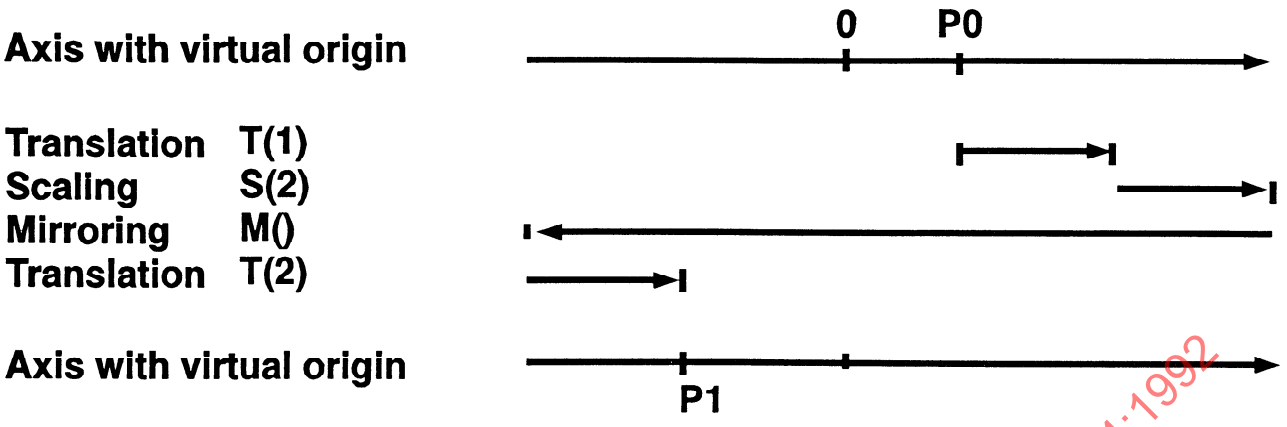

Figure 13: Sequence of Primitive Axis Value Transformations

5.1.4.3 Coordinate Transformations

A coordinate transformation defines the position and rotation of one coordinate system in terms of another coordinate system. Such transformation should consist of a rotation (Euler angles) and a vector, in accordance with the conventions of ISO 841.

Objects defined in this part of ISO/IEC 9506 which may represent pallets, fixtures, parts, or partitions may have offset attributes which refer to coordinate transformations. An example of coordinate transformations is shown in figure 14.

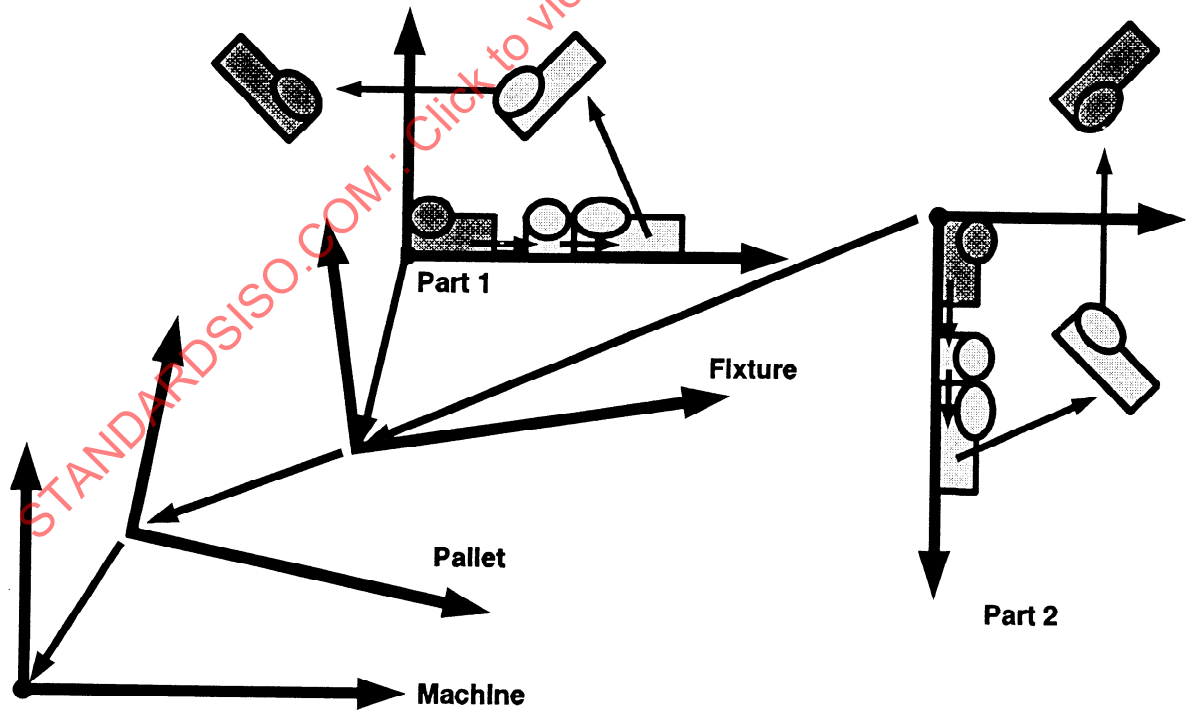

Figure 14: Coordinate Transformations

User programs are written for workpieces or parts in terms of their own specific coordinate systems. To execute correctly, sufficient information about such a coordinate system is required within the NC. This information shall be contained in an object known as the "Active Program Coordinate System".

5.1.4.4 Geometry Transformation

Geometry Transformation defines the transformation applied to the coordinates of a point in three dimensional space, to change its reference from the original coordinate system, usually the "program coordinate system", to some target coordinate system, usually the coordinate system of the machining environment. The effect of geometry transformations is the change of location, orientation, or geometry of the workpiece produced by the part program.

The following defines the primitives used in geometry transformations. The original coordinate system is assumed to be in terms of the coordinates x, y, and z:

```
Translation:
      Translation of point P0 with vector v
      primitive T(v):            P1 = T(v)*P0          (matrix notation)

Scaling:
      Scaling of the x,y,z components of P0 with scale factors s1, s2, s3.
      primitive S(s1,s2,s3):     P1 = S(s1,s2,s3)*P0     (matrix notation)


Rotation:
     Sequence of three rotations (Rotation defined by ISO 841), defined in the
     following order:
        1. Rotation about x axis with angle a
        2. Rotation about y axis with angle b
        3. Rotation about z axis with angle c

      primitive R(a,b,c):        P1 = R(a,b,c)*P0       (matrix notation)

Mirroring:
      M ( xy, yz, xz )
            For xy, yz and xz only the two values 0 and 1 are valid.
  0 means no mirroring about the specified plane.
            1 means mirroring about the specified plane.


      Examples:
            mirroring about the xy plane:                M(1,0,0)
            mirroring about the yz plane:                M(0,1,0)
            mirroring about the xz plane:                M(0,0,1)
            mirroring about the xy and yz plane:         M(1,1,0)
```

Mirroring by itself is commutative. Otherwise, any sequence of primitive operations is order dependent. Following is a commonly used notation which reflects the order (see figure 15):

```
      P1 = T(1,0,0) * R(10,20,30) * M(1,0,0) * P0
                                    ^first primitive
                    ^second primitive
           ^third primitive
```

NOTE 1 - When using the mirroring primitive, the shape of a workpiece remains the same. For example, if the program describes a cylinder, then after a mirroring operation the cylinder remains a cylinder. When using the scaling primitive with negative scale factors, the shape changes. Thus, the cylinder program, after scaling with the scale factor -1, will produce a hole in the work piece.

NOTE 2 - This standard does not limit the number of primitive transformations, which shall be implementation specific.

NOTE 3 - Complex geometric transformations may be expressed by a sequence of primitives.

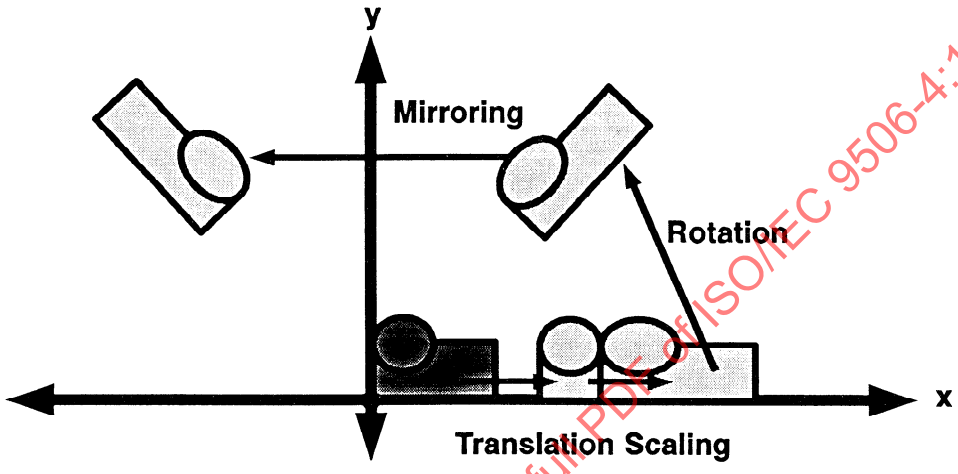

Figure 15: Sequence of Primitive Geometry Transformations

5.1.5 NC Alarm Model

In order to properly perform its task within a manufacturing environment, an NC shall have the ability to signal faults to an NC-CS user.

NOTE - Such faults may be measuring system faults, control unit temperature problems, control unit battery problems, and so on. Typically, within an NC there exists a list of hundred or more fault conditions, each of which may cause an alarm.

The monitoring and signaling of alarms is represented by the NC Alarm Model. If an Alarm occures and alarm monitoring is enabled, an NC Alarm Notification is sent to the NC-CS user. The NC Alarm Notification includes the following information:

NC Alarm Name - Every alarm which is to be signaled to an NC-CS user is uniquily identified by its NC Alarm Name

NC Alarm State - This parameter represents the current state of Alarm Monitoring (DISABLED, IDLE, ACTIVE).

NC Alarm State = DISABLED indicates that Alarm monitoring is disabled. When alarm monitoring is disabled, a NC Alarm Notification may be send to the NC-CS user (NC Alarm State = DISABLED) to inform the NC-CS user.

NC Alarm State = IDLE indicates that the monitored alarm is not active or the alarm has ceased to exits.

NC Alarm State = ACTIVE indicates that the monitored alarm is active.

NC Alarm Severity - This parameter represents some measure of severity for NC-CS user receiving the NC Alarm Notification.

NC Alarm Time - This parameter contains the time (date and time or a time sequence identifier) at which the transition in the NC alarm state was detected.

NC Alarm Acknowledgment Rule - This parameter indicates how the NC-CS user shall acknowledge the receipt of the NC Alarm Notification.

NC Alarm Acknowledgement Rule = NONE indicates that there are no acknowledgements required.

NC Alarm Acknowledgement Rule = ACK-ACTIVE - Acknowledgement is required for Alarm Notifications indication the NC Alarm State ACTIVE. For the NC Alarm State IDLE, acknowledgements are allowed but not required.

NC Alarm Acknowledgement Rule = ACK-ALL - Acknowledgements are required for Alarm Notifications indicating the ACTIVE or IDLE Alarm State.

NOTE - The Alarm Acknowledgement Rule may be different for every alarm. For NC Systems not supporting the acknowledgement of alarms, this parameter shall have the value NONE.

5.1.6 NC Data Store Model

Most numerical controllers provide data storage, as well as the capability of remote selection, inspection, and/or retrieval of any data required for NC operations. Such data typically consist of user programs and operational parameters, such as tooling, fixturing, process, and other parameters. They are usually generated remotely, and transferred to the NC either via a communication network, or by local means. The Data Store Model defines the organization of the data store, and the manner in which the data may be viewed.

While this part of ISO/IEC 9506 does not specify any particular data store hierarchy, it does provide for multiple data stores. Such data stores may be within or external to the numerical controller. External store(s) are included in this model so that they may be visible to the NC-CS user.

NOTE - ISO/IEC 9506-4 only defines the model of a logical data storage system from the viewpoint of an NC-CS user, and the mapping of such model to a real data store is outside the scope of this part of ISO/IEC 9506.

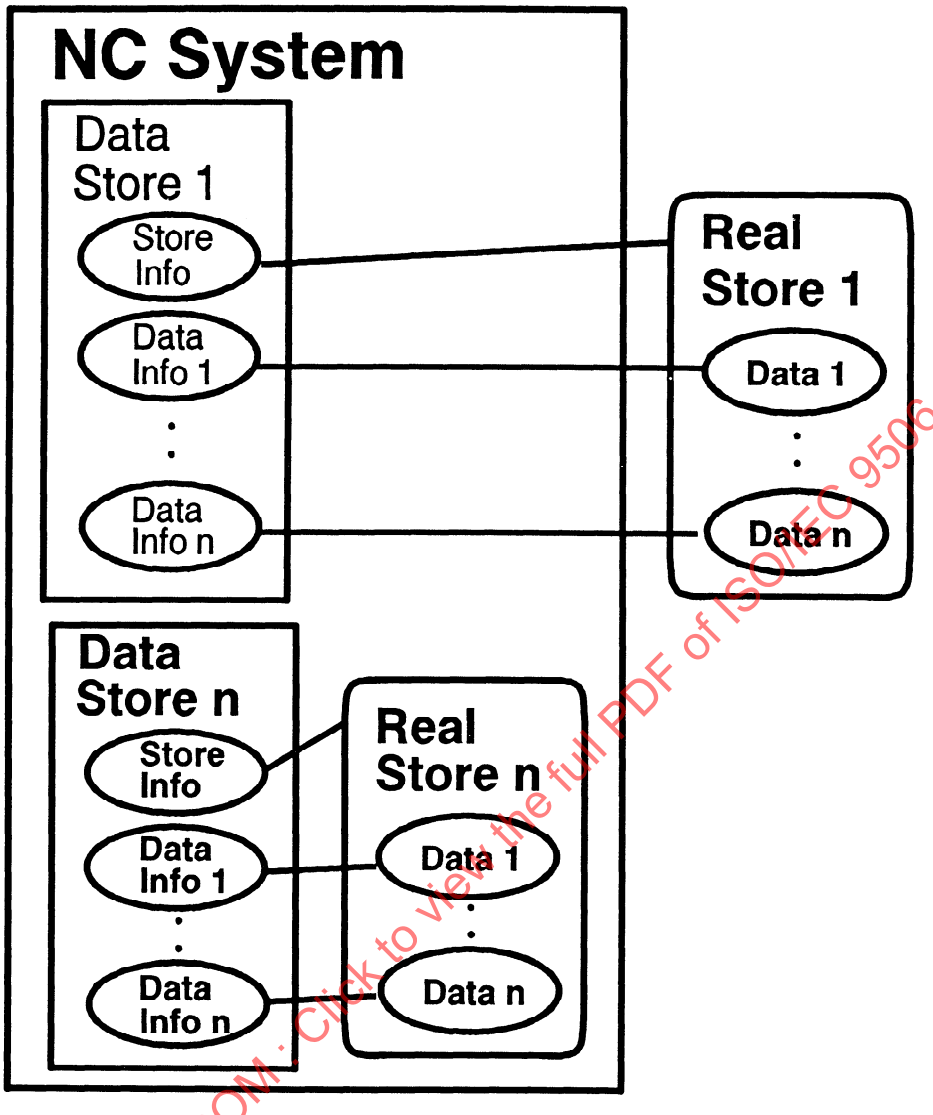

Figure 16: NC Data Store Model

5.2 NC Specific Functions

This part of ISO/IEC 9506 does not define any functional classes of NC systems. Instead, it defines the following, general NC specific functions, any of which may or may not be supported by a particular NC system. The mapping of these functions to MMS is defined in Chapter 6.

5.2.1 Data Transfer

Numerical controllers conforming to this part of ISO/IEC 9506 are able to transfer data from the NC-CS user to the NC system and back. There are 3 different ways to transfer data to and from an NC system:

a) Block Transfer to Datastore - The transfer of "bulk" data between the NC-CS user and the NC system (Download and Upload).

b) Block Transfer Dynamic - The NC system executes the data during transfer with minimal buffering (Dynamic Download).

c) Variable Transfer - In this way data are viewed as variables and the data transfer is done by reading or writing these variables.

The NC-CS user may also direct the NC system to load or store data from a remote fileserver.

Some data, such as tooling data, may be handled either as data blocks or as variables, or both, depending on the complexity of a particular application. Simple numerical controllers may handle tooling data using block transfer, and more complex systems may use either block transfer or variable transfer, or both. Variable data structures to handle a variety of applications are defined in the normative Annexes to ISO/IEC 9506-4.

