ALLEN-BRADLEY

PLC–5 MAP/OSI Software
(Cat. No. 1785–OSI)

User Manual

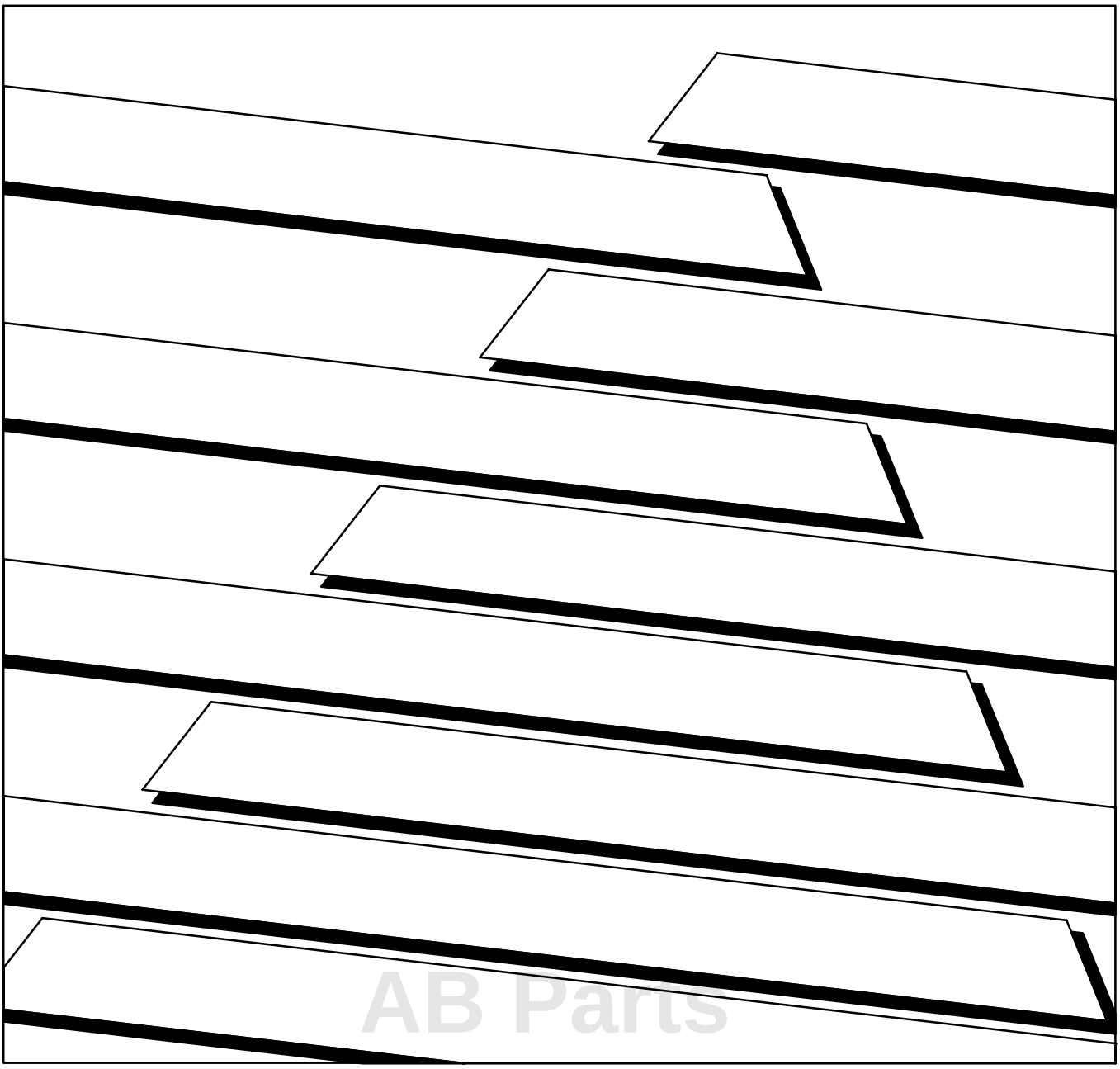


Table of Contents

Preface

Manual Contents

This manual provides information on the Allen–Bradley PLC–5™ MAP/OSI Software (cat. no. 1785–OSI) and the PLC–5 MAP/OSI Coprocessor (1785–O5G/B). It contains information on the following topics:

For this type of information:	See:
overview of Open System Interconnect (OSI), Manufacturing Automation Protocol (MAP), and Manufacturing Message Specification (MMS)	Chapter 1
introduction to the PLC–5 MAP/OSI Coprocessor	Chapter 2
the MMS services supported by your PLC–5 MAP/OSI Software and Coprocessor, examples of mapping MMS data types onto PLC–5 controller data files, and how MMS relates to your OSI coprocessor	Chapter 3
programming your PLC–5 MAP/OSI Coprocessor, including rules for using MMS named variables and address strings	Chapter 4
advanced programming techniques	Chapter 5
reference information on mapping MMS data types onto PLC–5 controller data files, including default data types	Appendix A
a list of the PLC–5 MAP/OSI Software's error codes	Appendix B
the PLC–5 MAP/OSI Coprocessor's PICS (Protocol Implementation Conformance Statement)	Appendix C
a list of the OSI communication layers' parameters, statuses, counters, and actions related to the PLC–5 MAP/OSI Software	Appendix D
a list of the OSI communications layer management options	Appendix E

Important: This manual **does not** cover installation procedures for the PLC–5 MAP/OSI software. You install the software via the Allen–Bradley MAP Station Manager, refer to publication 6630–6.5.2 for installation instructions.

We refer to the PLC–5 MAP/OSI Software as the OSI software, and the PLC–5 MAP/OSI Coprocessor as the OSI coprocessor throughout this manual.

Who Should Read This Manual

You should read this manual before installing and using your Allen–Bradley PLC–5 MAP/OSI Software. We assume you have experience using and programming Allen–Bradley PLC® Controllers, and some knowledge of the MAP 3.0 Specification.

What You Should Receive

With your order of the OSI software, along with this manual, you should have received the OSI software on:

- one 3 1/2” diskette and
- one 5 1/4” diskette

If you did not receive these items, contact your local Allen–Bradley integrator or sales office.

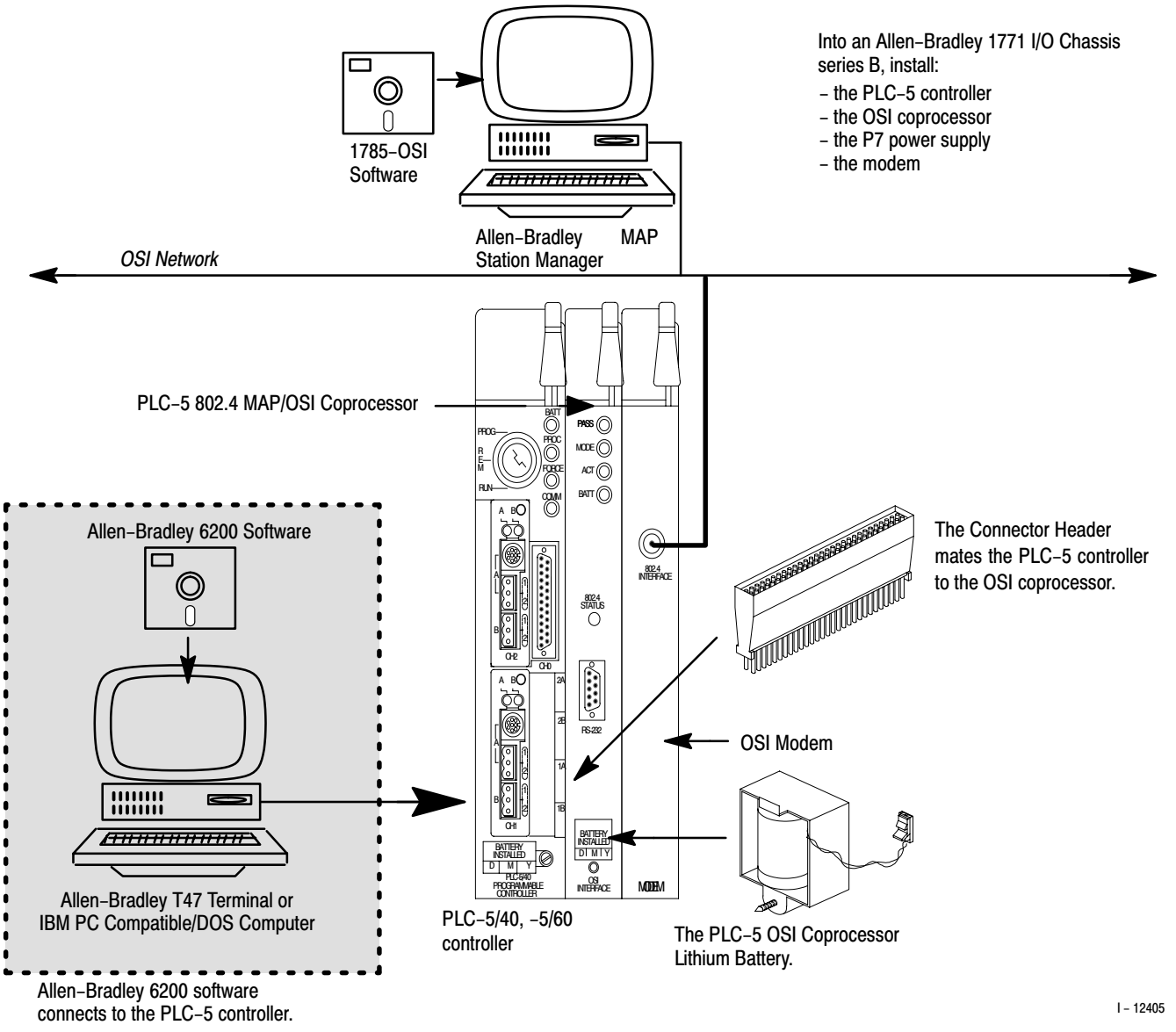
The Equipment You Will Need

Your PLC–5 MAP/OSI Software is part of the Allen–Bradley PLC–5 MAP/OSI interface package. You must also have the following Allen–Bradley equipment installed:

Product:	Catalog Number:
PLC–5 Controllers	1785–L40L/B, –L40B/B, –L60L/B, –L60B/B, –L30B/A, –L20B/A, –L11B/A. Contact your local Allen–Bradley distributor or sales office for a current list of compatible controllers.
PLC–5 Programming Software	6200–PLC5, version 4.3 or higher
PLC–5 802.4 MAP/OSI Broadband Modem or PLC–5 802.4 MAP/OSI Carrierband Modem	1785–O5A, B, C (broadband) or 1785–O5CB (carrierband)
PLC–5 Connector Header	1785–CNH
Power Supply	1771–P7
1771 I/O Chassis (series B)	Contact you local Allen–Bradley distributor of sales office for the correct I/O chassis catalog number for your application.
PLC–5 OSI Coprocessor Lithium Battery	1785–U1

You must have each of these products installed and running before using the 1785–OSI software (see figure P.1). For installation and usage instructions, refer to the installation instructions shipped with each of these products.

Figure P.1
You Must Have This Equipment Installed Before Using the OSI Software



If you need more information on these products, contact your local Allen–Bradley integrator or sales office for assistance.

Related Publications

Refer to the following table for information on where to read more about Allen–Bradley MAP/OSI products:

Product:	Refer to this publication for installation/usage instructions:
PLC–5 802.4 MAP/OSI Coprocessor	1785–2.23
PLC–5 802.4 MAP/OSI Broadband Modem	1785–2.12
PLC–5 802.4 MAP/OSI Carrierband Modem	1785–2.20
PLC–5 Connector Header	1785–2.23 and 1785–2.14
PLC–5 Lithium Battery	1785–2.23 and 1785–2.24
PLC–3® MAP Interface	6632–6.5.2
Pyramid Integrator™ OSI Interface Module	5820–6.5.1 and 6632–2.11
MAP Station Manager	6630–6.5.2 and 6632–2.11

Chapter 1

Overview of the MAP Communication Environment

Chapter Objectives

This chapter introduces you to MAP communication and shows how it relates to your OSI coprocessor. This chapter covers:

- Open Systems Interconnect (OSI)
- Manufacturing Automation Protocol (MAP)
- Manufacturing Message Specification (MMS)
- the MMS Modeling Concept
- MAP communications and your OSI coprocessor

The following sections contain general overviews of these subjects and are not meant to provide in–depth information. If you are already familiar with these topics, you may want to proceed to Chapter 2.

Open Systems Interconnect (OSI)

The Open Systems Interconnect (OSI) is a standard that provides the framework for defining the process of communication between nodes (i.e., computers, terminals, PLC controllers). There are many activities that need to be accomplished when two nodes communicate with each other over the network. The OSI standard defines these activities in its **seven–layer reference model** (figure 1.1).

Figure 1.1
The OSI Seven–layer Reference Model

7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

I – 12406

The seven layers define the:

- activities involved in communicating on the network
- services required to perform those activities

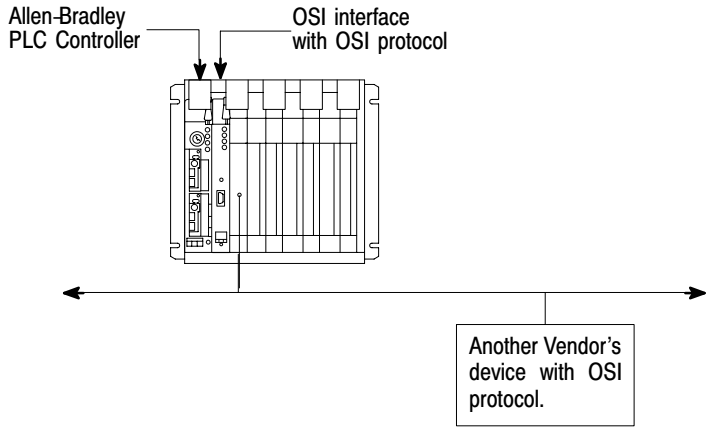
The individual layer specifications dictate *how* the functions are accomplished. The tasks within the layers are carried out by **protocols.**

Protocols are rules for how information is coded and passed between two nodes. The protocols are actually the part that is implemented, the OSI model serves merely as a reference to discuss the different aspects of communication between devices. The following table lists the functions that the protocols at each layer must accomplish.

This layer:	Contains the functions that:
7 Application	manipulate information to support applications. This layer's protocols contain the most functionality.
6 Presentation	ensure information is delivered in a form the receiving system can understand and use.
5 Session	manage communications between two application processes.
4 Transport	transfer reliable data between communicating nodes.
3 Network	route communication between the communicating nodes.
2 Data Link	perform synchronization and error control for information passed over the physical link (manages the access to the medium).
1 Physical	activate, maintain, and deactivate the physical connection.

The OSI system enables many different vendors' devices on the same network to communicate with each other. In other words, as long as two different vendors construct protocols that ''fit" in the seven layer model the same way, those two devices will be able to communicate (figure 1.2).

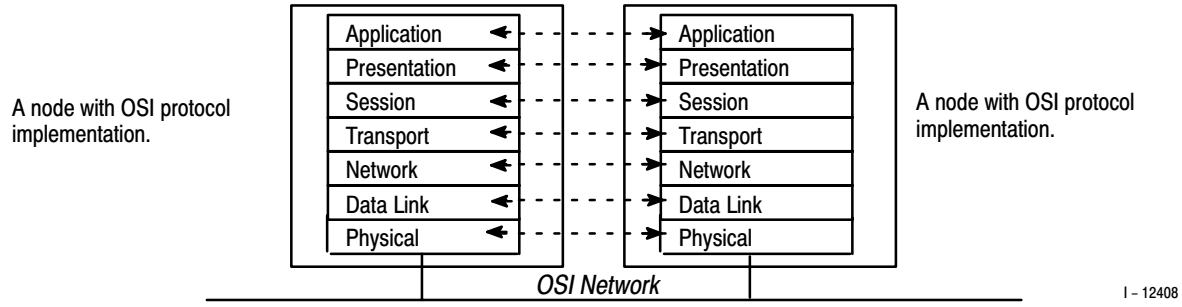
Figure 1.2
Different Vendors' Devices Can Communicate on OSI Networks



I – 12407

Each node on an OSI network is equipped with a layer mechanism that incorporates the ‘‘rules” of the OSI standard. Each layer is able to ‘‘talk” with only its counterpart within the node sending/receiving the data (figure 1.3).

Figure 1.3
Each Layer Communicates With its Counterpart Within Another Node



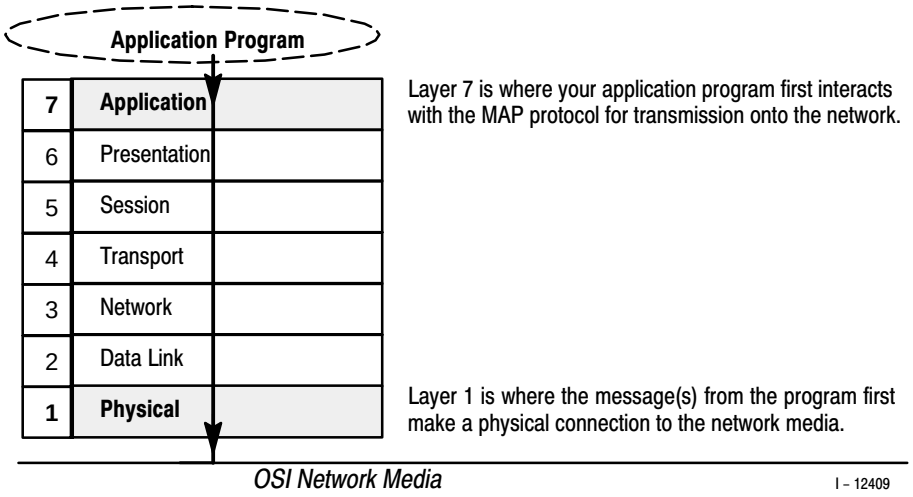
Your OSI coprocessor implements Manufacturing Automation Protocol (MAP). MAP is one set of OSI protocols.

Manufacturing Automation Protocol (MAP)

Manufacturing Automation Protocol (MAP) is a set of protocols based on the OSI seven–layer reference model described in the previous section. MAP specifies a set of protocols that must accomplish certain tasks within each of the model’s seven layers. Your OSI coprocessor implements the MAP 3.0 Protocol.

The seventh and first layers of the reference model are the two layers that are most distinct to your application. The seventh layer is the *Application* layer, with which your OSI coprocessor first interacts with your application program. The first layer is the *Physical* layer, with which your OSI coprocessor connects to the network media (figure 1.4). Your OSI coprocessor physically connects to MAP 802.4 network media.