5.2.2 NC Data Store Management

The NC Data Store Model provides to the NC-CS user the information which addresses the following data management capabilities:

a) Storage Organization - the names of the NC-CS user programs in the data store(s).

b) Version Control - the origin and/or version attribute of NC-CS user programs in the data store(s).

c) Memory Usage - the amount of data store space allocated to NC-CS user programs.

d) Memory Location - the kind of storage medium represented by the data store.

e) Data Type - the format of NC-CS user programs in the data store.

f) Protection Schema - the integrity, or accessibility of the data in the data store.

g) Deletion of data store entries.

5.2.3 Process management

For remote control of the NC system, the following functions which relate to program control are required:

a) Select a machine program for execution.

b) Start the execution of a Controlling Process. (Remote cycle start)

c) Pause and resume a Controlling Process.

d) Stop the execution of a Controlling Process.

5.2.4 Interaction with an operator

The operator interface device provides the means for the NC-CS user to communicate with the NC system operator.

5.2.5 Alarm Processing

In order to signal faults to the NC-CS user, NC systems should have the capability to:

a) mask/unmask alarms

b) signal alarms to the NC-CS user

c) receive alarm acknowledgments

d) provide summary information of active alarms

6 NC Specific Context Mapping

This clause relates the model of the NC system which has been defined in clause 5, to the abstract model of the Virtual Manufacturing Device (VMD) defined in MMS. For the purpose of inter-operability the uniform application of the abstract concepts in MMS to real systems is essential. While this part of ISO/IEC 9506 may not anticipate every variant of NC System existing or yet to be developed, it provides general guidelines for associating real systems with the abstract model, which should be applicable to any conforming NC system.

6.1 Mapping Of The NC Specific Model To The VMD Object

This clause shows how different elements of NC systems relate to the MMS abstract concepts of VMD, Domains, Program Invocations, and Variables etc. It is equally applicable to (simple) NC systems which map to a single VMD, as well as those which map to multiple VMDs. The distinction depends largely upon the application.

6.1.1 Mapping Of The NC System To The VMD Model

This clause defines how the NC specific models of clause 5.1 map to the abstract objects defined in part 1 of this International Standard.

Figure 17 shows the major components of an NC VMD and their relation to each other. An NC VMD may include variables, domains, and one or more Program Invocations.

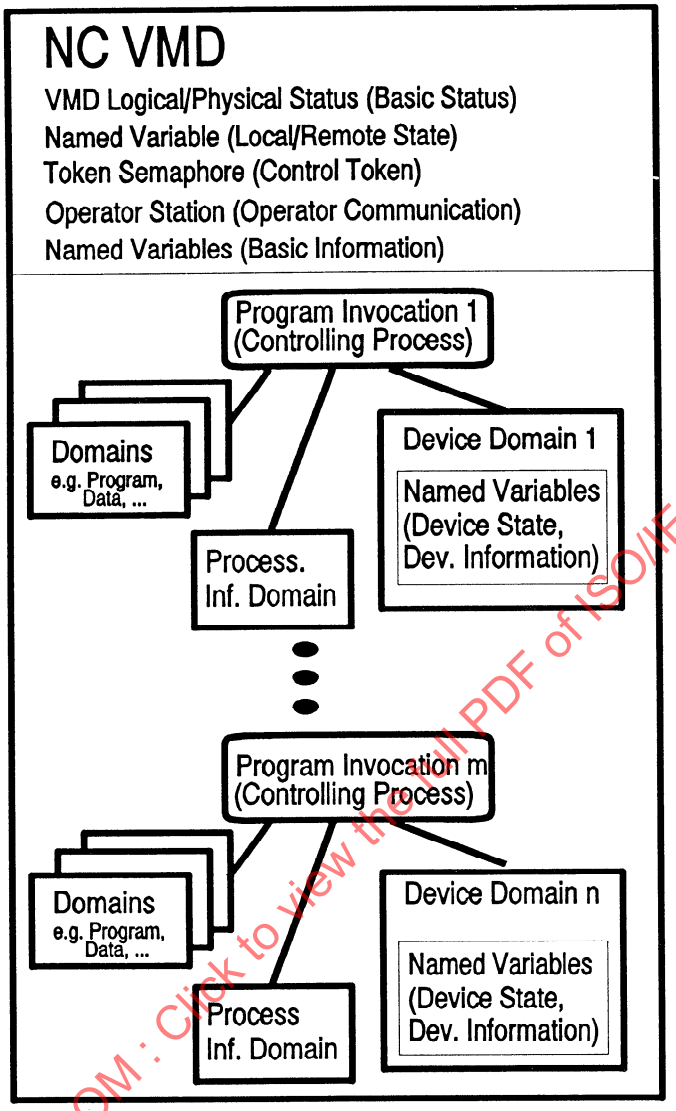

Figure 17: NC VMD Model

6.1.1.1 Basic Status

The Basic Status maps to the VMD Logical Status and to the VMD Physical Status.

6.1.1.2 Local/Remote State

The Local/Remote State maps to a VMD specific Named Variable Object with the name N_REMOTE.

6.1.1.3 Control Token

The control token maps to a Token Semaphore Object of the name N_CONTROL.

6.1.1.4 Device Image

A device image maps to a predefined domain object (Device Domain) with domain specific Named Variable objects which represent the physical resources and the state of the Device.

The device state maps to a Device Domain specific Named Variable N_DEV_STATE.

6.1.1.5 User Program

A user program is modeled as a domain object and is linked to a controlling process.

6.1.1.6 User Data

User data are modeled as domain objects and may be linked to a controlling process.

6.1.1.7 Controlling Process

The Controlling Process is modeled as a Program Invocation object linked to the domain objects which are germane to the controlling process. The Controlling Process state maps to the Program Invocation state.

6.1.1.8 NC Information

The Basic Information, the Device Information and the Controlling Process Information are mapped to variables. The Basic Information maps to VMD specific Named Variables, the Device Information maps to device domain specific variables. The Process Information maps to domain specific variables. The Process Information domain which contains these variables is locally created with creation of the assigned Program Invocation and shall include the name of the Program Invocation.

Instead of accessing different variables separately, the NC-CS user may define Named Variable List objects consisting of the desired variables. This part of ISO/IEC 9506 provides Variable List objects for the Basic Information (Basic Information List), for the Device Information (Device Information List) and for the Process Information (Process Information List). The use of these objects, however, is optional.

6.1.1.9 NC Alarm Model

The NC Alarm Model maps to the MMS Event Model.

6.1.1.10 NC Data Store

The following object classes are used to provide the functionality of the data store as required by the NC Data Store Model:

a) NC Domain Object Class:
- NC Data Store Domain
(N_STORE_...)

b) Named Variable Object Class:

- Data Store Information
  (N_StoreType,
  N_MaxStoreEntries,
  N_MaxStoreCapacitiy,
  N_CurrentEntries,
  N_RemainingEntries,
  N_CurrentCapacityUsed,
  N_RemainingCapacity)
- Data Store Entry

c) Named Variable Type Object Class:
- Data Store Entry Type
  (N_DataStoreEntry)

d) Named Variable List Object Class:
- Data Store Entry List
  (N_EntryList)

6.1.1.11 Operator Interface Device

The Operator Interface Device maps to the Operator Station object in MMS. Several Operator Station objects may exist in an NC system.

6.1.2 NC VMD

6.1.2.1 NC VMD Object

Object: NC VMD

All MMS defined VMD attributes

Additional Detail - There are no additional attributes defined in this standard.

6.1.2.2 VMD Logical Status and VMD Physical Status

The VMD Logical and Physical Status produce the Basic Status according to the following table:

Table 2 - Status Coherence

Basic Status	VMD Logical Status	VMD Physical Status
NORMAL PRODUCTION	State Changes Allowed	Operational
LOCAL ACTION	any	Needs Commissioning, Inoperable REQUIRED Partially Operational

6.2 Definition Of NC Specific Objects Which Map To Domains

6.2.1 NC Domain Object

Object: NC Domain

All MMS defined Domain attributes

Additional Detail - There are no additional attributes defined in this standard.

6.2.2 Device Domain

The Device Domain represents the physical resources of a Device. It is a predefined domain which may be set up by the manufacturer, and which provides a naming space for domain specific Named Variables, such as Device State, Number of Axes, and so on.

6.2.3 Process Information Domain

The Process Information Domain contains the Process Information of the corresponding Controlling Process. This domain is locally created with creation of a Controlling Process and exists only during the lifetime of the same, that is, it is locally deleted with deletion of the Controlling Process.

The Attribute List of Domain References of the associated Program Invocation contains a reference to the Process Information Domain.

6.2.4 Program Domain

User programs usually contain a sequence of part execution steps and are coded in a high level language (such as ISO 6983). Such programs are modelled as domains, which provides the means to upload, download or delete user programs from the NC store, and enables an NC-CS user to assemble several user programs into one program invocation.

6.2.5 Data Domain

Data used by user programs may be modelled as domains or variables in domains, for example:

- Tool Data domain: manufacturer defined tooling information.
- Means of Production Data domain: manufacturer defined means of production information, such as fixture information.

Like program domains these objects may be uploaded, downloaded, or deleted from the NC store, and may be assembled, together with user programs, into a program invocation.

6.2.6 NC Data Store Domain

An NC Data Store domain object shall be created for each logical data store in the NC VMD. Such a domain shall contain certain domain specific objects, namely those which define the NC data store, and those which define domains assigned to the data store. The name of an NC Data Store domain shall begin with "N_STORE_...". The creation of an NC Data Store domain is a local matter. This standard does not map a logical data store to a physical data storage device.

The NC Data Store Domain list of subordinate objects includes the following:

a) Data Store Information - corresponds to the "Store Info" element of the NC Data Store model (see figure 18).

b) Data Store Entry Type - defines the content of "Data Info" element of the model.

c) Data Store Entries - corresponds to the "Data Info" element of the model.

d) Data Store Entry List - provides a list of Data Info element names.

The relationship between these objects and the NC Data Store model is shown in figure 18.

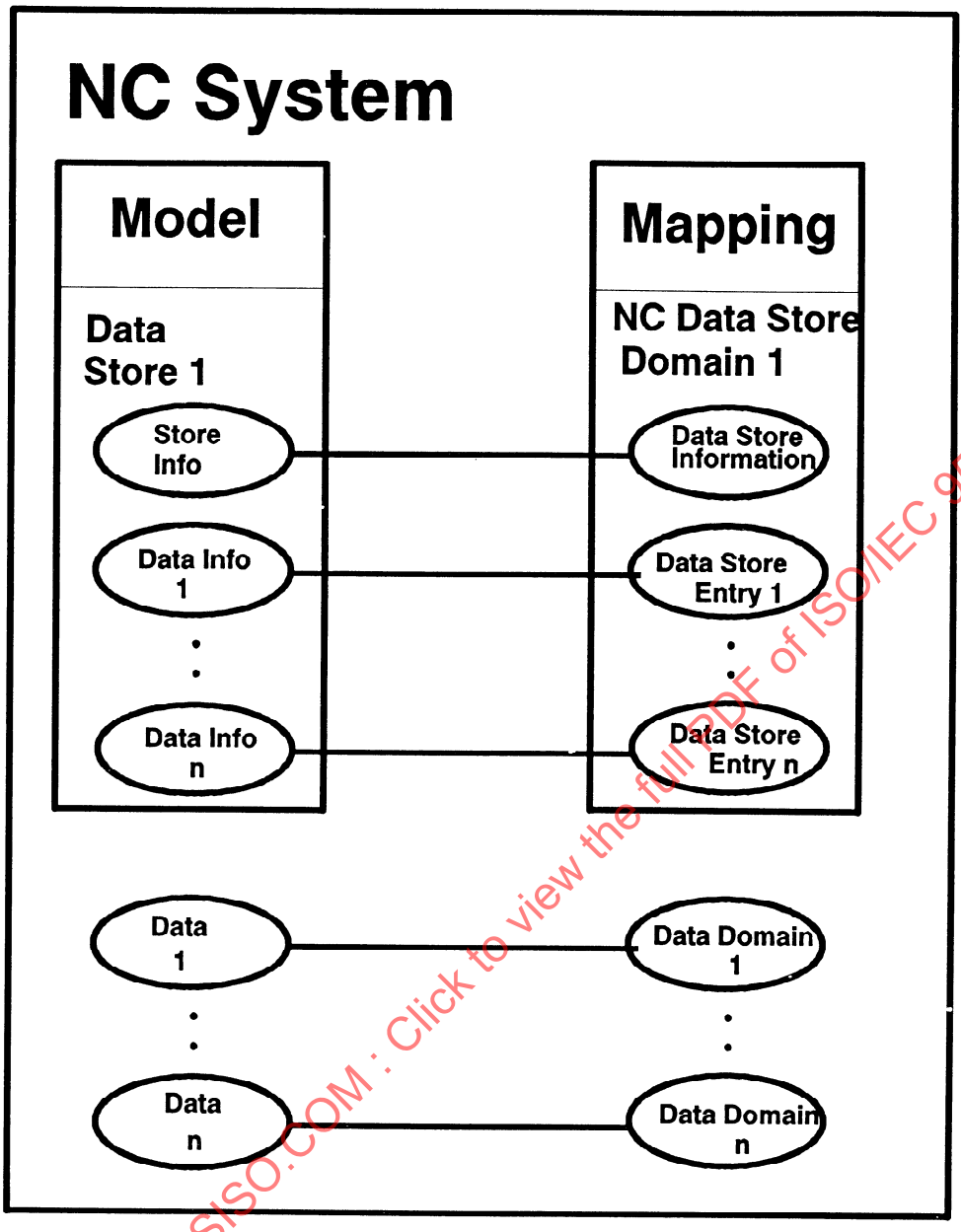

Figure 18: Data Store Mapping

The NC Data Store is normally not modified directly. Its objects are created and deleted locally as a result of the creation and removal of domains from the NC VMD. How the NC Data Store image is altered, that is, the creation of NC Data Store entries, and the assignment of objects which are NC Data Store domain specific, shall be a local issue. Following are the possible and visible side effects of an upload and download of domains in terms of the NC Data Store:

When a Domain state transition to READY occurs, a Data Store Entry is created in the NC Data Store domain associated with its Domain Content. The attributes of the Data Store Entry should be derived locally.
When a Domain state transition to NON-EXISTENT occurs, the associated Data Store Entry is removed from the NC Data Store domain.
The Data Store Information Variable values are also assumed to be modified when an assigned Domain state transition to READY or NON-EXISTENT occurs.
This part of ISO/IEC 9506 does not define the client usage of the values or existence of any NC Data Store objects. It is intended for the NC-CS user to use this information to achieve the functionality defined in clause 5 of this part of ISO/IEC 9506.

6.2.7 Mapping Of Other Objects To Domains

Naming conventions provide a grooping allowing an easy way of finding all of these objects, mostly vendor defined.

- Setup Data: These domains shall contain static machine parameters.

- Statistical Data: These domains shall contain Data which are defined dynamically during process and deal for example which part production times.

- Probe Data Table: These domains shall contain data which describe the probe and probe measuring data which may be exchanged with probe systems and are used in user programs.

6.3 Definition Of NC Specific Objects Which Map To Program Invocation

6.3.1 NC Program Invocation Object

Object: NC Program Invocation

All MMS defined Program Invocation Attributes

Additional Detail - There are no additional attributes defined in this standard.

6.3.2 Controlling Process Program Invocation

The Controlling Process State Model shall be mapped to the NC Program Invocation State Model. Transitions of the Controlling Process state may be the result of a Program Invocation service or of local action.

Figure 19 shows the transitions of the Controlling Process states which may occur as a positive result of Program Invocation services. The states and transitions of the Controlling Process map directly to the MMS Program Invocations states defined in ISO/IEC 9506/Part 1. To keep the diagram simple it does not show the transient states (STARTING, STOPPING, RESUMING, RESETTING). These transient states shall be used exactly as defined in ISO/IEC 9506/Part 1.