Figure 1.4
The Layers 7 and 1 Are Most Distinct To Your Application

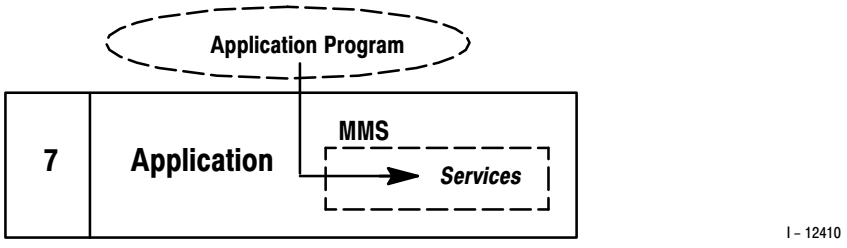


The Application Layer for the MAP 3.0 Specification contains a unique set of services called **Manufacturing Message Specification (MMS).**

Manufacturing Message Specification

Manufacturing Message Specification (MMS) provides services directly visible to (and used by) the user. MMS specifies a method for communicating with intelligent plant–floor devices. MMS is *not* an application program, it provides services to application programs. These programs then use the services to communicate with devices on the network (figure 1.5).

Figure 1.5
MMS is a MAP 3.0 Layer Seven Protocol that Provides Services



MMS gives MAP network devices a set of services that they can all access, allowing them to communicate freely.

The following sections provide a brief overview of MMS. Note that this is an *introduction* to the MMS communication environment. For *detailed* information, refer to the ISO/IEC 9506 Part 1 (MMS — Service

Definition). For a complete listing of the MMS services supported by your OSI coprocessor, refer to Chapter 3 of this manual.

The MMS Modeling Concept

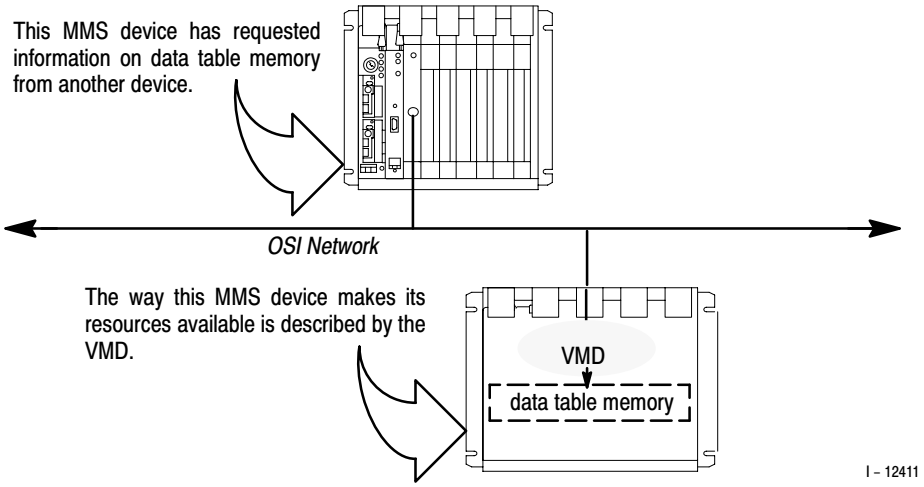
The dominant aspect of MMS is the concept of *modeling*. MMS defines models that describe the way in which resources are made available and the way in which these resources are accessed. At the center of the MMS modeling concept are:

- the Virtual Manufacturing Device (VMD)
- abstract object models

The VMD

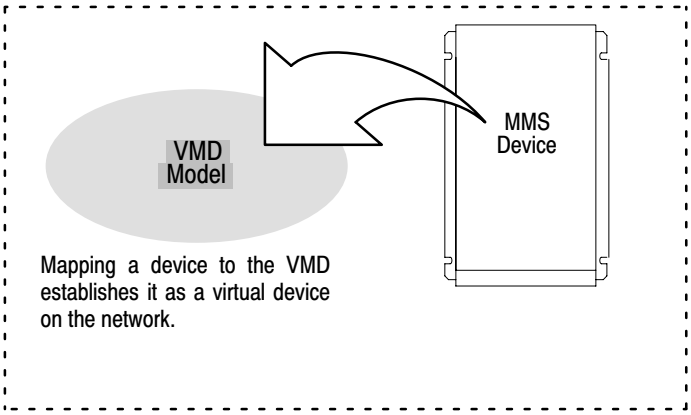
The VMD describes the externally visible behavior of an MMS device when it makes resources (i.e., data table memory, program files) available to other MMS devices. Assume, for example, that an MMS device exists and it makes system data table memory available to other MMS devices. It is the VMD that describes the way in which that MMS device makes the data table memory available (figure 1.6).

Figure 1.6
The VMD Describes the Way a Device Makes its Resources Available



It is each vendor's responsibility to associate the VMD model with their device (figure 1.7).

Figure 1.7
Vendors Must Map Their Devices to the VMD Model



The *mapping* concept.

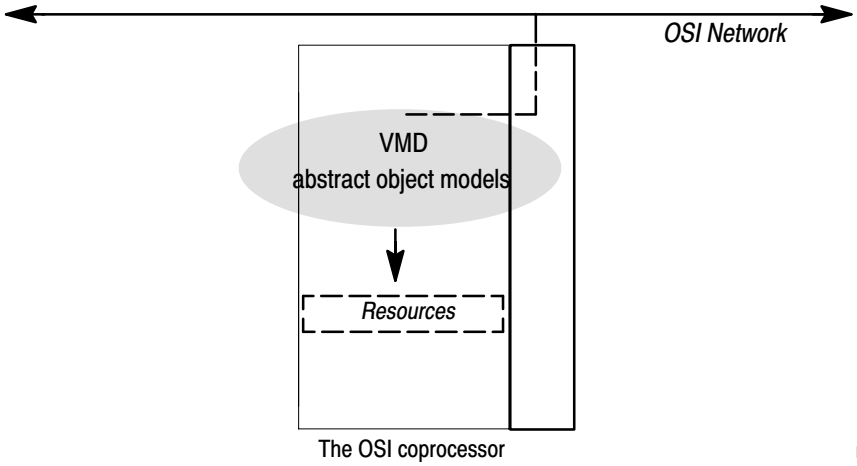
I – 12412

This *mapping* of a device to the VMD model establishes a device as a **virtual device** on the MAP network, allowing it to be accessed by other virtual devices through the use of MMS services. Note that the VMD model *theoretically* exists within an MMS device.

The MMS Abstract Object Models

MMS also defines a series of **abstract object models** that describe the externally visible behavior associated with a particular MMS service or group of services. The concept of object models is entirely abstract within the MMS specification, but represents real resources within a system. This results in a standard external view of all MMS devices that make system resources available, but allows each MMS device to implement the model in a manner that is appropriate for the system (figure 1.8).

Figure 1.8
Abstract Object Models are Part of the VMD Model



The models define abstract *objects* which are part of the VMD. An application program calls on the MMS services associated with a particular object to perform operations on that object. The objects implemented within your OSI coprocessor are listed in the table below:

The:	Are acted upon by MMS:	And allow an MMS client[1] to:
domain objects	Domain Management Services	upload or download a memory image within the system associated within an MMS server
program invocation objects	Program Invocation Management Services	control the operational state of the program(s) associated with an MMS server
variable objects	Variable Access Services	access data within the system associated with an MMS server.

[1]Refer to the section titled *Clients, Servers and the VMD*, later in this chapter, for an explanation of MMS clients and servers.

There are other object models within the MMS specification. The three listed above are the only objects currently implemented by your OSI coprocessor. The following sections briefly describe how the OSI coprocessor implements them.

Domain Objects Implemented by the OSI Coprocessor
Your OSI coprocessor implements a single domain object that represents the PLC–5 controller's entire memory image.

Program Invocation Objects Implemented by the OSI Coprocessor
Your OSI coprocessor implements a single program invocation object, which allows an MMS client to place the PLC–5 controller into different modes (i.e., program, test, and run mode). A program invocation object is

merely a grouping of domains within a system. Therefore, within the OSI coprocessor, the single program invocation object is made up of a single domain object.

Variable Objects Implemented by the OSI Coprocessor
Your OSI coprocessor implements the following types of variable objects:

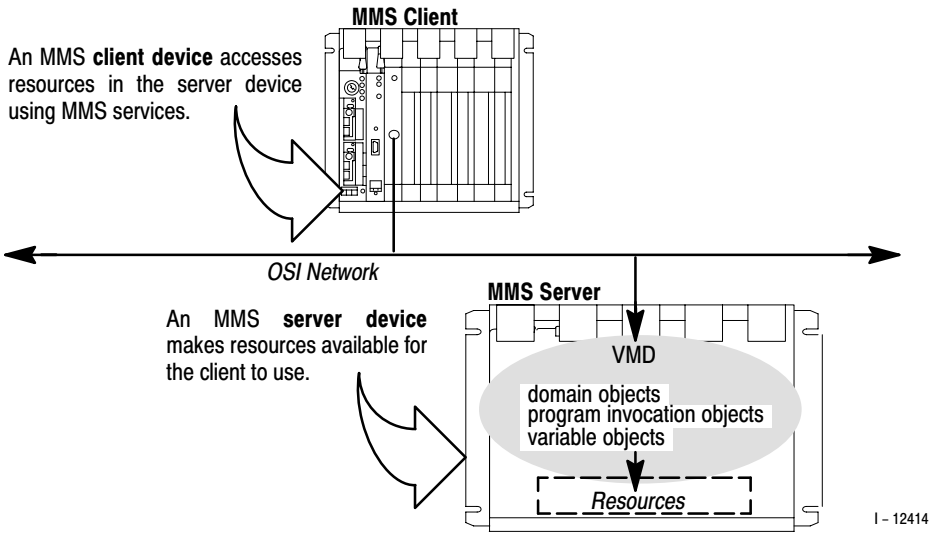
Variable Object:	Description:
Named Variables	MMS allows you to associate a name with an address within the node (i.e., 'paint_station' for address N12:1). The names are identification strings. To gain access to the address, simply specify the name.
Unnamed Variables (or address strings)	Variables that do not have an identifier string associated with an address. To access the location, you must specify the address.
Named Variable Lists	This is a grouping of named and/or unnamed variables referenced by a single identification string.
Named Types	These are objects that pre-define a data-type template for use when accessing unnamed variables or defining named variables. For example, if a client has defined a template for an array of 100 integers called 'int_array_100' within the OSI coprocessor, and the client sends a read request for "N7:0", specifying 'int_array_100' as the data type to be returned, the client will receive an array of 100 integers starting at "N7:0".