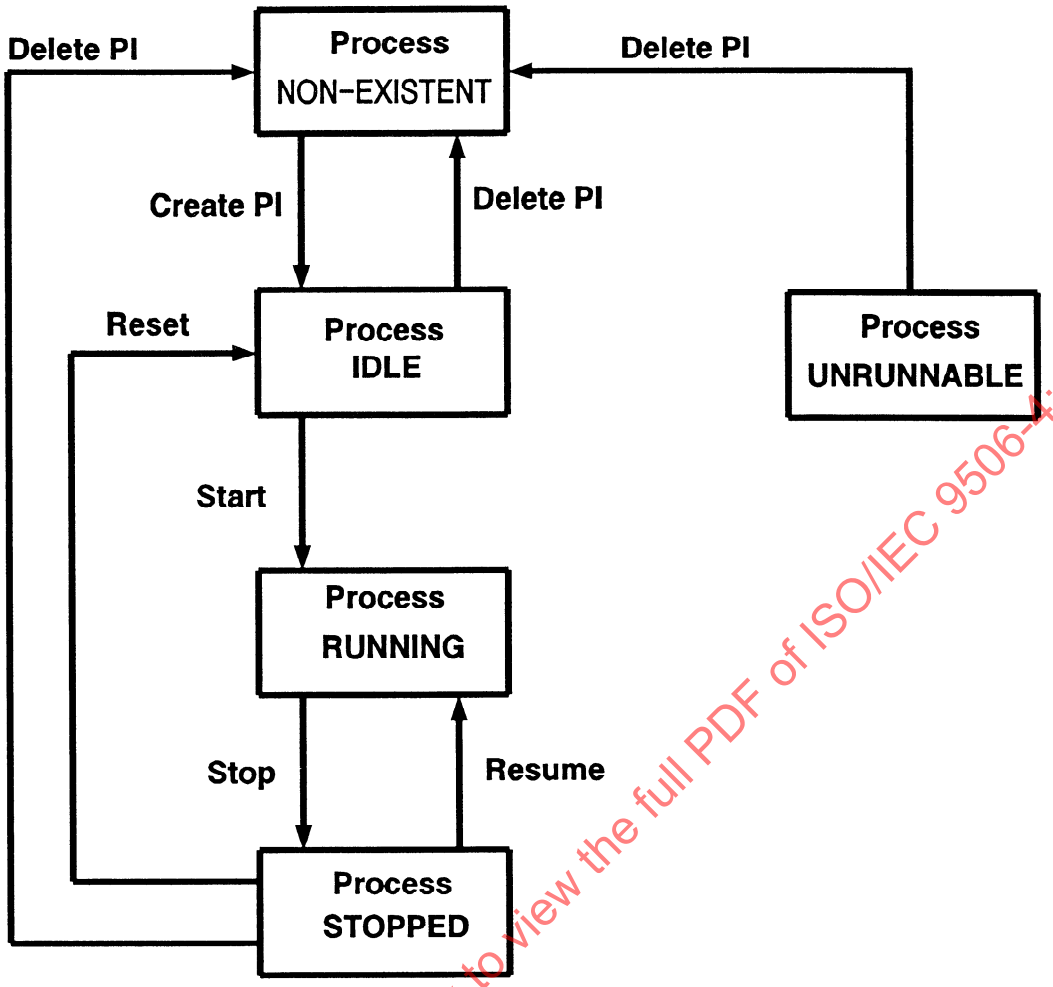

Figure 19: Transitions of the Controlling Process state as a result of Program Invocation services

Figure 17 shows one device domain linked to one Controlling Process. This configuration should be the normal case. But this standard does not restrict the VMD model to this one to one relationship.

Thus, an NC-CS user may create a second controlling process linking other programs and data to the same Device Domain while the first program is still executing. Figure 20 shows an example of an NC VMD with two Program Invocations linked to a single Device Domain.

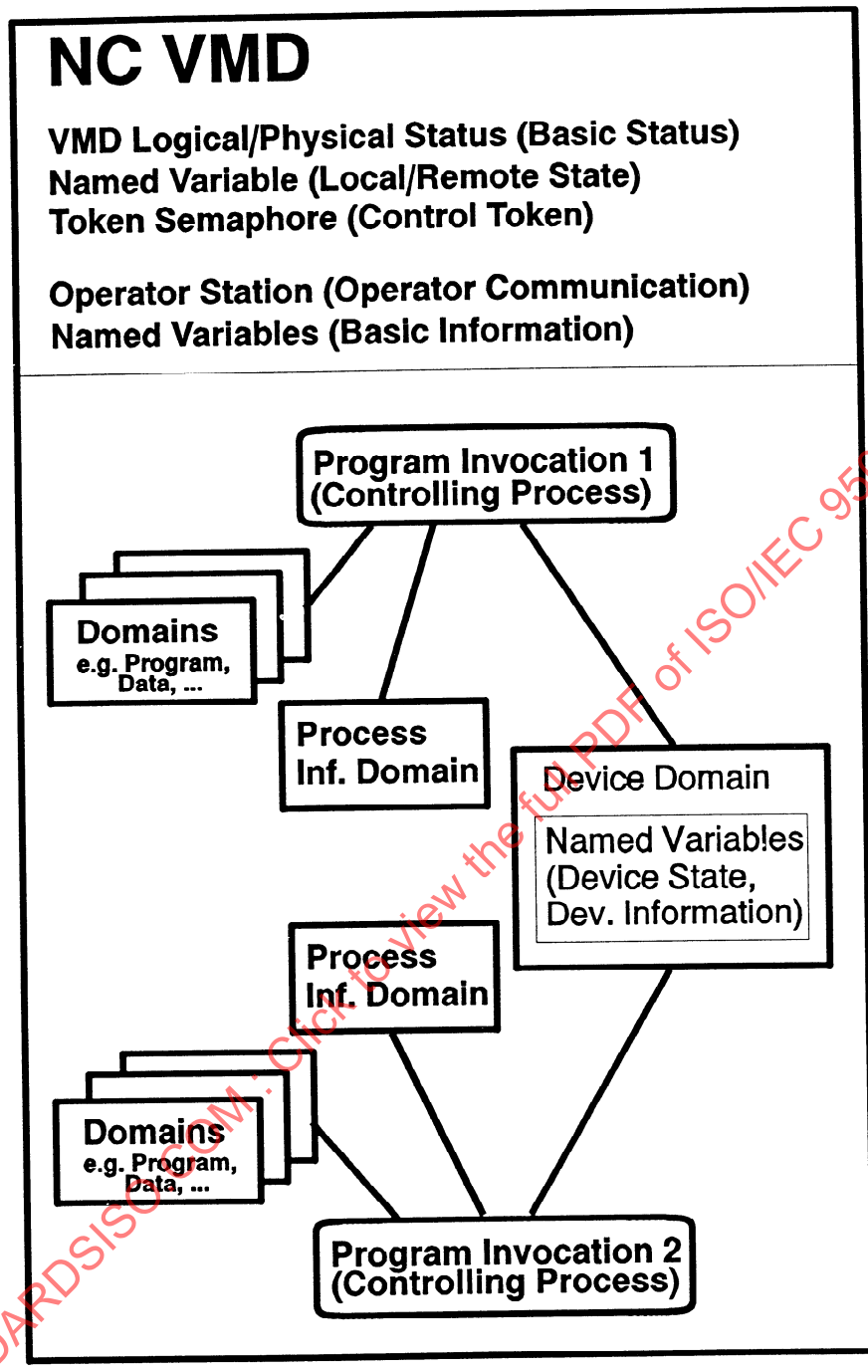

Figure 20: NC VMD Model (2 Program Invocations, 1 Device Domain)

There may also be situations which require that several devices be linked to one Controlling Process. Such a process is the Device Origin Setup procedure which may require access to more than one device. Figure 21 shows the example of an NC VMD with two Device Domains linked to one Program Invocation.

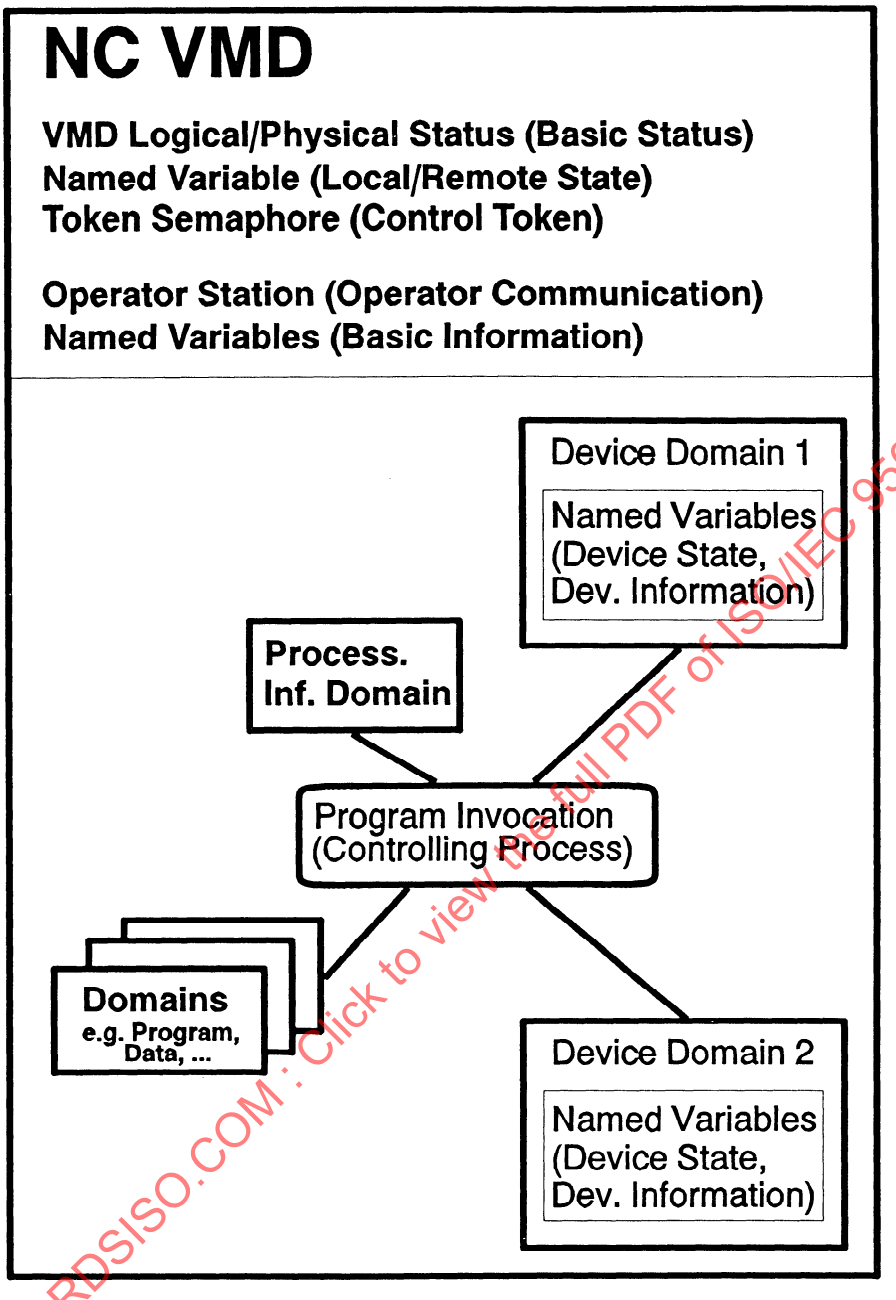

Figure 21: NC VMD Model (1 Program Invocation, 2 Device Domains)

6.4 Definition Of NC Specific Objects Which Map To Other MMS Abstract Objects

6.4.1 Definition Of NC Specific Objects Which Map To Named Variables

The MMS Variable Access services provide facilities by which typed variables of an MMS VMD may be accessed. MMS defines the Unnamed Variable object, the Named Variable object, and the Scattered Access object.

NOTE - Of these MMS variable objects, only the support of the Named Variable object is required for conformance with the NC VMD defined in this part of ISO/IEC 9506.

6.4.1.1 NC Named Variable Object

Object: NC Named Variable

All MMS defined Named Variable Attributes

6.4.1.2 Device State

The Device State maps to a Device Domain specific Named Variable. The name and structure of this variable is defined in clause 8.3.

6.4.1.3 Basic Information

The Basic Information maps to VMD specific Named Variables.

6.4.1.4 Device Information

The Device Information maps to Device Domain specific Named Variables.

6.4.1.5 Process Information

The Process Information maps to domain specific Named Variables. The Process Information domain which contains these variables only exists during the life of its corresponding Controlling Process, and so do these Process Information variables.

6.4.1.6 Data Store Information

Data Store Information shall be mapped to the following Named Variable Objects which are contained in the NC Data Store Domain:

a) N_StoreType - shall identify the type or other characteristic of the storage medium.

b) N_MaxStoreEntries - shall contain the maximum number of data store entry objects this data store may hold.

c) N_MaxStoreCapacity - shall contain the maximum octet count of data which this data store may hold.

d) N_CurrentEntries - shall contain the current number of data store entries contained in this data store.

e) N_RemainingEntries - shall contain the number of available entries which are unused.

f) N_CurrentCapacityUsed - shall contain the octet count of the space used by data currently contained in this data store.

g) N_RemainingCapacity - shall contain the octet count of the space which is currently unused in this data store.

6.4.1.7 Data Store Entry

Data store entries shall be mapped to Named Variable objects. The structure of these variables is defined in clause 8.3. Part of the identification of a data store entry shall be the name of the NC Domain object which the entry represents.

6.4.1.8 Offset model

The offset model shall be described by the following Named Variable Objects:

- N_ZeroAxes
- N_AxisValTrans
- N_Machine_CS
- N_ActProg_CS
- N_GeometryTrans

6.4.2 Definition Of NC Specific Objects Which Map To Named Variable List Objects

The MMS Named Variable List object provides for the assignment of one name for a group of (more or less) independent MMS Variables which may be accessed together. Thus, the NC-CS user may read the Device Information, by accessing its variable list name, without the need to separately access the variables contained therein.

6.4.2.1 NC Named Variable List Object

Object: Named Variable List

All MMS defined Named Variable List Attributes

6.4.2.2 Basic Information List

The Basic Information List shall contain one or more of the named variables representing Basic Information. Naming of the Basic Information List is defined in clause 8.5.

6.4.2.3 Device Information List

The Device Information List shall contain one or more of the named variables representing Device Information. Naming of the Device Information List is defined in clause 8.5.

6.4.2.4 Process Information List

The Process Information List shall contain one or more of the named variables representing Process Information. Naming of the Process Information List is defined in clause 8.5.

6.4.2.5 Data Store Entry List

The Data Store Entry List shall contain for each data store entry one named variable object representing the data store entry. Naming and structure of the Data Store Entry List are defined in clause 8.5.

The unique identifier of the Data Store Entry List shall be N_EntryList.

6.4.3 Definitions Of NC Specific Objects Which Map To Named Type Objects

Object: NC Named Type Objects

All MMS defined Named Type Object Attributes

NOTE - This standard does not standardize any services for NC specific named type objects.

6.4.3.1 Data Store Entry Type

There shall be only one Named Type Object in the NC Data Store domain. This use of a single named type object shall allow implementers to extend the scope of the data store model to include attributes not defined in this standard. The only restriction on implementers is that they shall include the following data type attributes with their new data types. If this restriction is followed, the conforming client will be able to retrieve the attributes of the Data Store Entry type, and to access any Data Store Entry object via the read variable service using the standardized attributes.

Naming and structure of the Data Store Entry Type are defined in clause 8.6. Its unique identifier shall be N_DataStoreEntry.

The attributes of the Data Store Entry Type shall include:

a) Data Name - a visible string equal to the name of a Domain object whose content is assigned to this data store.

b) Data Type - a visible string identifying the format of the domain content as it would be uploaded or downloaded. This string shall be equal to "Non-Coded" if the non-coded choice of loadData is used, and equal to the name of the OBJECT IDENTIFIER of the syntax used if the coded choice of load Data is used.

c) Data Size - a number representing the count of octets assigned to the domain which this entry represents, from the N_CurrentCapacityUsed attribute of the data store domain.

d) Creation Date - a value representing the time of creation of the domain which this entry represents.

e) Last Edit - a value representing the time of the last modification of the content of the domain which this entry represents.

f) Protection Mode - a visible string representing the access level or privilege required to access the content of the domain which this entry represents.

Default = empty.

g) Data Source -- a visible string representing the location from which the content of the domain which this entry represents, was obtained.
Default = empty.

6.4.4 Definition Of NC Specific Objects Which Map To Event Condition, Event Action And Event Enrollment Objects

6.4.4.1 NC Alarm Model

The NC Alarm Model maps to the MMS Event Model. The sending of an NC Alarm Notification maps to the MMS EventNotification Service. Acknowledgment of an NC Alarm Notification maps to the MMS AcknowledgeEventNotification Service. If the NC supports the request of summary information about the current status of NC Alarm Conditions, then it shall use the MMS GetAlarmSummary Service for this. NC-CS users may mask or unmask alarms by using the MMS AlterEventConditionMonitoring service. It should also be possible to mask or unmask alarms by local action.

6.4.4.1.1 NC Event Condition Object

Object: NC Event Condition

All MMS defined Event Condition Attributes

Additional Detail - No additional attributes are defined in this standard.

NC Event Conditions typically are predefined. This standard specifies that the MMS Event Condition Attributes shall be restricted to the following:

Attribute: Event Condition Name = NC Alarm Name
Attribute: Event Condition Class = MONITORED
Attribute: State = NC Alarm State
Attribute: Severity = NC Alarm Severity

6.4.4.1.2 NC Event Enrollment Object

Object: NC Event Enrollment

All MMS defined Event Enrollment Attributes

Additional Detail - No additional attributes are defined in this standard.

NC Event Enrollments typically are created in the connection establishment phase by local means. This standard specifies that the MMS Event Enrollment Attributes be restricted to the following:

Attribute: Enrollment Class = NOTIFICATION
Attribute: Alarm Acknowledgment Rule = NC Alarm Acknowledgment Rule

6.4.5 Mapping of the Local/Remote State to MMS objects

The Local/Remote State shall be mapped to a VMD specific Named Variable Object with the name N_REMOTE. The variable shall be of the type boolean.

Table 3 - Relation between the Local/Remote State and the Value of the Variable N_Remote

Local/Remote State	N_REMOTE
Local Control	False
Remote Control	True

An NC-CS user may initiate a transition from Remote Control to Local Control by sending a MMS Write.Request (N_REMOTE = False). If the transition succeeds, a Write.Response(+) shall be sent by the NC System. If the transitions fails, a Write.Response(-) shall be sent.
An NC-CS user initiating a transition from Remote Control to Local Control shall own the N_CONTROL semaphore.

For a transition from Local Control to Remote Control local action by the operator shall be required.

6.5 Definition Of New MMS Abstract Objects To Support Other NC Specific Objects

There are no new MMS abstract objects defined in this standard.

7 Services and Protocol

7.1 Numerical Control application context definition

For the purpose of being able to use an application which only contains the ACSE and MMS as ASEs, this part of ISO/IEC 9506 uses the object identifier value and the object descriptor value defined in 17.12 of ISO/IEC 9506-2:1990.

7.2 Numerical Control abstract syntax definition

This part of ISO/IEC 9506 assigns the ASN.1 object identifier value

{iso standard 9506 part(4) mms-nccs-syntax-version1(1)}

to the abstract syntax definied in this clause.

7.3 Use of the MMS services

7.3.1 Numerical Control specific ASN.1 module definition

The MMS services and protocol were developed to be used by a wide range of manufacturing devices. This clause defines the numerical control services and protocol for those elements identified as requiring Companion Standard definition in ISO/IEC 9506-2. These definitions shall be used when the abstract syntax defined in this part of ISO/IEC 9506 is negotiated.

All ASN.1 definitions provided in this part of ISO/IEC 9506 are part of the ASN.1 Module "ISO-9506-MMS-NCCS-1".

The beginning and closing statements indicating that each ASN.1 definition provided is a part of this module is omitted in order to make the reading of the document easier. Each ASN.1 definition provided implicitly contains the statement:

```
ISO-9506-MMS-NCCS-1
  {iso standard 9506 part(4) mms-nccs-module-version1(2)}
DEFINITIONS ::= BEGIN
```

at the beginning of the definition and contains the keyword "END" at the end of gwe definition.

NOTE - ISO-95aw-MMS-NCCS-1 represents the revision number 1 of the Companion Standard for Numerawal Control (ISO/IEC 9506-4).

```
IMPORTS
        MMSpdu,
```

```
        ServiceSupportOptions,
        ParameterSupportOptions,
        Integer16
    FROM MMS-General-Module-1
    {iso standard 9506 part(2) mms-general-module-version1(2)};
```

7.3.2 Program invocation management services and protocol

This subclause describes the services and protocol for the Numerical Control specific usage of Program invocation management operations.

7.3.2.1 CreateProgramInvocation service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.1.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.3.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.1.2 CreateProgramInvocation protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

```
CS-CreateProgramInvocation-Request ::= NULL
CS-CreateProgramInvocation-Response ::= NULL
```

7.3.2.2 Start service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.2.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.4.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.2.2 Start protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

CS-Start-Request ::= NULL
CS-Start-Response ::= NULL

7.3.2.3 Stop service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.3.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.5.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.3.2 Stop protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

CS-Stop-Request ::= NULL
CS-Stop-Response ::= NULL

7.3.2.4 Resume service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.4.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.6.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.4.2 Resume protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

CS-Resume-Request ::= NULL
CS-Resume-Response ::= NULL

7.3.2.5 Reset service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.5.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.7.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.5.2 Reset protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

```
CS-Reset-Request ::= NULL
CS-Reset-Response ::= NULL
```

7.3.2.6 Kill service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.6.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.8.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.6.2 Kill protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

```
CS-Kill-Request ::= NULL
CS-Kill-Response ::= NULL
```

7.3.2.7 DeleteProgramInvocation service

This part of ISO/IEC 9506 defines no additional parameters.

7.3.2.7.1 Extended service procedure

The service procedure is conditioned on whether the NC-CS user has control of the N_CONTROL semaphore.

a) The error checks of the service procedure in 11.9.2 of ISO/IEC 9506-1:1990 shall be performed.

b) If the requesting NC-CS user does not own the N_CONTROL semaphore, a Result(-) shall be returned.

7.3.2.7.2 DeleteProgramInvocation protocol

This part of ISO/IEC 9506 does not define any syntax extensions.

```
CS-DeleteProgramInvocation-Request ::= NULL
CS-DeleteProgramInvocation-Response ::= NULL
```

7.3.3 Other productions

The following productions, required for the MMS-General-Module (See A.1 of ISO/IEC 9506-2:1990), are not used in this part of ISO/IEC 9506, and are set equal to NULL. The services and protocol corresponding to these productions are fully defined in ISO/IEC 9506-1 and ISO/IEC 9506-2, and do not require numerical control specific extensions. They may be used directly in the abstract syntax defined in this part if ISO/IEC 9606.

```
CS-Status-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-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-GetNameList-Request ::= NULL

CS-Status-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-GetProgramInvocationAttributes-Response ::= NULL

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-AdditionalObjectClasses ::= NULL

EN-Additional-Detail ::= NULL

EE-Additional-Detail ::= NULL

JOU-Additional-Detail ::= NULL

AS-Filter ::= NULL

7.4 Definition and use of Application-specific Services

There are no NC specific Services defined in this standard, other than the NC specific Initiate service.

AdditionalService-Request ::= NULL

AdditionalService-Response ::= NULL

AdditionalService-Error ::= NULL

AdditionalUnconfirmedService ::= NULL

7.5 The Initiate Service and Protocol

Two related NC-specific parameters are defined by this part of ISO/IEC 9506 for the MMS Initiate service. For the MMS Initiate-RequestPDU this standard defines the CSRequestDetail parameter, and for the MMS Initiate-ResponsePDU, the CSResponseDetail parameter.

7.5.1 Definition of NC-specific Initiate Service

For this service the following additional parameters are required to establish the NC context, and identify the functionality to be used in that context.

Table 4: NC-specific CS parameters for Initiate

Parameter Name	Req	Ind	Rsp	Cnf
Proposed Version Number	M	M		
Proposed Parameter CBB	M	M		
Services Supported Calling	M	M		
Negotiated Version Number			M	M(=)
Negotiated Parameter CBB			M	M(=)
Services Supported Called			M	M

Proposed version number - is the highest version number which the requester supports

Negotiated version number - is the version number which shall be used during the association.

Proposed Parameter CBB - is the set of parameter building blocks which the calling MMS-user shall support. The value in the indication primitive shall be the intersection of the set requested, and the set supported by the MMS-provider.

Services Supported calling - is the set of services which the calling MMS-user shall support. The value in the indication primitive shall be the intersection of the set in the request primitive, and the set supported by the MMS-provider.

Services Supported called - is the set of services which the called MMS-user shall support. The value in the confirm primitive shall be the intersection of the set in the response primitive, and the set supported by the MMS-provider.

The definition of "supported" shall be as defined in MMS.

The version number of this version of ISO/IEC 9506-4 shall be 1.

7.5.2 Definition of NC-specific Initiate Protocol

```
   InitRequestDetail ::= SEQUENCE {
     proposedVersionNumber          [0] IMPLICIT Integer16,
     proposedParameterCBB           [1] IMPLICIT ParameterSupportOptions,
servicesSupportedCalling       [2] IMPLICIT ServiceSupportOptions          }

   InitResponseDetail ::= SEQUENCE {
     negotiatedVersionNumber        [0] IMPLICIT Integer16,
     negotiatedParameterCBB         [1] IMPLICIT ParameterSupportOptions,
servicesSupportedCalled        [2] IMPLICIT ServiceSupportOptions          }
```

7.6 End of Module

The following END statement closes the module, opened by begin in section 7.1

```
END
```

8 Standardized NC Specific Objects

NC specific objects are standardized in this clause by standardizing the name attributes of these objects. All names and name prefixes which are standardized in this part of ISO/IEC 9506 shall begin with a two character sequence containing a capital letter "N" in the first character position of the name and an underscore character, "_", in the second character position of the name.

While not all NC specific objects standardized herein may be supported by all implementations, those which are supported shall be listed in the appropriate PICS entry (see the conformance clause).

If an implementation uses an object whose semantics match the semantics of any standardized NC specific object defined herein, then such implementation shall use the standardized object, rather than a privately defined object with a non-standardized name. Conversely, implementations may not use any standardized name in any way other than defined in this part of ISO/IEC 9506.

8.1 Domain Objects

The name of each of the standardized NC specific domain objects defined in the following subclauses, shall be constructed in the way specified.

8.1.1 Device Domain

The name of any device domain shall begin with the six character prefix "N_DEV_", complemented by a user or implementer defined device identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_DEV_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable = FALSE
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.2 Program Domain

The name of any program domain shall begin with the six character prefix "N_PRG_", complemented by a user or implementer defined machine program identifier.

```
Object: Domain
```

```
Key Attribute:  Domain Name = "N_PRG_..."

        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.3 Process Information Domain

The name of any process information domain shall begin with the six character prefix "N_PID_", complemented by its associated program invocation identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_PID_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.4 Tool Data Domain

The name of any tool data domain shall begin with the six character prefix "N_TLD_", complemented by the user or implementer defined tool data table identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_TLD_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.5 Means Of Production Data Domain

The name of any means of production data domain shall begin with the six character prefix "N_MOP_", complemented by a user or implementer defined identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_MOP_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.6 Setup Data

The name of any setup data domain shall begin with the six character prefix "N_SET_", complemented by a user or implementer defined machine setup data identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_SET_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.7 Statistical Data

The name of any statistical data domain shall begin with the six character prefix "N_SDD_", complemented by a user or implementer defined statistical data identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_SDD_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.8 Probe Data Table

The name of any probe data table shall begin with the six character prefix "N_PRB_" complemented by a user or implementer defined probe data table identifier.

```
Object: Domain

Key Attribute:  Domain Name = "N_PRB_..."
        Attribute:  List of Capabilities
        Attribute:  State
        Attribute:  MMS Deletable
        Attribute:  Sharable
        Attribute:  Domain Content
        Attribute:  List of Subordinate Objects
        Attribute:  List of Program Invocation References
        Attribute:  Upload in Progress
```

8.1.9 NC Data Store Domain

The name of any NC Data Store domain shall begin with the seven character prefix "N_STORE_", optionally complemented by a user or implementer defined store identifier.

```
Object:  Domain

Key Attribute:  Domain Name                        = "N_STORE_..."
      Attribute:  List of capabilities
      Attribute:  State                            = READY
      Attribute:  MMS Deletable                    = FALSE
      Attribute:  Sharable                         = FALSE
      Attribute:  Domain Content
      Attribute:  List of Subordinate Objects      = {
                        N_StoreType,
                        N_MaxStoreEntries,
                        N_MaxStoreCapacity,
                        N_CurrentEntries,
                        N_RemainingEntries,
                        N_CurrentCapacityUsed,
                        N_RemainingCapacity,
                        N_DataStoreEntry,                    -- NOTE 1
                        N_EntryList,
                        data store entries                   -- NOTE 2
                        }
      Attribute:  List of Program Invocation References = {}  -- NOTE 3
      Attribute:  Upload in Progress
```

NOTE 1 - Implementers may substitute a different named type object which includes the attributes of N_DataStoreEntry.

NOTE 2 - Data store entries may also be of an extended type defined by a substitute named type object.

NOTE 3 - The List of program invocation references shall be an empty list.

8.2 Program Invocation Objects

Each of the standardized NC specific objects defined in the following subclauses, which are mapped to Program Invocation objects, shall be given the name specified.

8.2.1 N_Referencing

The Controlling Process Program Invocation object with the standardized name "N_Referencing" shall perform the device origin setup procedure.

```
Object: Program Invocation

Key Attribute: Program Invocation Name      = "N_Referencing"
        Attribute:  State
        Attribute:  List of Domain References
        Attribute:  MMS Deletable                = FALSE
        Attribute:  Reusable                     = TRUE
        Attribute:  Monitor                      = FALSE
        Attribute:  Event Condition Reference
        Attribute:  Event Action Reference
        Attribute:  Event Enrollment Reference
        Attribute:  Execution Argument
```

8.3 Named Variable Objects

The name of each of the standardized NC specific Named Variable objects defined in the following subclauses, shall either be given the exact name prescribed or a name constructed in the way specified, as the case may be. The values of some of the "type description" attributes may be of an arbitrary range, in which case the range of values (as well as the size of the attribute field) shall be determined and specified by the implementer.

8.3.1 NC VMD Specific Named Variable Objects

8.3.1.1 N_REMOTE

The Named Variable object with the standardized name "N_REMOTE" represents the Local/Remote State of the NC System.

```
Object: Named Variable

Key Attribute: Variable Name      = VMD-specific "N_REMOTE"
   Attribute:  MMS Deletable      = FALSE
   Attribute:  Type Description   = boolean
   Attribute:  Access Method      = locally defined
```

8.3.2 NC Domain Specific Named Variable Objects

The following Named Variable objects shall be of domain-specific scope. More specifically, the scope is that of the device domain, because these objects are device related. Because of this, the name of the corresponding device domain shall be included in the object name.

8.3.2.1 N_NumAxes - Number Of Machine Axes

The Named Variable object with the standardized name "N_NumAxes" contains the "number of axes" value. It may be used within other configuration related data structures of the device.

```
Object: Named Variable

Key Attribute: Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_NumAxes"}
        Attribute:  MMS deletable     = FALSE
        Attribute:  Type Description = unsigned 8
        Attribute:  Access Method     = locally defined
```

The NC-CS user shall have only read access to this variable.

8.3.2.2 N_ZeroAxes - Axes Zero Offsets

The Named Variable object with the standardized name "N_ZeroAxes" contains the zero offset values for all axes of the device. The number of zero offsets supported within an NC system is implementation dependent, and (along with the number of axes) it shall be specified for each device in the appropriate PICS entry.

```
Object: Named Variable

Key Attribute:      Variable Name = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_ZeroAxes"}
    Attribute: MMS Deletable      = FALSE
    Attribute: Type Description   = structure {
            components {
                {componentName   "on",                  -- Zero offsets on
                 componentType   boolean },             -- Switch for all axes
                {componentName   "zeroOffsets",
                 componentType   array {
                     numberOfElements = n-NumAxes-value
                                                        -- one offset per axis
                     elementType     structure {
                        components {
                            {componentName   "nameOfAxis",
                             componentType   visible-string 1 },
                                                        -- "x", "y", "z", etc.
                     {componentName   "on",
                                                        -- Zero offsets on for
                                                        -- this axis
                             componentType   boolean },
                            {componentName   "zeroOffsets",
                             componentType   array {
                                  numberOfElements = number-of-zero-offsets,
```

```
                                                -- device specific constant
                                elementType   structure {
                                     components {
                                        {componentName   "on",
                                                         -- zero offset on/off
                                         componentType   boolean },
                                        {componentName   "zeroOffset",
                                                         -- zero offset value
                                         componentType   floating-point {
                                                         format-width 32,
                                                         exponent-width 8 }
                  } } } } } } }
    Attribute: Access Method      = locally defined
```

8.3.2.3 N_AxisValTrans - Axis Value Transformation

The Named Variable object with the standardized name "N_AxisValTrans" contains the axis value transformation matrix for each axis of the device. The number of axis value transformations supported within an NC system is implementation dependent, and (along with the number of axes) it shall be specified for each device in the appropriate PICS entry.