Clients, Servers, and MMS Modeling

In the previous sections we covered the various concepts of MMS modeling. It is important to note here that the MMS specification defines models for the behavior of a device acting as a **server**, but not as a **client.**

Within the MMS specification, the terms client and server are used to describe the tasks that a device performs when it carries out an MMS service. An MMS device that makes resources available for use by another MMS device is called a **server.** And the MMS device that makes use of those resources to perform some type of application function is called a **client** (figure 1.9).

Figure 1.9
An MMS Server Makes Resources Available to an MMS Client

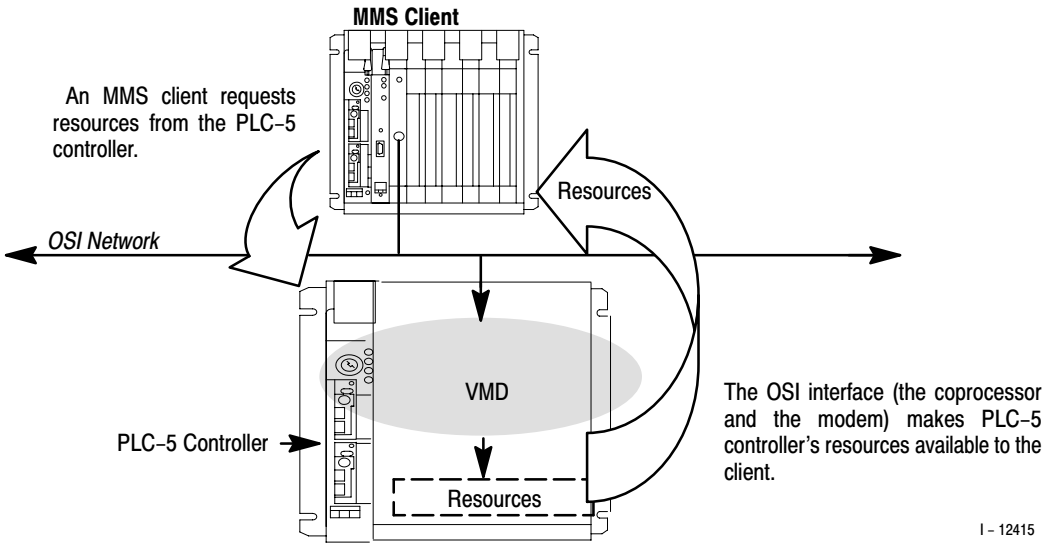


Clients, Servers, the VMD Model, and Your OSI Coprocessor

Your OSI coprocessor has the ability to act as both MMS server and client, depending on the particular MMS service. It is your PLC–5 controller that is modeled as a VMD.

The coprocessor and its OSI software make the PLC–5 controller's resources accessible to MMS client devices because the OSI software supports the MMS VMD model (figure 1.10).

Figure 1.10
An Allen–Bradley PLC Controller as an MMS Server



Your Coprocessor's OSI Communication Layers

Each of the seven OSI communication layers has its own set of attributes (characteristics) that in some way control or help to control the communication process. These attributes include:

- parameters
- statuses
- counters
- actions

You can manipulate the values of many of the attributes. Your OSI coprocessor is shipped with its attributes pre–set to Allen–Bradley default settings. Some of these settings **should not be changed.** Refer to Appendix D of this manual for a complete listing of the OSI coprocessor's attributes, including applicable **warnings** for those you should not change.

Implementing MMS

The formal description of the MMS options a vendor has implemented comes in the form of a **PICS** (Protocol Implementation Conformance Statement)**.** The PICS for your Allen–Bradley OSI coprocessor is listed in Appendix C of this manual.

Chapter 2

The PLC–5 802.4 MAP/OSI Coprocessor

Chapter Objectives

This chapter contains information on using the OSI coprocessor (cat. no. 1785–O5G). It covers the following topics:

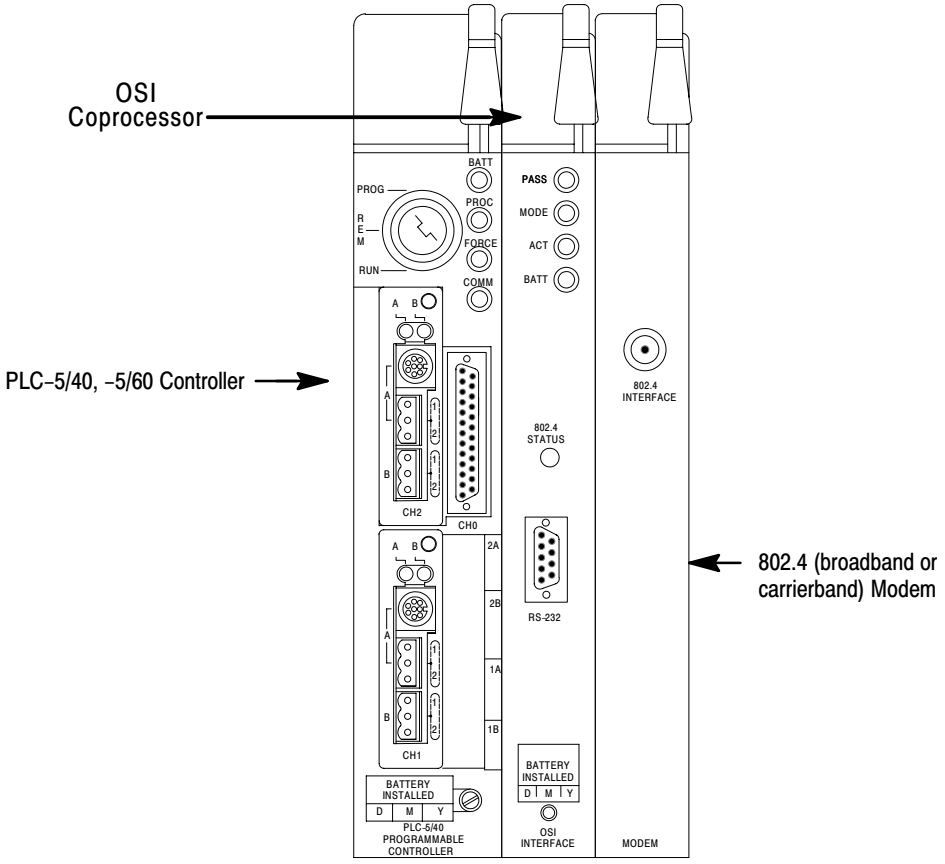
- introduction to the OSI coprocessor
- non–volatile memory and the lithium battery
- the LEDs
- the switches

This chapter **does not** cover installation instructions. For installation procedures, handling precautions, and additional information on the OSI coprocessor, refer to the PLC–5 802.4 MAP/OSI Coprocessor Installation Data (publication 1785–2.23 and 1785–2.24), shipped with the OSI coprocessor.

Introduction to the OSI Coprocessor

The OSI coprocessor connects a PLC–5 controller to a MAP 802.4 network (figure 2.1).

Figure 2.1
The OSI Coprocessor



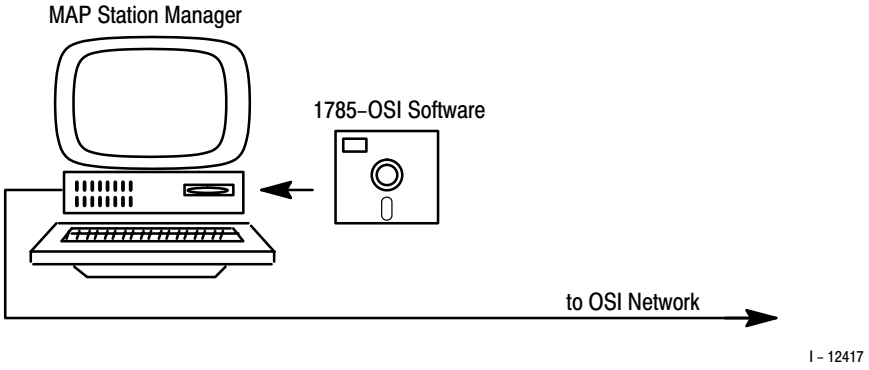
The coprocessor is only half of the controller's connection to the network, you also need a modem. The type of modem depends upon the type of network you are using:

For a:	You need an Allen–Bradley:	Catalog Number:
broadband network	PLC–5 802.4 MAP/OSI Broadband Modem	1785–O5A, B, or C
carrierband network	PLC–5 802.4 MAP/OSI Carrierband Modem	1785–O5CB

Installing the OSI Software

You install the OSI software (cat. no. 1785–OSI) via your Allen–Bradley MAP Station Manager (cat. no. 6630–PM, –PMC) (figure 2.2).