NOTE - If an NC supports only mirroring, for example, the array value "number-of-axis-value-transformations" within the structure of the N_AxisValTrans object shall be set to 1 and the component value "primitive" to 2 (mirroring), which should be READ ONLY. How to enter the component value shall be a local matter.

```
Object: Named Variable

Key Attribute:      Variable Name = domain-specific {
                                    domainID "N_DEV_...",
                                    itemID   "N_AxisValTrans" }
   Attribute:     MMS Deletable  = FALSE
   Attribute: Type Description  = structure {
      components {
         {componentName   "on",                 -- Transformations on
          componentType   boolean },            -- Switch for all axes
         {componentName   "transformations",
          componentType   array {
             numberOfElements = n-NumAxes-value
                                                -- one per axis
             elementType   structure {
                components {
                   {componentName   nameOfAxis,
                    componentType   visible-string 1 },
                                                -- "x", "y", etc.
                   {componentName   "on",       -- transformation on/off for axis
                    componentType   boolean },
                   {componentName   "transformationSequence",
                     componentType   array {    -- elements in the order of
                                                -- application
                        numberOfElements = number-of-axis-value-transformations,
                                                -- implementation dependent
                        elementType   structure {
                           components {
                              {componentName   "on",      -- primitive on/off
                               componentType   boolean },
                              {componentName   "primitive",
                               componentType   unsigned 8 },
                                                -- Translation(0),
                                                -- Scaling(1),
                                                -- Mirroring(2)
```

```
                                {componentName   "value",     -- parameter value
                                 componentType   floating-point {
                                                          -- ignored if M()
                                                      format-width 32,
                                                  exponent-width 8 } } } } }
                            } } } } }
        Attribute: Access Method          = locally defined
```

8.3.2.4 N_Machine_CS -- Machine Coordinate System

The Named Variable object with the standardized name "N_Machine_CS" shall be used as the origin of the machine coordinate system.

```
Object: Named Variable

Key Attribute:    Variable Name  = domain-specific {
                                       domainID "N_DEV_...",
                                       itemID   "N_Machine_CS" }
    Attribute: MMS Deletable     = FALSE
    Attribute: Type Description  = structure {
        components {
            {componentName   "x",
             componentType   floating-point {
                                  format-width 32,
                                  exponent-width 8 } },
            {componentName   "y",
             componentType   floating-point {
                                  format-width 32,
                                  exponent-width 8 } },
            {componentName   "z",
             componentType   floating-point {
                                  format-width 32,
                                  exponent-width 8 } }
            } }
    Attribute: Access Method     = locally defined
```

8.3.2.5 N_ActProg_CS -- Active Program Coordinate System

The Named Variable object with the standardized name "N_ActProg_CS" contains the identifier for the coordinate system of the currently active, or to be activated, machine program.

```
Object: Named Variable

Key Attribute:    Variable Name  = domain-specific {
                                       domainID "N_DEV_...",
                                       itemID   "N_ActProg_CS"
    Attribute: MMS Deletable     = FALSE
    Attribute: Type Description  = visible-string -32
              -- references the actual coordinate system, which may be one of
              -- the following:
              --      N_UNDEFINED_CS  if undefined
              --      N_DEFAULT_CS for default coordinate system
              --      N_MACHINE_CS for machine coordinate system
            --      Any other, such as N_Pallet1_CS for pallet 1 coordinate
              --      system
    Attribute: Access Method     = locally defined
```

8.3.2.6 N_GeometryTrans -- Geometry Transformation

The Named Variable object with the standardized name "N_GeometryTrans" contains the geometry transformation matrix for the device. The number of geometry transformations supported within an NC system is implementation dependent, and (along with the number of axes) it shall be specified for each device in the appropriate PICS entry.

NOTE - If an NC supports only mirroring, for example, the array value "number-of-geometric-transformations" within the structure of the N_GeometryTrans object shall be set to 1 and the component value "primitive" to 2 (mirroring), which should be READ ONLY. How to enter this component value shall be a local matter. If an NC supports only a predefined sequence of primitives, a write access to change the order of the primitives shall fail.

```
Object: Named Variable

Key Attribute:   Variable Name    = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_GeometryTrans" }
    Attribute: MMS Deletable      = FALSE
    Attribute: Type Description   = structure {
        components {
           {componentName    "on",                  -- Transformation on
            componentType    boolean},
           {componentName    "transformations",
            componentType    array {
                numberOfElements = number-of-geometry-transformations,
                                                    -- implementation dependent.
                elementType    structure {          -- Elements in the exact
                                                    -- order of transformations.
                   components {
                       {componentName   "on",    -- primitive on/off
                        componentType   boolean },
                       {componentName   "primitive",
                        componentType   unsigned 8 {
                                                    -- Translation (0),
                                                    -- Scaling (1),
                                                    -- Mirroring (2),
                                                    -- Rotation (3),
                       {componentName   "value1",
                        componentType   floating-point {
                                                    format-width 32,
                                                    exponent-width 8 } },
                       {componentName   "value2",
                        componentType   floating-point {
                                                    format-width 32,
                                                    exponent-width 8 } },
                       {componentName   "value3",
                        componentType   floating-point {
                                                    format-width 32,
                                                    exponent-width 8 } } }
                   } }
    Attribute:   Access Method    = locally defined
```

NOTE - The meaning of the components value1, value2, and value3 depends on the current value of the component primitive: In case of translation the 3 components represent a vector in terms of the coordinate system x, y, and z. In case of scaling for each coordinate axis a scaling factor may be specified. In case of rotation the angles about x, y and z axis may be specified. In case of mirroring the planes about which mirroring is effective may be selected. In this case for each component only the values 0 and 1 are valid. A value of 1 means mirroring about the plane. The xy plane may be selected by the component value1, the xz plane by the component value2, and the xz plane by the component value3.

8.3.2.7 N_FRL -- Feedrate Limit

The Named Variable object with the standardized name "N_FRL" shall contain the value of the maximum feedrate permitted for the associated device.
Object: Named Variable

```
Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_FRL"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = array {
                          numberOfElements = n_NumAxes-value,
                          elementType   floating-point {
                                              format-width 32,
                                              exponent-width 8 } }
    Attribute:  Access Method     = locally defined
```

8.3.2.8 N_FRO_ON -- Feedrate Override ON/OFF

The Named Variable object with the standardized name "N_FRO_ON" shall be used to activate feedrate override. If its value is TRUE, the current value contained in the variable N_FRO shall be applied to the program feedrate. Otherwise the local override value shall be active.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_FRO_ON"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = boolean
    Attribute:  Access Method     = locally defined
```

8.3.2.9 N_FRO - Feedrate Override

The Named Variable object with the standardized name "N_FRO" shall contain the feedrate override, in percent or actual value, to be applied to the programmed feedrate.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_FRO"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = structure {
        components {
             {componentName   "overridePerCent",
```

```
            componentType   boolean },          -- if TRUE then
                                                -- value given in percent
           {componentName   "overrideValue",
            componentType   floating-point {
                                format-width 32,
                                exponent-width 8 } }
           } }
    Attribute:  Access Method     = locally defined
```

8.3.2.10 N_OSP -- Optional Stop

The Named Variable object with the standardized name "N_OSP" shall be used to activate the optional stop mode. If its value is TRUE, the optional stop mode shall be active.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                      domainID "N_DEV_...",
                                      itemID   "N_OSP"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = boolean
    Attribute:  Access Method     = locally defined
```

8.3.2.11 N_RPO_ON - Rapid Override ON/OFF

The Named Variable object with the standardized name "N_RPO_ON" shall be used to activate rapid override. If its value is TRUE, the current value contained in the variable N_RPO shall be applied to the program rapid feedrate. Otherwise the local default value shall be active.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                      domainID "N_DEV_...",
                                      itemID   "N_RPO_ON"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = boolean
    Attribute:  Access Method     = locally defined
```

8.3.2.12 N_RPO - Rapid Override

The Named Variable object with the standardized name "N_RPO" shall contain the rapid override, in percent or value, to be applied to all programmed rapid moves, for those devices which do use separate overrides for rapid moves and machining feedrates.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                      domainID "N_DEV_...",
                                      itemID   "N_RPO" }
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = structure {
        components {
            {componentName   "overridePerCent",
             componentType   boolean },          -- if TRUE then
                                                 -- value given in percent.
            {componentName   "overrideValue",
             componentType   floating-point {
```

```
                                        format-width 32,
                                        exponent-width 8 } }
            } }
    Attribute:  Access Method     = locally defined
```

8.3.2.13 N_NumSpindles - Number Of Spindles

The Named Variable object with the standardized name "N_NumSpindles" contains the "number of spindles" value. It may be used within other spindle related, structured data objects of the device, such as spindle speed limits.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_NumSpindles" }
    Attribute:  MMS deletable     = FALSE
    Attribute:  Type Description  = unsigned 8
    Attribute:  Access Method     = locally defined
```

The NC-CS user shall have only read access to this variable.

8.3.2.14 N_SSL - Spindle Speed Limit

The Named Variable object with the standardized name "N_SSL" shall contain the value of maximum spindle speed permitted by the associated device.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_SSL" }
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description  = array {
                          numberOfElements = n-NumSpindles-value,
                          elementType   floating-point {
                                              format-width 32,
                                              exponent-width 8 } }
    Attribute:  Access Method     = locally defined
```

8.3.2.15 N_SSO_ON - Spindle Speed Override ON/OFF

The Named Variable object with the standardized name "N_SSO_ON" shall be used to activate spindle speed override. If its value is TRUE, the current value of the variable N_SSO shall be applied to the programmed spindle speed. Otherwise the local (default) spindle speed override value shall be active.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_SSO_ON" }
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description  = boolean
    Attribute:  Access Method     = locally defined
```

8.3.2.16 N_SSO - Spindle Speed Override

The Named Variable object with the standardized name "N_SSO" shall contain the override value, in percent or value, to be applied to the programmed spindle speed.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_DEV_...",
                                        itemID   "N_SSO" }
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = array {
                        number of elements = n-NumSpindles-value,
                        elementType   structure {
                            components {
                              {componentName  "overridePerCent",
                               componentType  boolean },  -- if TRUE then
                                                          -- value given in
                                                          -- percent
                              {componentName   "overrideValue",
                               componentType   floating-point {
                                                   format-width 32,
                                                   exponent-width 8 } } } }
    Attribute:  Access Method     = locally defined
```

8.3.2.17 N_DEV_STATE -- Device State

The Named Variable object with the standardized name "N_DEV_STATE" shall contain the device state.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID   "N_DEV_...",
                                        itemID     "N_DEV_STATE"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = unsigned 8
    Attribute:  Access Method     = locally defined

Possible values of this variable are:

        1 = POWER OFF
        2 = READY
        3 = NOT READY
```

The NC-CS user shall have only read access to this variable.

8.3.2.18 N_Referenced -- Device Origin Setup Query

The Named Variable object with the standardized name "N_Referenced" shall indicate the calibration state of the NC device. A value of TRUE shall indicate that the NC device is calibrated.

```
Object: Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID  "N_DEV_...",
                                        itemID    "N_Referenced"}
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = boolean
```

```
    Attribute:  Access Method     = locally defined
```

8.3.2.19 N_StoreType

The Named Variable object with the standardized name "N_StoreType" shall contain a visible string describing the type of storage medium, or other pertinent information about it.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                          domainID "N_STORE_...",
                                          itemID   "N_StoreType"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = visible-string -32
    Attribute:     Access Method  = locally defined
```

8.3.2.20 N_MaxStoreEntries

The Named Variable object with the standardized name "N_MaxStoreEntries" shall contain an unsigned integer specifying the maximum number of data store entry objects which may be assigned to this data store.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                          domainID "N_STORE_...",
                                          itemID   "N_MaxSgwreEntries"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.21 N_MaxStoreCapacity

The Named Variable object with the standardized name "N_MaxStoreCapacity" shall contain an unsigned integer specifying the maximum octet count of data which may be loaded into this data store.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                          domainID "N_STORE_...",
                                          itemID   "N_MaxStoreCapacity"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.22 N_CurrentEntries

The Named Variable object with the standardized name "N_CurrentEntries" shall contain an unsigned integer indicating the current number of data store entries assigned to this data store.

```
Object:  Named Variable
```

```
Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID   "N_CurrentEntries"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.23 N_RemainingEntries

The Named Variable object with the standardized name "N_RemainingEntries" shall contain an unsigned integer indicating the number of entries which remain available.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID   "N_RemainingEntries"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.24 N_CurrentCapacityUsed

The Named Variable object with the standardized name "N_CurrentCapacityUsed" shall contain an unsigned integer indicating the octet count of the data currently loaded into this data store.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID   "N_CurrentCapacityUsed"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.25 N_RemainingCapacity

The Named Variable object with the standardized name "N_RemainingCapacity" shall contain an unsigned integer indicating the octet count which is currently unused in this data store.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID   "N_RemainingCapacity"}
    Attribute:     MMS Deletable  = FALSE
    Attribute: Type Description   = unsigned 32
    Attribute:     Access Method  = locally defined
```

8.3.2.26 Data Store Entry

This part of ISO/IEC 9506 does not standardize any NC-specific Data Store Entry objects. However, the following template should be used by an implementer to describe those entries that are required by the implementation.

```
Object:  Named Variable

Key Attribute:  Variable Name     = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID    "..." } -- the name of the
                                                          -- referenced domain.
    Attribute:  MMS Deletable     = FALSE
    Attribute:  Type Description = typeName {
                                        domain-specific {
                                                domainID  "N_STORE_...",
                                                 -- same as above
                                                 -- domainID
                                    itemID    "N_DataStoreEntry" }
    Attribute:    Access Method   = locally defined
```

The components of the data store entry shall be the same as the components of the Data Store Entry Type named type object present in the NC Data Store Domain.

8.4 Scattered Access Objects

This part of ISO/IEC 9506 does not standardize any NC-specific Scattered Access objects.

8.5 Named Variable List Objects

8.5.1 NC VMD-specific Named Variable List Objects

8.5.1.1 Basic Information List

The Named Variable List object with the standardized name "N_INF" shall identify the List of Variables which contain the names of one or more Named Variable objects representing Basic Information.

```
Object: Named Variable List

Key Attribute: Variable List Name = VMD-specific "N_INF"
        Attribute:  MMS Deletable
        Attribute:  List of Variable
```

8.5.2 NC Domain-specific Named Variable List Objects

8.5.2.1 Device Information List

The Named Variable List object with the standardized name "N_DEVINF" shall identify the list of variables which contain the names of one or more Named Variable objects representing device information.

```
Object: Named Variable List

Key Attribute: Variable List Name = domain-specific {
                                        domainID  "N_DEV_...",
                                        itemID    "N_DEVINF"}
        Attribute:  MMS Deletable
        Attribute:  List of Variable
```

8.5.2.2 Data Store Entry List

The Named Variable List object with the standardized name "N_EntryList" shall contain the names of the Named Variable objects of type Data Store Entry from the specific data store.

```
Object:  Named Variable List

Key Attribute:  Variable List Name = domain-specific {
                                        domainID "N_STORE_...",
                                        itemID   "N_EntryList"}
        Attribute:  MMS Deletable  = FALSE
        Attribute:  List of Variable {
                        name = domain-specific {
                                domainID  "N_STORE_...", -- same store as above.
                                itemID    "..." }        -- name of a Data
                                                         -- Store Entry object.
         }  -- repeated for each Data Store Entry object in the data store.
```

8.5.2.3 Controlling Process Information List

All Named Variable List objects containing lists of "Controlling Process Information" shall have the name "N_PRINF". Each such Variable List object shall identify a list containing the names of one or more Named Variable objects representing Controlling Process Information.

```
Object: Named Variable List

Key Attribute: Variable List Name  =  domain-specific {
                                          domainID "N_PID_...",
                              itemID   "N_PRINF"}
    Attribute:  MMS Deletable
    Attribute:  List of Variable
```

8.6 Named Type Objects

8.6.1 Data Store Entry Type

The Named Type object with the standardized name "N_DataStoreEntry" shall define the information content of a data store entry.

The named attributes of this object may be used to access information about a domain which is assigned to the Data Store domain which contains this named type object.
Example: Given a domain of name "N_PRG_1" assigned to a data store of name "N_STORE", then, a reference to the N_STORE (domain-specific) variable N_PRG_1.DataSize would yield the octet count assigned to the domain "N_PRG_1".

```
Object:  Named Type

Key Attribute: Type Name       = domain-specific {
                                    domainID "N_STORE_...",
                                    itemID   "N_DataStoreEntry"}
      Attribute: MMS Deletable = FALSE
      Attribute: Type Description structure {
                    components {
```

```
{componentName = "DataName",
 componentType = visible-string -32 },
{componentName = "DataType",
 componentType = visible-string -32 },
{componentName = "DataSizeValid",
 componentType = boolean },
{componentName = "DataSize",
 componentType = unsigned 32 },
{componentName = "CreationDateValid",
 componentType = boolean },
{componentName = "CreationDate",
 componentType = generalized-time },
{componentName = "LastEdit",
 componentType = generalized-time },
{componentName = "ProtectionMode",
 componentType = visible-string -32 },
{componentName = "DataSource"
 componentType = visible-string -32 } } }.
```

NOTE 1 - Implementers may substitute a different named type object which includes the attributes of N_DataStoreEntry.

NOTE 2 - Data store entries may also be of an extended type defined by a substitute named type variable.

8.7 Semaphore Objects

8.7.1 Control Token

The Token Semaphore object shall have the standardized name "N_CONTROL". This object shall be of VMD-specific scope.