Figure 2.2
Install the OSI Software via the MAP Station Manager



Refer to the station manager's user manual (publication 6630–6.5.2) for instructions on installing the software and ‘‘downloading” (system loading) the software to the OSI coprocessor.

Non–volatile Memory and the Lithium Battery

Non–volatile memory is a portion of memory in the OSI coprocessor where the following information is stored:

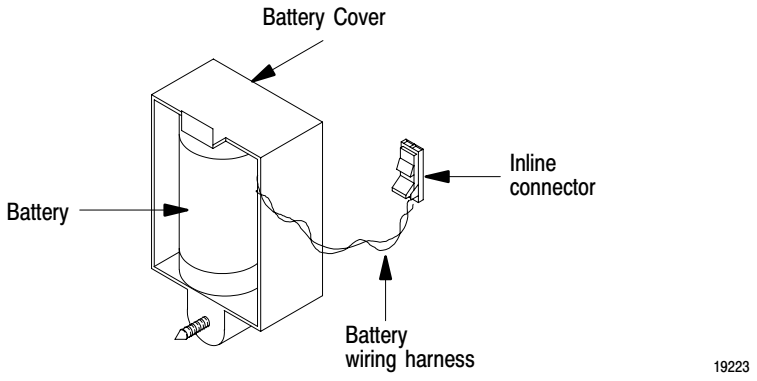
- the system load file (the OSI software)
- the Local Directory Information Base (LDIB) entries
- other configurable parameters (all MAC–, MMS–, and system–layer parameters, network–layer static routing table, transport– and session–layer buffer information, and RS–232 port configuration parameters)
- MMS objects you have saved

Through the Allen–Bradley MAP Station Manager, you download the OSI software, manage LDIB entries, configure parameters, and save MMS objects. The OSI software image, the LDIB entry information, and the configurable parameters are automatically stored in the OSI coprocessor's non–volatile memory. You have the option of also saving MMS objects through the station manager's *Retain MMS Objects* menu. Refer to the station manager's user manual (publication 6630–6.5.2) for more information.

The Lithium Battery

You received one 3.6 volt lithium battery (cat. no. 1785–U1) with your shipment of the OSI coprocessor (figure 2.3).

Figure 2.3
The PLC–5 OSI Coprocessor Lithium Battery



The battery provides the power necessary to retain the information in non–volatile memory during the time power is not supplied to the OSI coprocessor. Refer to publications 1785–2.23 and 1785–2.24 for installation instructions and precautionary information.

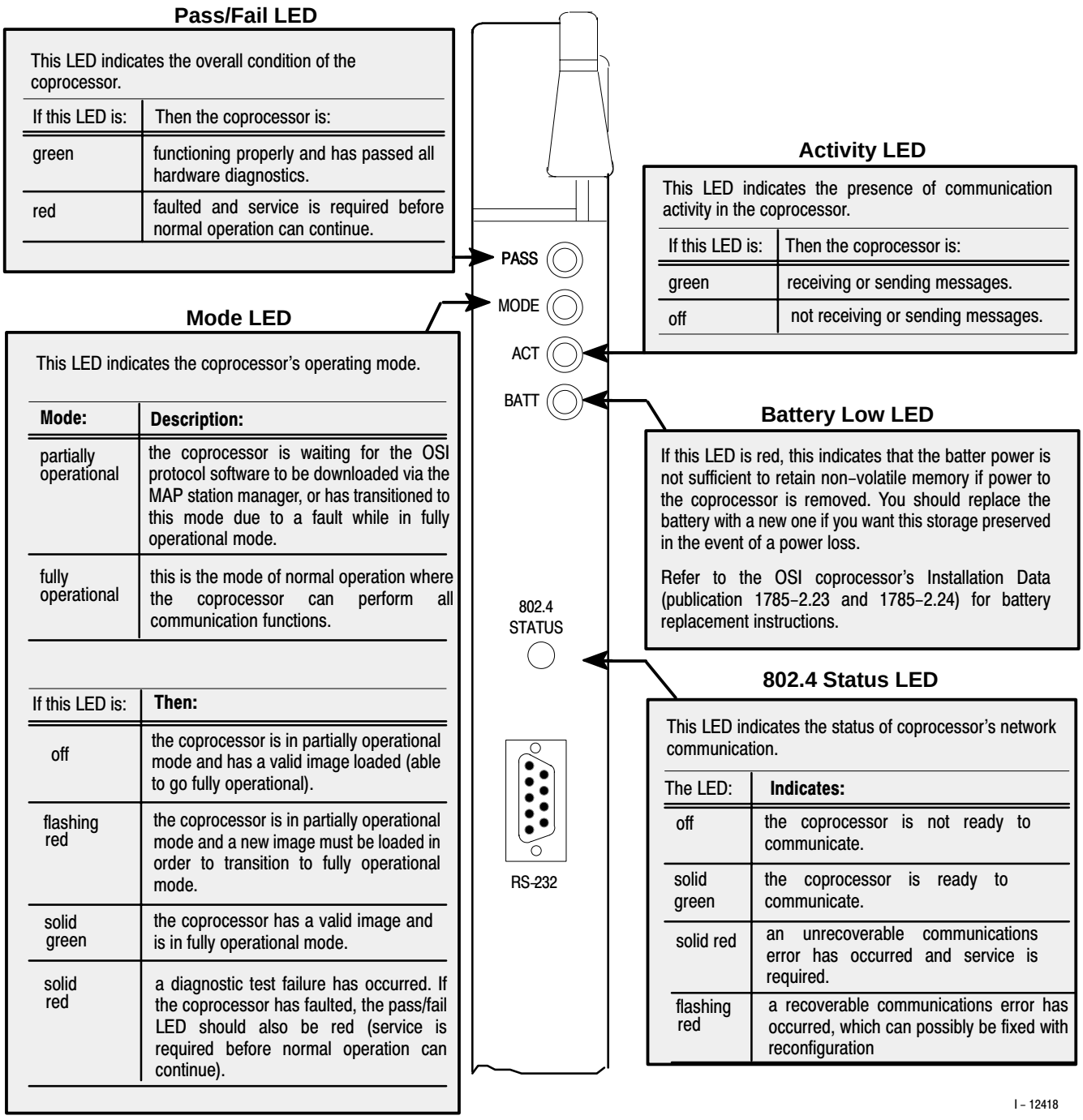
Battery Maintenance

The life of the lithium battery is related to the time it is in use (when the power to the OSI coprocessor is removed), so it is not possible to predict when you will need to replace it. Generally, we recommend that you replace the lithium battery every year or when the BATT status LED is red.

The LEDs

The OSI coprocessor has five LEDs on the front panel. At power–up, the LEDs cycle through each of their colors (either red or green) so you can visually inspect them. Each LED has a specific function. Figure 2.4 provides an overview of each LED.

Figure 2.4
The OSI Coprocessor's LEDs



Pass/Fail LED

This LED indicates the overall condition of the coprocessor.

If this LED is:	Then the coprocessor is:
green	functioning properly and has passed all hardware diagnostics.
red	faulted and service is required before normal operation can continue.

Activity LED

This LED indicates the presence of communication activity in the coprocessor.

If this LED is:	Then the coprocessor is:
green	receiving or sending messages.
off	not receiving or sending messages.

Mode LED

This LED indicates the coprocessor's operating mode.

Mode:	Description:
partially operational	the coprocessor is waiting for the OSI protocol software to be downloaded via the MAP station manager, or has transitioned to this mode due to a fault while in fully operational mode.
fully operational	this is the mode of normal operation where the coprocessor can perform all communication functions.

If this LED is:	Then:
off	the coprocessor is in partially operational mode and has a valid image loaded (able to go fully operational).
flashing red	the coprocessor is in partially operational mode and a new image must be loaded in order to transition to fully operational mode.
solid green	the coprocessor has a valid image and is in fully operational mode.
solid red	a diagnostic test failure has occurred. If the coprocessor has faulted, the pass/fail LED should also be red (service is required before normal operation can continue).

Battery Low LED

If this LED is red, this indicates that the batter power is not sufficient to retain non–volatile memory if power to the coprocessor is removed. You should replace the battery with a new one if you want this storage preserved in the event of a power loss.

Refer to the OSI coprocessor's Installation Data (publication 1785–2.23 and 1785–2.24) for battery replacement instructions.

802.4 Status LED

This LED indicates the status of coprocessor's network communication.

The LED:	Indicates:
off	the coprocessor is not ready to communicate.
solid green	the coprocessor is ready to communicate.
solid red	an unrecoverable communications error has occurred and service is required.
flashing red	a recoverable communications error has occurred, which can possibly be fixed with reconfiguration

The Switches

The OSI coprocessor has a four–position dip switch you can access without removing any covers. The switches are labelled 1, 2, 3, and 4 and are located at the back of the coprocessor. Above the switches you will see a small legend that illustrates the ON and OFF position of the switches (figure 2.5).

Important: Your OSI coprocessor is shipped with all of its switches in the OFF position. If you need to change the switch settings, you must do so **before** you install the OSI coprocessor into the chassis.

Figure 2.5
The OSI coprocessor's switches

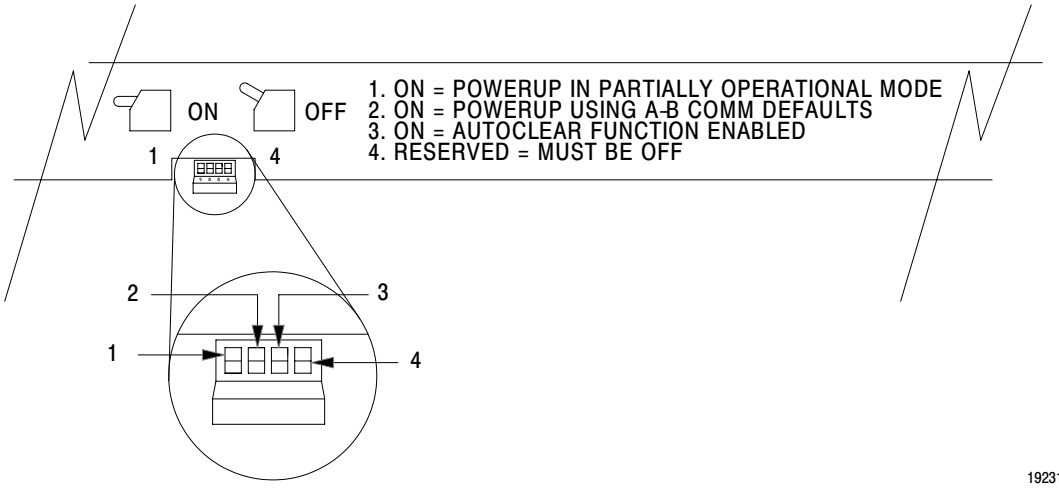


Figure 2.6 provides a larger view of the switches and a description of each one.

Figure 2.6
A Description of Each OSI Coprocessor Switch

(Switch 1)
Fully Operational/Partially Operational Switch

Allows you to select the default mode of operation the coprocessor enters on power–up or reset.

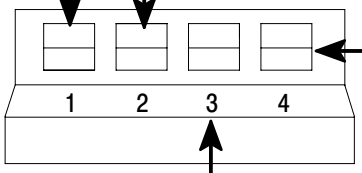
When this switch is:	The coprocessor powers up in:
ON	partially operational mode.
OFF	fully operational mode.

If you select:	Then the coprocessor:
fully operational mode and there is a valid image in non–volatile memory	enters fully operation mode.
fully operational mode and there is not a valid image in non–volatile memory	enters partially operational mode.
partially operational mode.	enters partially operational mode.

(Switch 2)
User/Allen–Bradley Communication Defaults Switch

Allows you to select the values the coprocessor uses for the MAC layer and RS–232 port parameters on power–up or reset.

When this switch is:	The coprocessor powers up using:	These settings are the values:
ON	Allen–Bradley communication default settings.	used for the MAC layer and RS–232 port configuration parameters whenever the coprocessor's non–volatile memory is reinitialized.
OFF	user communication default settings. If no user defaults are stored, Allen–Bradley communication defaults are used.	of the MAC layer and RS–232 port parameters when you edit them via the Allen–Bradley MAP Station Manager. The coprocessor uses these values in place of the Allen–Bradley communication defaults (unless this switch is set to ON).



(Switch 4)

Important: Keep this switch OFF. This should be turned on by authorized Allen–Bradley personnel only.

OFF

(Switch 3)
Auto Clear Switch

Allows you to specify whether or not the contents of non–volatile memory will be erased (cleared) when the coprocessor is detached from the PLC–5 controller, and reattached to a different controller.

When this switch is:	Then:
OFF	the auto clear function is disabled and you are free to interchange coprocessors and PLC–5 controllers at anytime WITHOUT losing the contents of non–volatile memory.
ON	the autoclear function is enabled. The contents of non–volatile memory will be erased if and only if you complete all of the following tasks: **1.** Unmate the coprocessor and PLC–5 controller. **2.** Reattach the coprocessor to a different PLC–5 controller. **3.** Install the new mated pair into a chassis and apply power. **4.** Place the coprocessor into fully operational mode.

Why Use the Auto Clear Switch?

Typically, a maintenance engineer can swap a non–functioning OSI coprocessor with another. A coprocessor, initially mated to a PLC–5 controller, could later be mated to another. Since the configuration information follows the OSI coprocessor, that information may be incorrect for the new PLC–5 controller location.

It is possible, for example, that some of the MMS named variables in the first application would match those in the second application in name, but not in content or meaning. This is a situation where you may decide to ENABLE the auto clear switch, allowing you to clear the contents of the coprocessor's non–volatile memory, even with the coprocessor mated to a new PLC–5 controller.

I – 12419

Chapter 3

MMS and Your OSI Coprocessor

Chapter Objectives

This chapter covers how your OSI coprocessor supports MMS services, it contains the following sections:

- The Supported MMS Services
- Mapping MMS Data Types onto PLC–5 Controller Data Files
- Additional Information on Using Data Types
- MMS Object Management
- MMS Security

We assume you are already familiar with MMS and programming PLC–5 controllers. Refer to Chapter 1 if you would like a brief overview of MAP and MMS.

The Supported MMS Services

The following tables list each MMS service that your OSI coprocessor supports, and a brief description of each service.

Unless otherwise stated, the OSI coprocessor performs all functions associated with the **server** side of an MMS service without user intervention. The OSI coprocessor allows use of the **client** side of an MMS service by the execution of instructions in a PLC–5 application program.

MMS Service:	Description:	Support:
Initiate	Enables communication between the OSI coprocessor and another MMS device by making a connection with the other device. All MMS communication (except for the Initiate request and response) must be sent across an open connection. Note that the PLC–5 controller message instruction can ask the OSI coprocessor to connect to itself, which enables the execution of message instructions that perform MMS services within the local PLC–5 system.	Client and server. To invoke the client side of the Initiate service, an OPEN command is programmed into the PLC–5 controller. The server side is invoked when the OSI coprocessor receives an Initiate request.
Conclude	Gracefully terminates a connection between the OSI coprocessor and another MMS device. Upon successful completion of the Conclude service, no further communication can take place between the OSI coprocessor and the other MMS device until another connection is opened. This is the preferred service to terminate a connection.	Client and server. To invoke the client side of the conclude service, a CLOSE command is programmed into the PLC–5 controller. The server side is invoked when the OSI coprocessor receives a conclude request.
Abort	Abruptly terminates a connection between the OSI coprocessor and another MMS device. Upon completion of the abort service, no further communication can take place between the OSI coprocessor and the other MMS device until another connection is opened. This MMS service should only be used in extreme conditions.	Client and server. To invoke the client side of the Abort service, an ABORT command is programmed into the PLC–5 controller. The server side is invoked when the OSI coprocessor receives an abort request.

MMS Service:	Description:	Support:
Cancel	This is used by a client to cancel a request sent to the OSI coprocessor, before the coprocessor responds.	Server. The server side of the cancel service is invoked when the OSI coprocessor receives a cancel request from another MMS device.
Reject	This is a special service used by the MMS devices to signify protocol errors.	Not applicable. The concept of client/server does not apply to the reject service.
Status	This is used by a client to determine the general condition of the PLC–5 controller.	Server. The server side of the status service is invoked when the OSI coprocessor receives a status request from another MMS device.
UnsolicitedStatus	This is used by the PLC–5 controller in conjunction with the OSI coprocessor to automatically report its general condition to an MMS client.	Server. The server side of the UnsolicitedStatus service within the OSI coprocessor in invoked by programming a USTAT command into the PLC–5 controller.
GetNameList	This is used by an MMS client to obtain a list of the names associated with a particular type of MMS object that exist within the OSI coprocessor. Names associated with the following MMS objects can be requested: the domain, the program invocation, named variables, named variable lists, named types.	Server. The server side of the GetNameList service is invoked when the OSI coprocessor receives a GetNameList request from another MMS device.
Identify	This is used by an MMS client to obtain identification information associated with the OSI coprocessor. The information includes the vendor name (“Allen–Bradley Company”), the model name (e.g., PLC–5 MAP/OSI Interface) and the revision identifier of the OSI software that changes with each new software update.	Server. The server side of the Identify service is invoked when the OSI coprocessor receives an Identify request from another MMS device.
Rename	This is used by an MMS client to request that the OSI coprocessor change the identification string associated with a particular object.	Server. The server side of the Rename service is invoked when the OSI coprocessor receives a Rename request from another MMS device.
GetCapabilityList	This is used by an MMS client to request that the OSI coprocessor return the list of strings that identify device–specific capabilities that are associated with the VMD. The OSI coprocessor reports capabilities for the attached PLC–5 processor, series, revision and user memory size.	Server. The server side of the GetCapabilityList service is invoked when the OSI coprocessor receives a GetCapabilityList request from another MMS device.
InitiateDownload Sequence	This is used by an MMS client to request that the OSI coprocessor prepare the PLC–5 controller to receive a download of the program and data table image. Note that if a domain object or domain/program invocation object pair exists within the OSI coprocessor when this request is received, and the name of the domain given in the request is different from the name associated with the existing domain object, the OSI coprocessor automatically deletes both the program invocation and domain objects so the download can continue.	Server. The server side of the InitiateDownloadSequence service is invoked when the OSI coprocessor receives an InitiateDownloadSequence request from another MMS device.
DownloadSegment	This is used by the OSI coprocessor to request an MMS client send a piece of data associated with the download that is taking place.	Server. The server side of the DownloadSegment service is invoked by the OSI coprocessor by sending an InitiateDownloadSequence request to the MMS client when the OSI coprocessor wants to receive each piece of download data.
TerminateDownload Sequence	The OSI coprocessor sends this request to the MMS client when it realizes it has received all of the download data. This terminates the download.	Server. The OSI coprocessor automatically performs all functions associated with the server side of the TerminateDownloadSequence request when the final piece of download data has been received.
InitiateUpload Sequence	An MMS client uses this service to request that the OSI coprocessor prepare to upload the PLC–5 controller’s program and data memory.	Server. The server side of the InitiateUploadSequence service is invoked when the request is received.