```
Object: Semaphore

Key Attribute: Semaphore Name    = VMD-specific "N_CONTROL"
        Attribute:  MMS Deletable    = FALSE
        Attribute:  Class            = TOKEN
        Attribute:  Number of Tokens = 1
        Attribute:  Number of Owned Tokens
        Attribute:  List of Owners
        Attribute:  List of Requesters
        Attribute:  Event Condition References (none)
```

8.8 Operator Station Objects

This part of ISO/IEC 9506 does not define any names for Operator Station objects.

8.9 Event Condition Objects

This part of ISO/IEC 9506 does not define any names for Event Condition objects.

8.10 Event Action Objects

This part of ISO/IEC 9506 does not define any names for Event Action objects.

8.11 Event Enrollment Objects

This part of ISO/IEC 9506 does not define any names for Event Enrollment objects.

8.12 Journal Objects

This part of ISO/IEC 9506 does not standardize any NC-specific journal objects.

8.13 Other NC Specific Objects

This part of ISO/IEC 9506 does not standardize any other NC specific objects.

9 Conformance

9.1 Conformance Class Description

This part of ISO/IEC 9506 defines just one conformance class called NC which has the minimal functionality to establish or de-establish an association, or to reject it in case of a protocol error. Any NC system which supports the conformance class NC shall provide the following Services:

Initiate
Abort
Reject
Identify
Read

An NC System which supports the Conformance Class NC shall support VNAM as mandatory conformance building block.

In addition to the definition of the conformance class NC this part of ISO/IEC 9506 defines minimum requirements for each functional group listed in clause 5.2. Any NC system supporting such a functional group shall conform to these minimum requirements.

9.1.1 Minimum Requirements For The Functional Group Data Transfer

Any NC system supporting Variable Transfer shall provide in the server role the following MMS services:

Read
Write

Any NC system supporting Download of Domains shall provide in the server role the following MMS services:

InitiateDownloadSequence
DownloadSegment
TerminateDownloadSequence
RequestDomainDownload

Any NC system supporting Upload of Domains shall provide in the server role the following MMS services:

InitiateUploadSequence
UploadSegment
TerminateUploadSequence
RequestDomainUpload

Any NC system supporting additionally Upload and Download of Domains to a Third Party Fileserver shall provide in the server role the following MMS services:

LoadDomainContent
StoreDomainContent

9.1.2 Minimum Requirements For The Functional Group NC Information

Any NC system supporting NC information shall provide in the server role the following MMS services:

Read

9.1.3 Minimum Requirements For The Functional Group Process Management

Any NC system supporting Process Management shall provide in the server role the following MMS service:

Start

9.1.4 Minimum Requirements For The Functional Group Operator Interaction

Any NC system supporting a Operator Interface Device shall provide in the server role the following MMS service:

Output

9.1.5 Minimum Requirements For The Functional Group Alarm Processing

Any NC system supporting signaling of Alarms shall provide in the server role the following MMS Service:

EventNotification

Any NC system supporting acknowledgment of alarms shall provide in the server role the following MMS Service:

AcknowledgeEventNotification

9.1.6 Minimum Requirements For The Functional Group NC Data Store Management

Any NC System supporting Data Store Management shall provide in the server role the following MMS Services:

Read
DeleteDomain

9.2 Restrictions On MMS Optional Parameters

This part of ISO/IEC 9506 places no restrictions on MMS optional parameters.

9.3 Conformance To Standardized Objects

NC systems claiming conformance to this part of ISO/IEC 9506 shall support the followingg NC specific domain objects:

N_DEV_...

NC systems claiming conformance to this part of ISO/IEC 9506 shall support the following NC specific variable objects for each NC specific domain object N_DEV_...

N_DEV_STATE

NC systems controlled by a single NC-CS user (Supervisory Computer) shall support the following procedure of automatic allocation of the N_CONTROL semaphore:

For the purposes of satisfying the requirements of the NC system state model described in clause 5, of the MMS model mapping described in clause 6, and of the elements of the service procedures described in clause 7, the NC-CS user proposing or accepting establishment of an application association specifying the abstract syntax defined in this part of ISO/IEC 9506 shall have control of the N_CONTROL semaphore immmediately upon successful establishment of the association. No explicit MMS service procedures for semaphores need to be supported in order to satisfy the conformance requirements of this part of ISO/IEC 9506.

9.4 Addition To MMS PICS

The following extensions to the MMS PICS are required by this part of ISO/IEC 9506:

9.4.1 PICS For Functional Classes

The vendor of an NC system shall state which functional group (outlined in section 5.2) it supports.

Functional Group	supported ?
Variable Transfer	
Domain Download	
Domain Upload	
Third Party Up/Download	
NC Information	
Process Management	
Operator Interaction	
Signaling of Alarms	
Acknowledgment of Alarms	
NC Data Store Management	

9.4.2 PICS For Standardized Objects

The vendor of an NC system shall state which standardized objects it supports.

Domain Objects
N_DEV_...
N_PRG_...
N_PID_...
N_TLD_...
N_MOP_...
N_SET_...
N_SDD_...
N_PRB_...
N_STORE_...

Program Invocation Objects
N_Referencing

Named Variable Objects (VMD-specific)
N_REMOTE

Named Variable Objects (domain-specific to the Device Domain)
N_NumAxes
N_ZeroAxes
N_AxisValTrans
N_Machine_CS
N_ActProg_CS
N_GeometryTrans
N_FRL
N_FRO
N_FRO_ON
N_OSP
N_RPO
N_RPO_ON
N_NumSpindles
N_SSL
N_SSO
N_SSO_ON
N_DEV_STATE

Named Variable Objects (domain-specific to the Data Store Domain)
N_StoreTyp
N_MaxStoreEntries
N_MaxStoreCapacity
N_CurrentEntries
N_RemainingEntries
N_CurrentCapacityUsed
N_RemainingCapacity

Named Variable List Objects
N_DEVINF
N_EntryList
N_PRINF
N_INF

Named Type Objects
N_DataStoreEntry

Semaphore Objects
N_CONTROL

Annex A: Dynamic Download and Coded Domains (Normative)

When the size of the NC program exceeds the storage capacity of the NC controller, it is necessary that the program data be downloaded dynamically, such that the execution of the program and the program downloading may take place simultaneously.

This kind of operation is sometimes known as "Behind the Tape Reader" operation, because the NC program is obtained from an external device same as from a paper tape reader. Advanced systems may make use of such a capability by using several simultaneous dynamic downloads for simultaneous tasks.

When a dynamic download is requested, the CreateProgramInvocation-Request shall include in its list of domain references one or more linking domains whose names shall begin with the predefined prefix "N_DDL_". The content of this domain shall be the information needed for the NC system to begin the dynamic download.

NOTE - This domain may be created during the creation of the Program Invocation by local means with information derived from the name of the linking domain.

Service Procedure:

When a dynamic download is initiated via CreateProgramInvocation-Request, the list of domain references shall include the name of the linking domain beginning with the prefix "N_DDL_". If several simultaneous downloads are requested, one linking domain for each dynamic download may be used. The content of the linking domain provides the information needed to accomplish the dynamic download, such as the name of the dynamic domain, which will be downloaded and the name of the file containing th data to be loaded.

When issuing a Start on the Program Invocation, a RequestDomainDownload for all dynamic domains shall be issued by the Program Invocation. The value of the Domain Name and File Name parameter in this request is derived from the information in the linking domain. After all InitiateDomainDownload-Requests are confirmed positive (the dynamic domains are in the loading state), a Start-Response shall be sent.

If a download is not possible, a Start-Response(-) shall be issued with an error class 'VMD-state' and an error code 'domain-transfer-problem'.

After receiving a DownloadSegment-Confirmation with the More Follows equal to false, a TerminateDownloadSequence-Request shall be issued with the 'discard' parameter containing an error class 'resource' and an error code 'memory-unavailable'. This will lead to the deletion of the dynamic domain.

The responding NC-CS user confirms this request and finally returns a response to the RequestDomainDownload service.

The linking domain shall contain all information for the dynamic download. It may include such items as the following Named Variables with predefined names:

```
Object: Named Variable

Key Attribute: Variable Name     = domain-specific {
                                        domainID "N_DDL_...",
                                        itemID   "N_DomainName" }
        Attribute:      MMS Deletable     =
        Attribute:      Type Description = visible-string-32   -- MMS Identifier
        Attribute:      Access Method     = locally defined
```

This variable specifies the name of the dynamic domain, which shall be downloaded.

```
Object: Named Variable
Key Attribute: Variable Name     = domain-specific {
                        domainID "N_DDL_...",
                        itemID   "N_FileName" }
      Attribute:    MMS Deletable     =
      Attribute:    Type Description = octet-string  -- MMS FileName
      Attribute:    Access Method     = locally defined
```

This variable specifies the name of the file containing the data to be downloaded.

```
Object: Named Variable
Key Attribute: Variable Name     = domain-specific {
                                        domainID "N_DDL_...",
                                        itemID   "N_TPYName" }
        Attribute:      MMS Deletable     =
        Attribute:    Type Description = octet-string  -- MMS Application Reference
        Attribute:      Access Method     = locally defined
```

This variable may be used to request the NC system to load the dynamic domain from a third party.

Write Access to one of these variables shall own the N_CONTROL semaphore.

Any other information required for controlling the dynamic download shall be included in the same manner with user defined objects in this domain.

Annex B: Extension of ISO 9506-4 To Include NC Milling Machine Functionality (Normative)

B Introduction

This normative annex extends the general applicability of ISO 9506-4.
The structure of this annex is the same as that of the main part of this standard, such that the clauses are in alignment. However, not every clause in the main part of ISO 9506-4 may have a counterpart in this annex, because its scope does not require it.

B.1 Scope

This Annex provides the additional objects and standardized names which may be required when this standard is used in NC milling machine applications, specifically in stand-alone usage (as opposed to integrated into a flexible manufacturing system).

B.2 References

This annex does not use any references besides those which are listed in clause 2 of ISO 9506-4.

B.3 Definitions

This annex does not use any terms besides those which are defined in clause 3 of ISO 9506-4.

B.4 Symbols and Abbreviations

This annex does not use any symbols and abbreviations besides those which are listed in clause 4 of ISO 9506-4.

B.5 Application Description

In terms of this annex, NC milling machines may consist of the axes, tools, spindles, tool magazines, and user programs which control these entities. This is illustrated in figure 22.

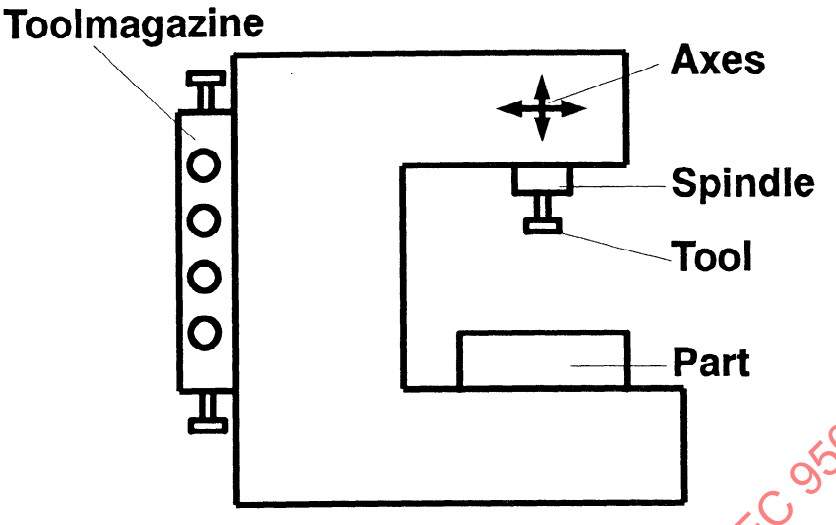

Figure 22: NC Milling Machine

B.5.1 Tool Magazines, Tool Holders, Tools, Edges

Milling tools consist of one or more edges. For most applications, in addition to the part program data, the NC controller may also require the geometrical description of the tools used - more precisely that of their cutting edges. The objects defined in this annex take into account the geometrical relationships which exist between tool holders, tools and edges. Also defined herein are objects which model the tool magazines or pallets containing the tools.

Tool magazine, tool holders, tools, and edges are illustrated in figure 23.

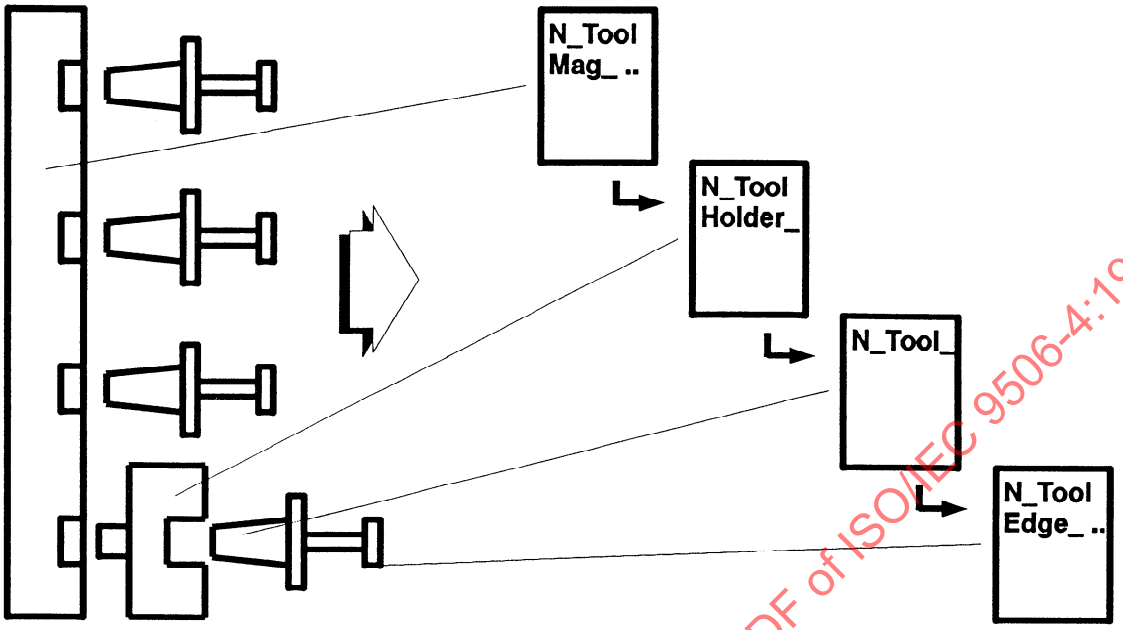

Figure 23: Example of Tool Magazine, Tool Holder, Tools, Edges

B.6 NC Milling Specific Context Mapping

B.6.2 Objects Which Map To Domains

B.6.2.1 Tool Data Object

The Tool Data domain described in section 8.1.4 in the main part of this standard may be used for up and downloading of named variable objects describing the tool magazine, tool holder, tool, and tool edge, which are defined in this section. These objects shall be subordinated objects of the tool data domain. In accord with the definitions of the main part of this standard, more than one real NC controlled device may exist within the NC VMD. Each of these devices may have its own tool storage, or they may have a common tool storage, or both.

NOTE - In case a pallet is used for tool storage, then the means of production data domain described in clause 8.1.5 of the main part of this standard should be used for up and downloading of the named variable objects tool holder, tool, and tool edge, which in this case shall be subordinated objects of the means of production data domain.

The names of Tool Data domains shall begin with the standardized prefix N_TLD_. An example of the tool data domain is shown in figure 24.

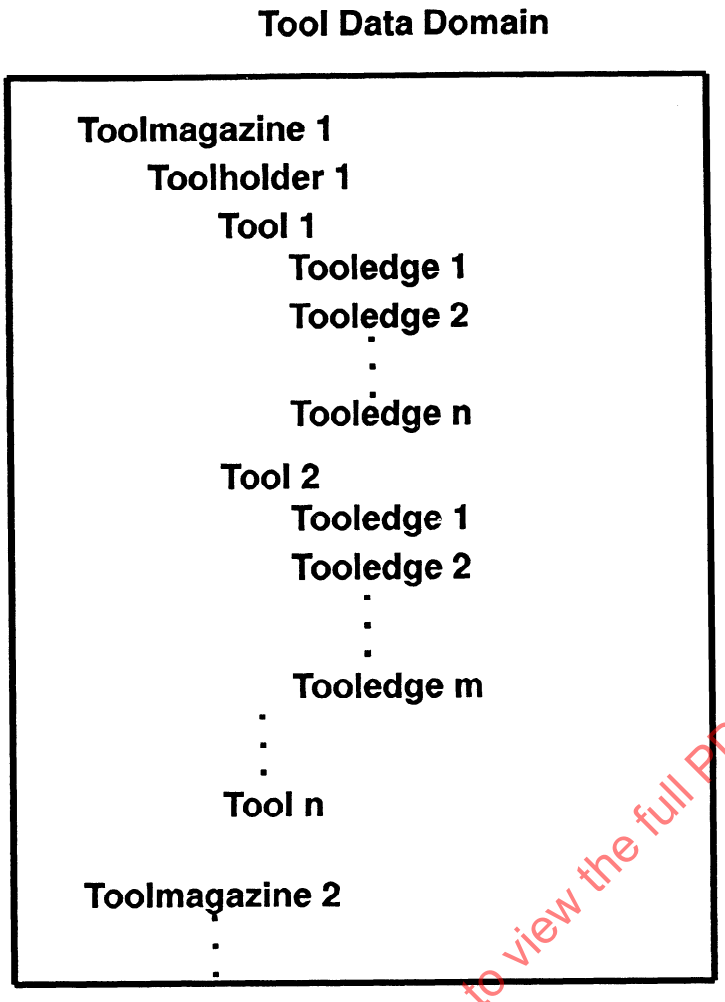

Figure 24: Tool Data Object

B.6.4.1 Objects which Map to Named Variables

B.6.4.1.1 Tool Magazine Object

This object describes the capability to store tools or tool holders on the machine, and which includes the mechanism for automatic interchange of tools or tool holders between tool storage and machine spindle.

Key Attribute: Tool Magazine ID

Attribute: List of Stored Objects
Attribute: Magazine Type
Attribute: Number of Positions

- Tool Magazine ID -- a selection from an implementation defined list of possible tool storage devices.
- List of Stored Objects -- a list of actual objects in the magazine. It shall be an array, with the array size equal to the Number of Positions.

- Magazine Type -- a selection from user defined tool magazine types.
- Number of Positions -- the total number of storage locations for tool objects or tool holder objects.

The Tool Magazine object is properly mapped into an MMS Named Variable object. The name and structure of this variable is described in clause B.8.3.2.1.

B.6.4.1.2 Tool Holder Object

This object contains information about a tool holder.

Key Attribute: Tool Holder ID

Attribute: Tool Holder Type
Attribute: Tool Holder Location ID
Attribute: Tool Holder Position
Attribute: List of Tools
Attribute: Tool Holder State
Attribute: List of Tool Holder Dimensions
Attribute: Process Information
Attribute: User Specific Information

- Tool Holder ID -- the serial number or user code associated with this tool holder.
- Tool Holder Type -- a selection from a user defined list of tool holder types.
- Tool Holder Location ID -- a back reference to the object which contains this tool holder, namely tool magazine or a pallet.
- Tool Holder Position -- a back reference to the position in the location object, such as a tool magazine pocket number or partition of a pallet.
- List of Tools -- a list of actual tool object identifiers.
- Tool Holder State -- one from a list of user defined conditions.
- List of Tool Holder Dimensions -- a user defined list of of tool holder measurements.
- Process Information -- a user defined list of information items needed for the process.
- User Specific Information -- a user defined list of information items about the tool holder.

The Tool Holder object shall be properly mapped into an MMS Named Variable object. The name and structure of this variable is described in clause B.8.3.2.2.

B.6.4.1.3 Tool Object

This object contains information about a tool.

Key Attribute: Tool ID

Attribute: Tool Type
Attribute: Tool Location ID
Attribute: Tool Position
^Xx Attribute: List of Tool Edges
Attribute: Tool State
Attribute: List of Tool Dimensions
Attribute: Process Information
Attribute: Tool Handling Information

- Tool ID -- serial number or user code associated with this tool.
- Tool Type -- a selection from a user defined list of tool types.
- Tool Location ID -- a back reference to the object which contains this tool, namely a tool magazine, a tool holder, or a pallet.

- Tool Position -- a back reference to the position within the location object, that is, tool magazine pocket number, or pallet partition.

- List of Tool Edges -- an array of the edges which are a part of this tool.
- Tool State -- one of the following conditions:

(0) In Use -- the tool is operating.
(1) Worn Out -- the remaining life is zero.
(2) Broken -- the tool has at least one broken edge.
(3) En Route -- the tool has been removed from or is en route to the magazine.
(4) Not Released -- the tool is not released for part operations
(5) Released -- the tool is released for part operations
(6) Needs Measurement -- the tool needs measurement
(7) Needs Measurement After First Operation -- the tool needs measurement after first operation

NOTE - The user may define additional states.

- List of Tool Dimensions -- a user defined list of the tool measurements.
- Process Information -- a user defined list of information items required for the process.
- Tool Handling Information -- a user defined list of information items required for the tool handling operations.

The Tool object shall be properly mapped into an MMS Named Variable object. Name and structure of this variable is defined in clause B.8.3.2.3.

B.6.4.1.4 Tool Edge Object

This object defines a cutting edge. For multiple edge tools there shall be a separate such object for each edge.

Key Attribute: Edge ID

Attribute:	Tool ID
Attribute:	List of Edge Offsets
Attribute:	Nominal Edge Life
Attribute:	Remaining Edge Life
Attribute:	User Specific Information

- Edge ID -- the serial number or user code associated with this edge.
- Tool ID -- back reference to the Tool object which contains this edge object.
- List of Edge Offset -- a user defined list of offsets.
- Nominal Edge Life -- a measure of the expected life for a new or sharpened edge. This shall be the starting value for actual life.
- Remaining Edge Life -- a measure of the remaining life of the edge.
- User Specific Information -- a list of user defined items containing information about this edge.

The Tool Edge object shall be properly mapped into an MMS Named Variable object. Name and structure of this variable is described in clause B.8.3.2.4.

B.8 NC Milling Specific Standardized Objects

B.8.2 Standardized Domains

B.8.2.1 Tool Data Domain

The Tool Data Domain is defined in clause 8.1.4 of the main part of this standard. Its name prefix shall be used for any domain associated with tool magazines, tool holders, tools, and tool edges.

B.8.3 Named Variable Objects

B.8.3.2 Domain-specific Standardized Named Variable Objects

B.8.3.2.1 Tool Magazine Object

The name of the Tool Magazine object shall begin with the standardized prefix "N_ToolMag_", complemented by a user or implementer defined tool magazine identifier.