MMS Service:	Description:	Support:
UploadSegment	An MMS client uses this service to request that the OSI coprocessor send a piece of the PLC–5 controller's memory image. This involves reading a block of the PLC–5 controller's program or data memory.	Server. The server side of the UploadSegment service is invoked when the request is received.
TerminateUpload Sequence	An MMS client uses this service to request that the OSI coprocessor terminate the upload that is currently taking place.	Server. The server side of the TerminateUploadSequence service is invoked when the request is received.
RequestDomain Download	The OSI coprocessor uses this service to ask an MMS client to perform a download to the PLC–5 controller.	Server. The server side of the RequestDomainDownload service is invoked when an MMS LoadDomainContent service request is received.
RequestDomain Upload	The OSI coprocessor uses this service to ask an MMS client to upload the PLC–5 controller's memory.	Server. The OSI coprocessor performs all functions associated with the server side of the RequestDomainUpload service without user intervention when an MMS StoreDomainContent service request is received.
LoadDomain Content	An MMS client uses this to invoke the server side of the RequestDomainDownload service within the OSI coprocessor. This is mainly used when an MMS client wants to tell the OSI coprocessor to request a download from a different MMS client.	Server. The server side of the LoadDomainContent service is invoked when the request is received.
StoreDomain Content	An MMS client uses this to invoke the server side of the RequestDomainUpload service within the OSI coprocessor. This is mainly used when the MMS client wants to tell the OSI coprocessor to request an upload from a different MMS client.	Server. The server side of the StoreDomainContent service is invoked when the request is received.
DeleteDomain	An MMS client uses this service to delete the domain object within the OSI coprocessor (the domain object provides access to the PLC–5 controller memory via the Domain Management Services). Note that this request does not cause the OSI coprocessor to clear the PLC–5 controller's memory, it only deletes the object within the coprocessor. The domain object must be deleted prior to downloading the PLC–5 controller. This can be done explicitly with the DeleteDomain service. Refer to the InitiateDownloadSequence description above for details on when the OSI coprocessor automatically deletes the domain object.	Server. The server side of the DeleteDomain service is invoked when the request is received.
GetDomain Attributes	An MMS client uses this service to obtain a list of characteristics associated with the domain object within the OSI coprocessor. This service is provided for client applications that make decisions based on these characteristics.	Server. The server side of the GetDomainAttributes service is invoked when the request is received.
CreateProgram Invocation	An MMS client uses this service to create a program invocation object within the OSI coprocessor. This enables the MMS client to control the operational state of the PLC–5 controller's program (e.g., start it, stop it). The OSI coprocessor implements a single program invocation object by which an MMS client can place the PLC–5 controller in program, test, or run mode. During the powerup cycle, a default program invocation object is created if one did not previously exist. As part of the download sequence, the program invocation object must be deleted. This service is provided to recreate the program invocation after a download occurs.	Server. The server side of the CreateProgramInvocation service is invoked when the request is received.
DeleteProgram Invocation	An MMS client uses this service to delete the program invocation object within the OSI coprocessor. When the program invocation object is deleted, an MMS client cannot control the operational state of the PLC–5 controller program. Note that this does not cause the OSI coprocessor to delete the program memory within the PLC–5 controller. Once the program invocation object within the OSI Interface has been deleted, the domain object can be explicitly deleted using the DeleteDomain service. Refer to the InitiateDownloadSequence description above for details on when the OSI Interface automatically deletes the program invocation object.	Server. The server side of the DeleteProgramInvocation service is invoked when the request is received.

MMS Service:	Description:	Support:
Start	An MMS client uses this service to put the PLC–5 controller into run or test mode. With this request, the client can specify an "execution argument" string that details whether the PLC–5 controller should enter run or test mode. If the start argument equals "RUN" or "run", the controller is placed into run mode. If the execution argument is "TEST" or "test", the controller is placed into test mode. The OSI coprocessor remembers the last execution argument specified in a start request and uses this when subsequent start requests are received that do not specify an execution argument. If a start is received that does not specify an execution argument and no start arguments have previously been received, the PLC–5 controller is placed into run mode (the default is to go to run mode). This service automatically re–starts any Sequential Function Chart (SFC) Programs.	Server. The server side of the Start service is invoked when the request is received.
Stop	An MMS client uses this service to put the PLC–5 controller into program mode. This stops the execution of the PLC–5 controller program.	Server. The server side of the Stop service is invoked when the request is received.
Resume	An MMS client uses this service to resume execution within the PLC–5 controller program from the point at which it was last stopped. An execution argument can be specified with this service, the same as with the Start service. If the PLC–5 program was stopped in an SFC program, this service will cause the SFC program to continue immediately after the last step executed.	Server. The server side of the Resume service is invoked when the request is received.
Reset	An MMS client would use this service to reset the PLC–5 controller program to its beginning point after it has been stopped.	Server. The server side of the reset service is invoked when the request is received.
GetProgram Invocation Attributes	An MMS client uses this service to obtain a list of characteristics associated with the program invocation object within the OSI coprocessor. This service is provided for client applications that make decisions based on these characteristics.	Server. The server side of the SetProgramInvocationAttributes service is invoked when the request is received.
Read	Provides a mechanism for an MMS device to read data that exists within the PLC–5 system or for the PLC–5 controller to read data from an MMS device.	Client and Server. The client side of the Read service within the OSI coprocessor is invoked by programming a MOVE command within a PLC–5 system message instruction (see Chapter 4 for more details). The server side of the Read service is invoked when the request is received.
Write	Provides a mechanism for another MMS device to write data within the PLC–5 system or for the PLC–5 controller to write data to another system.	Client and server. The client side of the Write service within the OSI coprocessor is invoked by programming a MOVE command within the PLC–5 controller message instruction (see Chapter 4 for more details). The server side of the Write service is invoked when the request is received.
InformationReport	This is used by the PLC–5 system to automatically send data to another system without the other system asking for it and without requiring the other system to send a response. The PLC–5 controller does not tell the other system where this data is to be stored (as it does with a write request), it simply sends it to the other system and assumes it will be handled appropriately.	Server. The server side of the InformationReport service within the OSI coprocessor is invoked by programming a UINFO command within a PLC–5 message instruction.
GetVariableAccess Attributes	This is used by an MMS client to obtain information relative to the type of data associated with a PLC–5 controller data address. This service may also be used to get the type of data as well as the address associated with a named variable object.	Server. The server side of the GetVariableAccessAttributes service is invoked when the request is received.

MMS Service:	Description:	Support:
DefineNamed Variable	This is used by the PLC–5 system, in conjunction with the OSI coprocessor, to define an MMS named variable. An MMS client can also use this to define a named variable within the coprocessor. The named variable would reference a location within the PLC–5 system.	Client and server. The client side of the DefineNamedVariable service within the OSI coprocessor is invoked by programming a DEFVAR command within a PLC–5 system message instruction. The server side is invoked when the OSI coprocessor receives a DefineNamedVariable request. The server side does not support domain–specific named variables.
DeleteVariable Access	This is used by the PLC–5 system in conjunction with the OSI coprocessor to delete an MMS named variable. An MMS client uses this to delete a named variable within the OSI coprocessor. Note that when this service is directed to an OSI coporcessor, no PLC–5 system memory is deleted, only the named variable reference within the interface is deleted.	Client and server. The client side of the DeleteVariableAccess service within the OSI coprocessor is invoked by programming a DELVAR command within a PLC–5 system message instruction. The server side is invoked when the OSI coprocessor receives a DeleteVariableAccess request.
DefineNamed VariableList	An MMS client uses this to define an identifier string that references a list of variable objects. Thus, by specifying a single name in a read or write request, the client could access multiple, possibly discontiguous, PLC–5 controller memory locations.	Server. The server side of the DefineNamedVariableList service is invoked when the OSI coprocessor receives a DefineNamedVariableList request. The server side does not support domain–specific named variable lists.
GetNamedVariable ListAttributes	An MMS client uses this to obtain a list of the variable objects that are members of the named variable list.	Server. The server side of the GetNamedVariableListAttributes service is invoked when the OSI coprocessor receives a GetNamedVariableListAttributes request.
DeleteNamed VariableList	An MMS client uses this to delete a previously defined named variable list. Note that this does not delete any memory within the PLC–5 system. It only removes the named variable list object from the OSI coprocessor.	Server. The server side of the DeleteNamedVariableList service is invoked when the OSI coprocessor receives a DeleteNamedVariableList request.
DefineNamedType	An MMS client uses this service to define a “data type template” to be used at a later time when either accessing memory within the PLC–5 system (using the read and/or write service) or when defining named variables (using the DefineNamedVariable service within the OSI coprocessor).	Server. The server side of the DefineNamedType service is invoked when the OSI coprocessor receives a DefineNamedType request.The server side does not support domain–specific named types.
GetNamedType Attributes	An MMS client uses this service to obtain the characteristics associated with a previously defined named type within the OSI coprocessor. When the OSI coprocessor receives this request, it returns the “data type template” that was previously defined.	Server. The server side of the GetNamedTypeAttributes service is invoked when the OSI coprocessor receives a GetNamedTypeAttributes request.
DeleteNamedType	An MMS client uses this service to remove a previously defined named type from the OSI coprocessor.	Server. The server side of the DeleteNamedType service is invoked when the OSI coprocessor receives a DeleteNamedType request.