```
        OBJECT: Named Variable

    Key Attribute:    Variable Name = domain-specific {
                                      domain ID "N_TLD_....",
                                      item ID "N_ToolMag_..." }
        Attribute:   MMS Deletable
        Attribute: Type Description = structure {
            components {
              {Component Name = "StoredObjects",
               Component Type = array {
                                numberOfElements,       -- number of positions
                                elementType = visible-string -32 } },
              {Component Name = "MagazineType",
               Component Type = visible-string -32 },
              {Component Name = "NumberOfPositions",
               Component Type = unsigned 16 }
            }
        }
        Attribute:    Access Method = locally defined
```

B.8.3.2.2 Tool Holder Object

The name of the Tool Holder object shall begin with the standardized prefix "N_ToolHolder_", complemented by a user or implementer defined tool holder identifier.

```
        OBJECT: Named Variable
```

```
Key Attribute:    Variable Name = domain-specific {
                                      domain ID "N_TLD_...",
                                             or "N_MOP_...",
                                      item ID "N_ToolHolder_..." }
    Attribute: MMS Deletable
    Attribute: Type Description = structure {
        components {
          {Component Name = "HolderType",
           Component Type = visible-string -32 },
          {Component Name = "Location",
           Component Type = visible-string -32 },
          {Component Name = "Position",
           Component Type = visible-string -32 },
          {Component Name = "HolderTools",
           Component Type = array {
                numberOfElements,         -- number of positions
                elementType = visible-string -32 } },
          {Component Name = "HolderState",
           Component Type = unsigned 8    -- value from a list
                                             of user defined states
                                  },
          {Component Name = "HolderDimensions",
           Component Type = array {
                numberOfElements,         -- number of holder
                                             dimensions
                elementType = floating-point {
                       format-width 32,
                       exponent-width 8 } },
          {Component Name = "ProcessInformation",
           Component Type       -- user defined },
          {Component Name = "UserInformation",
           Component Type       -- user defined }
        }
      }
    Attribute:    Access Method = locally defined
```

B.8.3.2.3 Tool Object

The name of the Tool object shall begin with the standardized prefix "N_Tool_", complemented by a user or implementer defined tool identifier.

```
    OBJECT: Named Variable

Key Attribute:     Variable Name = domain-specific {
                                      domain ID "N_TLD_...",
                                             or "N_MOP_...",
                                      item ID "N_Tool_..." }
    Attribute:  MMS Deletable
    Attribute:  Type Description = structure {
        components {
          {Component Name = "ToolType",
           Component Type = visible-string },
          {Component Name = "Location",
           Component Type = visible-string -32 },
          {Component Name = "Position",
           Component Type = visible-string },
          {Component Name = "ToolEdges",
           Component Type = array {
                numberOfElements,         -- number of tool edges
                elementType = visible-string -32 } },
```

```
            {Component Name = "ToolState",
             Component Type = unsigned 8    -- 0 In Use
                                            -- 1 Worn
                                            -- 2 Broken
                                            -- 3 En Route
                                            -- 4 Not Released
                                            -- 5 Released
                                            -- 6 Needs Measurement
                                            -- 7 Needs Measurement After
                                            -- :     First Operation
                                            -- n User defined state(s)
                                               },
            {Component Name = "ToolDimensions",
             Component Type = array {
                   numberOfElements,        -- number of tool dimensions
                   elementType = floating-point {
                      format-width 32,
                      exponent-width 8 } } },
            {Component Name = "ProcessInformation",
             Component Type     -- user defined },
            {Component Name = "HandlingInformation",
             Component Type     -- user defined }
          }
        }
      Attribute: Access Method      = locally defined
```

B.8.3.2.4 Tool Edge Object

The name of the Tool Edge object shall begin with the standardized prefix "N_ToolEdge_", complemented by a user or implementer defined tool edge identifier.

```
      OBJECT: Named Variable

  Key Attribute:      Variable Name = domain-specific {
                                        domain ID "N_TLD_...",
                                               or "N_MOP_...",
                                        item ID "N_ToolEdge_..." }
      Attribute:     MMS Deletable
      Attribute: Type Description   = structure {
          components {
            {Component Name = "ToolID",
             Component Type = visible-string -32 },
            {Component Name = "EdgeOffsets",
             Component Type = array {
                   numberOfElements,        -- number of edge offsets
                   elementType = floating-point {
                      format-width 32,
                      exponent-width 8 } } },
            {Component Name = "NominalEdgeLife",
             Component Type = unsigned 32 },
            {Component Name = "RemainingEdgeLife",
             Component Type = unsigned 32 },
            {Component Name = "UserInformation",
             Component Type     -- user defined }
          }
        }
      Attribute: Access Method      = locally defined
```

B.9 Conformance

B.9.4 PICS for standardized objects

Domain specific Named Variable Objects
N_ToolMag_...
N_ToolHolder_...
N_Tool_...
N_ToolEdge_...

Annex C: Extension Of ISO 9506-4 To Include Turning Machine Functionality (Normative)

C Introduction

This normative annex extends the general applicability of ISO 9506-4. The structure of this annex is the same as that of the main part of this standard, such that the clauses are in alignment. However, not every clause in the main part of ISO 9506-4 may have a counterpart in this annex, because its scope does not require it.

C.1 Scope

This annex provides the additional objects and standardized names which may be required when this standard is used in NC turning machine applications, specifically in stand-alone usage (as opposed to integrated into a flexible manufacturing system).

C.2 References

This annex does not use any references besides those which are listed in clause 2 of ISO 9506-4.

C.3 Definitions

This annex does not use any terms besides those which are defined in clause 3 of ISO 9506-4.

C.4 Symbols and Abbreviations

This annex does not use any symbols and abbreviations besides those which are listed in clause 4 of ISO 9506-4.

C.5 Application Description

In terms of this annex, NC turning machines may consist of the machine axes, tools, spindles, tool magazines, and user programs which control these entities. This is illustrated in figure 25.

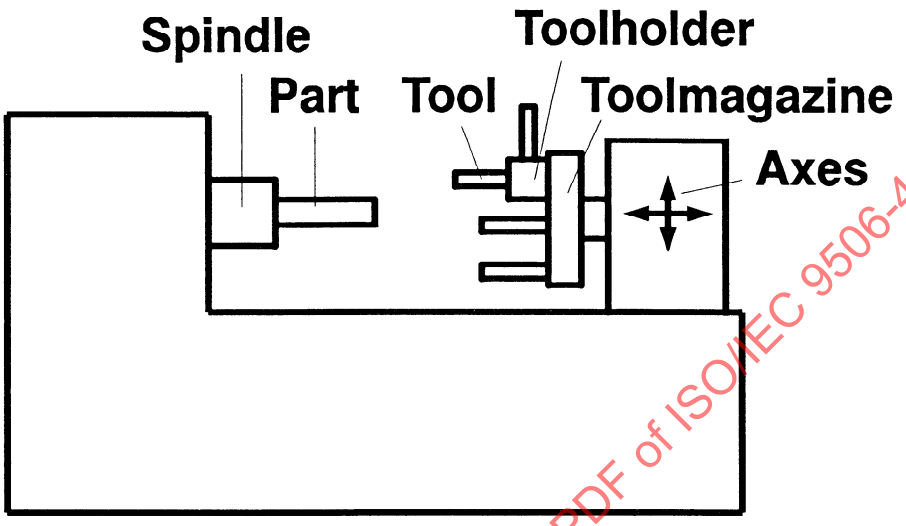

Figure 25: Turning Machine

The objects Tool Magazine, Tool Holder, Tools and Tool Edges for the turning machine are in concept identical to those used on milling machines. Therefore, the definitions of these objects in Annex B also apply to this Annex C.

C.5.1 Part Setup

In terms of the NC turning machine, part setup refers the combination of chuck jaws, part fixturing, and the mechanism used to position the part in the machine's work area. Portions of the setup may be re-configured for different parts or groups of parts, and may be changed manually or automatically.

C.5.2 Taper Trims

The taper trims are offsets which are provided to be applied when long flexible parts are being machined to prevent taper cuts induced when the unsupported section of a part is deflected by the tool.

C.6 NC Turning Specific Context Mapping

C.6.4.1 Objects Which Map To Named Variables

C.6.4.1.1 Part Setup Object

The purpose of the setup object is to define the configuration of chuck jaws, fixture, and part which may occupy the work area of the machine.

Key Attribute: Setup ID

Attribute: Chuck ID
Attribute: Chuck Size
Attribute: Setup Objects
Attribute: Setup State
Attribute: Setup Offsets

- Setup ID -- serial number or alphanumeric string which identifies the setup.
- Chuck ID -- serial number or alphanumeric string which identifies the chuck used with this setup.

- Chuck Size -- provides the maximum dimensions of the chuck.

- Setup Objects -- provides a list of fixture or part objects which are or may be used in this setup.

- Setup State -- one of the following:

0. Empty - There are no parts in this setup
1. Ready - This setup is ready to run
2. Finished - The parts associated with this setup are finished
3. Manual - This setup requires manual intervention
4. Fault - A fault has been detected with this setup

- Setup Offsets -- a measurement of the distance from the origin of the machine work area to the setup origin.

The Part Setup object is properly mapped into an MMS Named Variable object. Name and structure of this variable is described in clause C.8.3.1.1.

C.6.4.1.2 Taper Trims Object

The pupose of the taper trims object is to provide the offsets required to compensate for deflection of cantilevered workpieces.

Key Attribute: Taper Trim ID

Attribute: X Axis Trim
Attribute: Z Axis Trim

- Taper Trim ID -- serial number or alphanumeric string which identifies this variable.

- X Axis Trim -- provides the offset dimension in X.

- Z Axis Trim -- provides the offset dimension in Z.

The Taper Trims object is properly mapped into an MMS Named Variable object. Name and structure of this variable is described in clause C.8.3.1.2.

C.8.3 Named Variable Objects

C.8.3.2 Domain specific Standardized Named Variable Objects

C.8.3.2.1 Setup Object

The name of any "setup" named variable object shall begin with the prefix "N_Setup_", complemented by a user or implementer defined identifier.

```
    OBJECT: Named Variable

Key Attribute:     Variable Name = domain-specific {
                                   domain ID "N_SET_...",
                                   item ID "N_Setup_..." }
    Attribute:  MMS Deletable
    Attribute:  Type Description = structure {
        components {
          {Component Name = "ChuckID",
           Component Type = visible-string },
          {Component Name = "ChuckSize",
           Component Type = array {
                 numberOfElements,
                 elementType = unsigned 16 } },
          {Component Name = "SetupObjects",
           Component Type = array {
                 numberOfElements,
                 elementType = visible-string } },
          {Component Name = "SetupState",
           Component Type = unsigned 8    -- 0 Empty
                                          -- 1 Ready
                                          -- 2 Finished
                                          -- 3 Manual
                                          -- 4 Fault
                                             },
          {Component Name = "SetupOffsets",
           Component Type = structure {
                          components {
                                {Component Name = "XComponent",
                                 Component Type = floating-point {
                                                  format-width 32,
                                                  exponent-width 8 },
                                {Component Name = "ZComponent",
                                 Component Type = floating-point {
                               format-width 32,
                                                  exponent-width 8 } }
          } } } }
    Attribute:  Access Method     = locally defined
```

C.8.3.2.2 Taper Trims Object

The name of any "taper trim" named variable object shall begin with the prefix "N_TaperTrims_", complemented by a user or implementer defined identifier.

```
        OBJECT: Named Variable

    Key Attribute:     Variable Name = domain-specific {
                                       domain ID " N_SET_...",
                                       item ID " N_TaperTrims_..." }
        Attribute:  MMS Deletable
        Attribute:  Type Description = structure {
            components {
              {Component Name = "XTrim",
               Component Type = floating-point {
                                    format-width 32,
                                    exponent-width 8 } },
              {Component Name = "ZTrim",
               Component Type = floating-point {
                                    format-width 32,
                                    exponent-width 8 } }
              } }
        Attribute:  Access Method    = locally defined
```

C.9 Conformance

C.9.4 PICS for standardized objects

Domain specific Named Variable Objects
N_ToolMag_...
N_ToolHolder_...
N_Tool_...
N_ToolEdge_...
N_Setup_...
N_TaperTrims_...

Annex D: Extension Of ISO 9506-4 To Include Flexible System Functionality (Normative)

D Introduction

This normative annex extends the general applicability of ISO 9506-4. The structure of this annex is the same as that of the main part of this standard, such that the clauses are in alignment. However, not every clause in the main part of ISO 9506-4 may have a counterpart in this annex, because its scope does not require it.

D.1 Scope

This annex provides the additional objects and standardized names which may be required when this standard is used in flexible systems applications, specifically when milling or turning machines, or both are integrated into a flexible manufacturing system.

D.2 References

This annex does not use any references besides those which are listed in clause 2 of ISO 9506-4.

D.3 Definitions

This annex does not use any terms besides those which are defined in clause 3 of ISO 9506-4.

D.4 Symbols and Abbreviations

This annex does not use any symbols and abbreviations besides those which are listed in clause 4 of ISO 9506-4.

D.5 Application Description

Milling or turning machines may be integrated into flexible manufacturing systems. The most important characteristic of such a system is automated transportation of parts, tools, and other means of production, to and from machines with automatic loading and unloading devices. Pallets and fixtures may be used for this transportation of parts and tools. These objects take part in the machining process controlled by an NC controller. All these objects may be stored and manipulated on a machine, they may be located in a pallet storage, a transfer station, the working area of a machine, or similar locations. Figure 26 shows a milling machine integrated into a flexible manufacturing system.

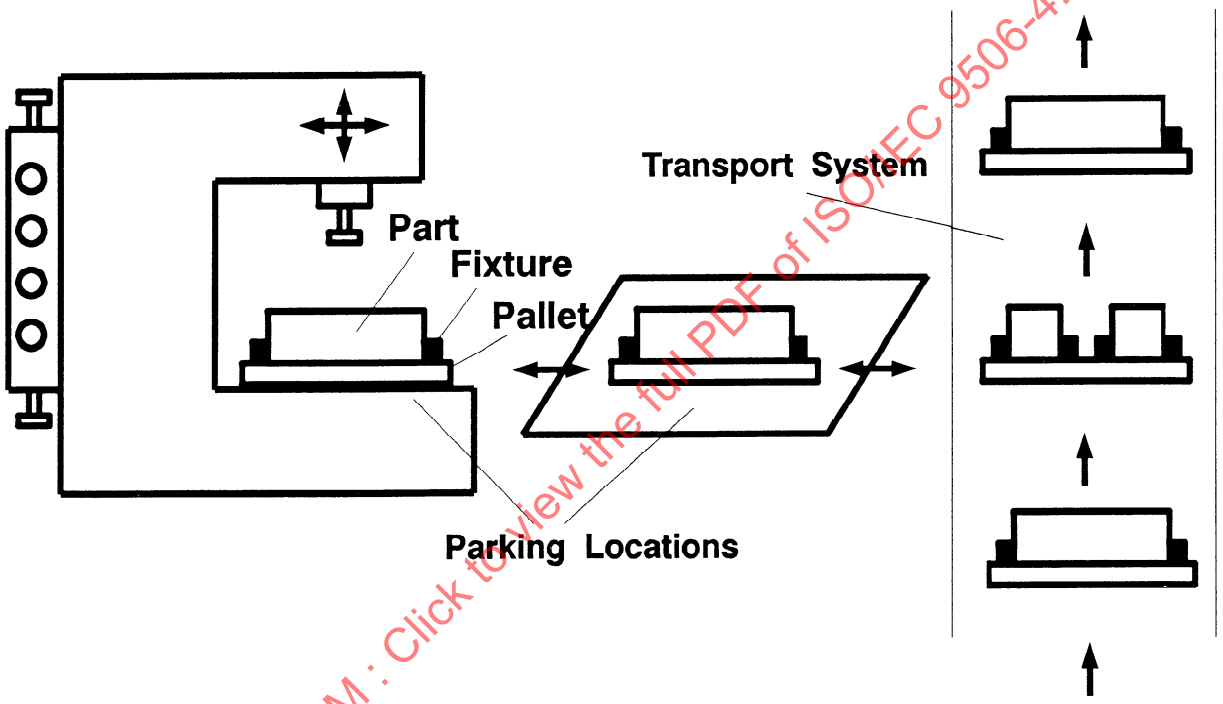

Figure 26: Milling Machine In A Flexible Manufacturing System

During the manufacturing process a robot, an auxiliary device, or an operator may exchange new tools for used ones in the tool magazine. An automated transport system, or an operator carries new parts to the machine and removes finished parts from the machine. The data for the exchanged tools, the parts delivered to the machine, the offsets for the pallets, fixtures, parts and other information needs to be transferred between the host system or a file store and the NC system. Therefore, it is necessary that these components which become part of the NC system during the manufacturing process, shall be mapped to MMS objects.

D.5.1 Parking locations, partitions

During an automated manufacturing process the means of production, such as pallets, parts, or similar, required objects usually are located at specific places within the work space of the system. This standard

defines these places as parking locations. A parking location may be subdivided into partitions, as illustrated in figure 27. A pallet or other means of production may be located at a parking location or within a partition of a parking location, and for the automated manufacturing process it is necessary to define such location accurately.

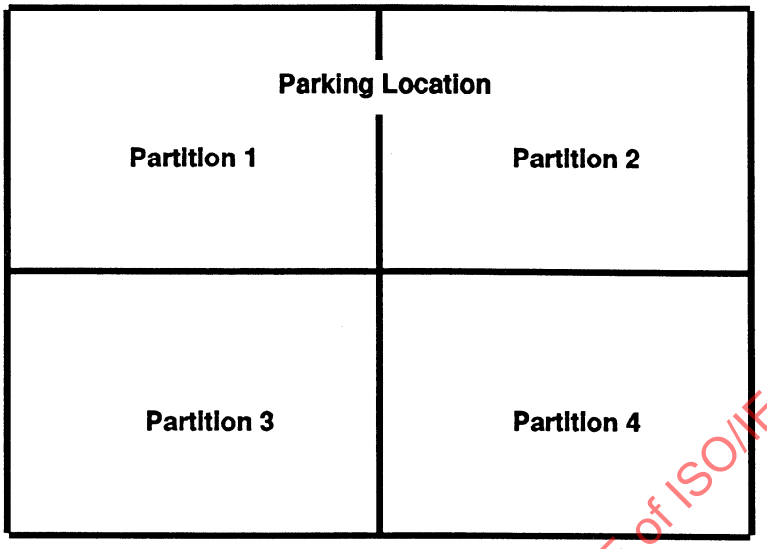

Figure 27: Example for a Parking Location with 4 Partitions

D.5.2 Pallets, partitions, fixtures, parts

For the transport of the means of production pallets are normally used. A Pallet may also be subdivided into partitions. Figure 28 illustrates a pallet subdivided into 4 Partitions. Figure 29 illustrates a pallet located within a partition of a parking location.

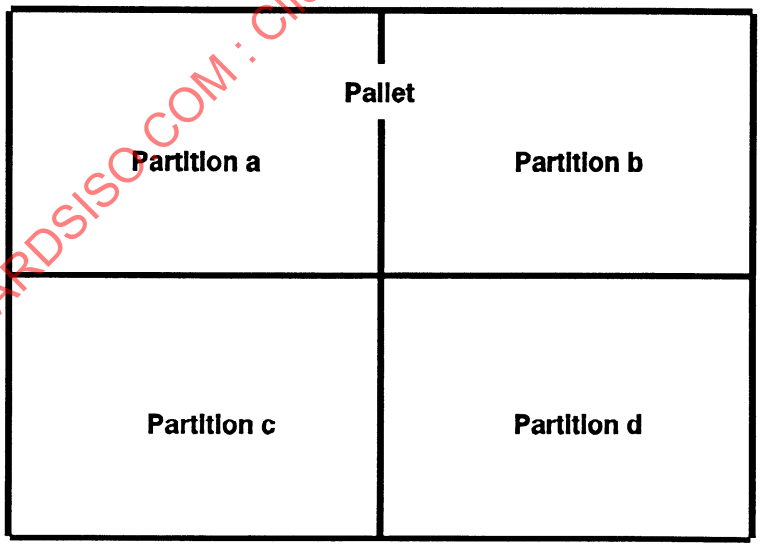

Figure 28: Example of a Pallet with 4 Partitions

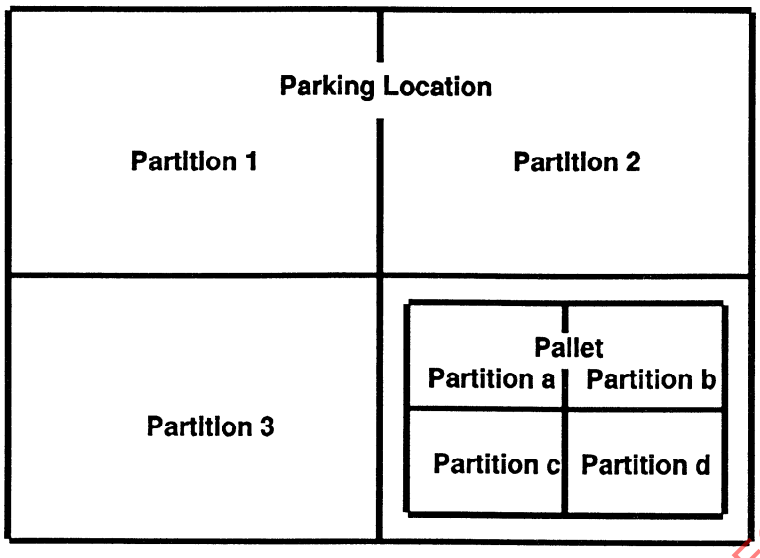

Figure 29: Example of a Parking Location with a Pallet

In addition to parts, tools or other means of production, pallets may also contain fixtures for the parts.

For the automated manufacturing process it is necessary to determine the position of the means of production precisely, such that for example, the position of a part mounted on a pallet located within the working envelope of the machine may be determined. It is necessary that the relationship of parking location, partition, pallet, fixture and part be available to the NC controller, and this shall be considered when mapping these objects to MMS objects. An example of such a relationship is illustrated in figure 30. There may be many other such combinations of object relationships.

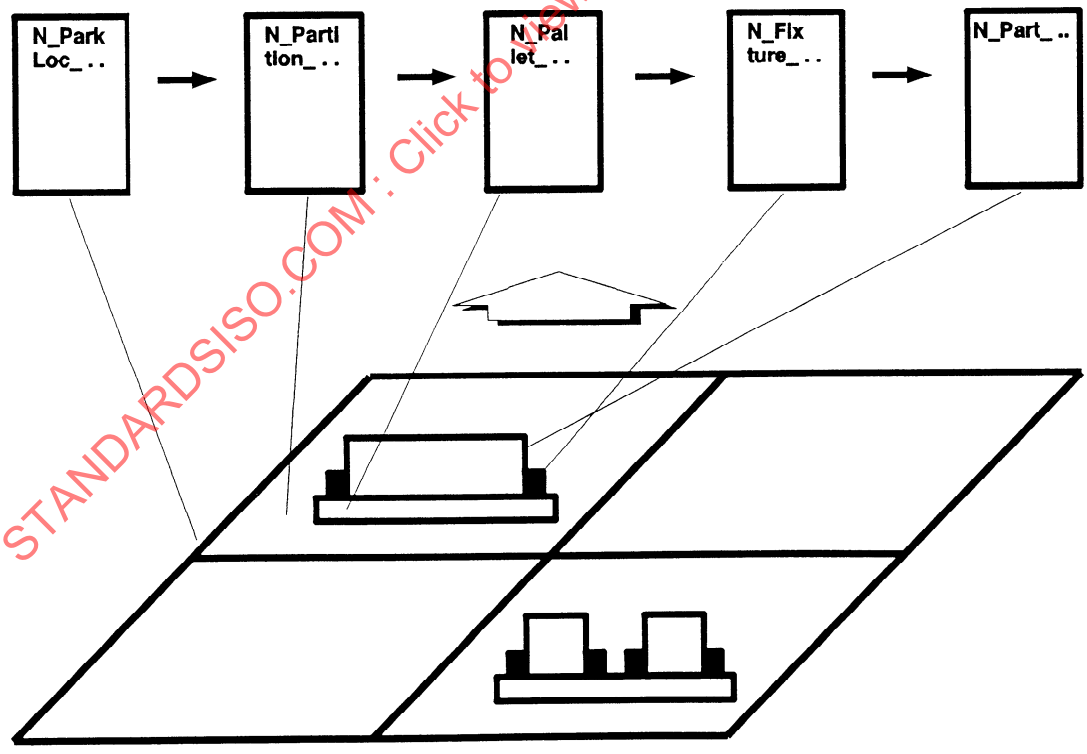

Figure 30: Example of Parking Location, Partition, Pallet, Fixture, and Part Relationship.

D.6 Flexible Systems Specific Context Mapping

D.6.2 Objects Which Map To Domains

D.6.2.1 Means of Production Data

The Means of Production Data Domain Object contains data which describe the means of production, such as parking locations, pallets and all the objects carried by such pallets, which may include fixtures, parts, and similar objects. This domain provides a naming space for Domain-specific Named Variables. An example of the Means of Production Data Domain is illustrated in figure 31.

For general definitions of these domains refer to chapter 6.2.1 in the main part of this standard.

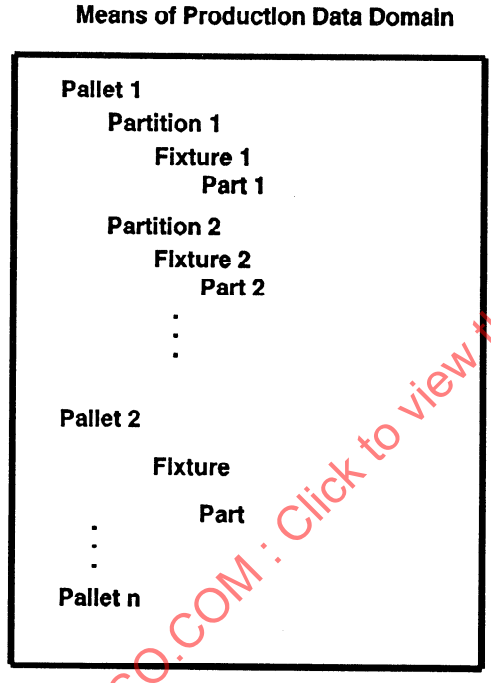

Figure 31: Means of Production Data Object

D.6.4.1 Objects Which Map To Named Variables

D.6.4.1.1 Parking Location Object

The Parking Location object is used to define a pallet stand, machine table, storage rack, or similar holding place for pallets, fixtures, parts, tools, and similar objects.

Key attribute: Location ID