Additional Information Regarding MMS Services

- All addresses must be in MMS ‘‘Symbolic Address” format
- VMD–specific and Application Association–specific named variables, named variable lists, and named types are supported. Domain–specific named variables, named variable lists, and named types are not supported.
- There is no fixed limit for the number of variables in a ‘‘list of variables” for the variable access services. The number is limited only due to negotiated PDU size and available memory within the coprocessor.
- The list of capabilities is always reported as NULL in GetDomainAttributes and you must specify as NULL in InitiateDownloadSequence.
- The maximum PDU size supported is 1800 octets.

Mapping MMS Data Types onto PLC–5 Controller Data Files

The MMS specification lists a set of **data types,** or formats, that your variables are allowed to take. The data types allowed for your OSI coprocessor are:

- boolean
- bitstring
- integer
- unsigned integer
- BCD
- floating point
- visible string
- octet string
- generalized time
- structure
- array

These data types directly correspond to areas in PLC controller memory that contain **data tables.** These data tables store information, in different data–type categories, that your application programs can access. The following list shows the PLC–5 controller's data tables you are allowed to access through your applications:

- ASCII (A)
- Binary (B)
- Signed Word (N)
- Floating Point (F)
- Timer (T)
- Counter (C)
- Input Image (I)
- Output Image (O)
- Block Transfer Data (BT)
- Control (R)
- Status (S)
- String (ST)
- PID Control (PD)
- Message Control (MG)
- BCD Data (D)
- Sequential Function Chart (SC)
- Token Data (TD)

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of data type structures. See Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

The sections that follow provide sample mappings of each PLC controller data table category listed above. We use the following conventions in our examples:

This:	Means:
B3:0,n	a bit–string of length n starting at bit 0 of the specified word.
B3:0/x,n	a bit–string of length n starting at bit x (octal) of the specified word.

The following sections show some mappings that form structures. In these cases, we have also included examples of the structure format as it is viewed by MMS. For the mappings that are not structures, this is not necessary.

You can use both upper and lower case letters when mapping MMS data types onto PLC controller data files; the use of a semi–colon (;) in place of a colon (:) is also allowed.

ASCII (A) Mappings

The following table contains samples of mapping ASCII data types.

Address:	MMS Data Type:
a10:0	VISIBLE STRING (1 byte)
a10:0,10	VISIBLE STRING (10 bytes)

Alternate mappings are available, see the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Binary (B) Mappings

The following table contains samples of mapping binary data types.

Address:	MMS Data Type:
B3:0	BIT STRING (16 bits)
B3:0,25	BIT STRING (25 bits)
B3:0/0	BOOLEAN
B3:0/3,5	BIT STRING (5 bits)

Signed Word (N) Mappings

The following table contains samples of mapping signed word data types.

Address:	MMS Data Type:
N7:0	INTEGER (16 bits)
N7:0,8	ARRAY of 8 INTEGERS
N7:0/0	BOOLEAN
N7:0/3,5	BITSTRING (5 bits)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Floating Point (F) Mappings

The following table contains samples of mapping floating point data types.

Address:	MMS Data Type:
F8:0	FLOATING POINT
F8:0,50	ARRAY of 50 FLOATING POINTS

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Timer (T) Mappings

The following is the format of a timer structure as viewed by MMS.

```
BOOLEAN     DN
BOOLEAN     TT
BOOLEAN     EN
INTEGER     PRE (16 bits)
INTEGER     ACC (16 bits)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table contains samples of mapping timer data types.

Address:	MMS Data Type:
T4:15	STRUCTURE of type TIMER
T4:3,10	ARRAY of 10 TIMER STRUCTURES
T4:1.DN	BOOLEAN
T4:1.TT	BOOLEAN
T4:1.EN	BOOLEAN
T4:0.ACC	INTEGER
T4:13.PRE	INTEGER
T4:0.DN,4	ARRAY of 4 BOOLEANS (This specifies 4 done bit members of TIMERS 0 to 3. This also applies to TT and EN members.)
T4:0.ACC,50	ARRAY of 50 INTEGERS (This example specifies 50 accumulator members of TIMERs 0 to 49. This also applies to PRE.)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Counter (C) Mappings

The following is the format of a counter structure as viewed by MMS:

```
BOOLEAN     UN
BOOLEAN     OV
BOOLEAN     DN
BOOLEAN     CD
BOOLEAN     CU
INTEGER     PRE (16 bits)
INTEGER     ACC (16 bits)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table contains samples of mapping counter data types.

Address:	MMS Data Type:
C5:15	STRUCTURE of type COUNTER
C5:3,10	ARRAY of 10 COUNTER STRUCTURES
C5:3.UN	BOOLEAN
C5:1.OV	BOOLEAN
C5:1.DN	BOOLEAN

Address:	MMS Data Type:
C5:4.CD	BOOLEAN
C5:1.CU	BOOLEAN
C5:0.ACC	INTEGER
C5:13.PRE	INTEGER
C5:0.DN,4	ARRAY of 4 BOOLEANS (This example specifies 4 done bit members of COUNTERS 0 to 3. This also applies to the other BOOLEAN members.)
C5:0.ACC,50	ARRAY of 50 INTEGERS (This example specifies 50 accumulator members of COUNTERS 0 to 49. This also applies to PRE member.)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Input Image (I) Mappings

The table that follows provides sample mappings of input image data types, where:

I:000/x,n is a bit–string of length n starting at bit x (octal) of the specified word.

Important: Note that addressing in the Input Image Mapping is in octal. This does not pertain to the ‘‘size” information. For example, ‘‘I:3/5,10” specifies a bit string of length 10 (vs. length 8).

Address:	MMS Data Type:
I:000	BIT STRING (16 bits)
I:001,25	BIT STRING (25 bits)
I:000/10	BOOLEAN
I:000/3,5	BIT STRING (5 bits)

See the section titled *Additional Information on Using Data Types* and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Output Image (O) Mappings

The table that follows provides sample mappings of output image data types, where:

O:000/x,n is a bit–string of length n starting at bit x (octal) of the specified word.

Important: Note that addressing in the Output Image Mapping is in octal. This does not pertain to the ‘‘size” information. For example, ‘‘O:003/5,10” specifies a bit string of length 10 (vs. length 8).

Address:	MMS Data Type:
O:000	BIT STRING (16 bits)
O:001,25	BIT STRING (25 bits)
O:000/10	BOOLEAN
O:001/3,5	BIT STRING (5 bits)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Block Transfer (BT) Mappings

The following is the format of a block transfer control structure as viewed by MMS:

```
BOOLEAN     RW
BOOLEAN     TO
BOOLEAN     NR
BOOLEAN     EW
BOOLEAN     CO
BOOLEAN     ER
BOOLEAN     DN
BOOLEAN     ST
BOOLEAN     EN
INTEGER     RLEN (16 bits)
INTEGER     DLEN (16 bits)
INTEGER     FILE (16 bits)
INTEGER     ELEM(16 bits)
INTEGER     RGS (16 bits)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides sample mappings of block transfer read data types.

Address:	MMS Data Type:
BT9:0	STRUCTURE of type BLOCK TRANSFER
BT9:0,10	ARRAY of 10 BLOCK TRANSFER STRUCTURES
BT9:0.RW	BOOLEAN (applies to all other BOOLEAN members)
BT9:0.RW,5	ARRAY of 5 BOOLEANS (applies to all other BOOLEAN members)
BT9:0.RLEN	INTEGER (applies to all other INTEGER members)
BT9:0.RLEN,5	ARRAY of 5 INTEGERS (applies to DLEN, FILE, ELEM, and RGS.)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Control (R) Mappings

The following is the format of a control structure as viewed by MMS:

```
BOOLEAN    FD
BOOLEAN    IN
BOOLEAN    UL
BOOLEAN    ER
BOOLEAN    EM
BOOLEAN    DN
BOOLEAN    DU
BOOLEAN    EN
INTEGER    LEN (16 bits)
INTEGER    POS (16 bits)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides samples of mapping control data types.

Address:	MMS Data Type:
R6:0	STRUCTURE of type CONTROL
R6:2,4	ARRAY of four CONTROL STRUCTURES
R6:1.FD	BOOLEAN (applies to all other BOOLEAN members)
R6:1.FD,6	ARRAY of 6 BOOLEANS (applies to all other BOOLEAN members)
R6:1.LEN	INTEGER
R6:1.POS	INTEGER
R6:1.POS,5	ARRAY of 5 INTEGERS (applies to all other INTEGER members)

See the section titled *Additional Information on Using Data Types* and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Status (S) Mappings

The following table provides samples of mapping status data types.

Address:	MMS Data Types:
S:10	INTEGER (16 bits)
S:0,30	ARRAY of 30 INTEGERS
S:0/0	BOOLEAN
S:0/3,5	BITSTRING (5 bits)

See the section titled *Additional Information on Using Data Types,* later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

String (ST) Mappings

The following table provides samples of mapping string data types.

Address:	MMS Data Types:
ST10:0	VISIBLE STRING (0 to 82 printable characters)
ST10:2,3	ARRAY of three VISIBLE STRINGS

Alternate mappings are available, see the section titled *Additional Information on Using Data Types,* later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

PID Structure (PD) Mappings

The following is the format of a PID structure as viewed by MMS:

BOOLEAN	PE
BOOLEAN	MO
BOOLEAN	CA
BOOLEAN	SWM
BOOLEAN	DO
BOOLEAN	PVT
BOOLEAN	CL
BOOLEAN	CT
BOOLEAN	EN
BOOLEAN	PVHA
BOOLEAN	PVLA
BOOLEAN	DVPA
BOOLEAN	DVNA
BOOLEAN	EWD
BOOLEAN	OLH
BOOLEAN	OLL
BOOLEAN	SPOR
BOOLEAN	INI
FLOATING POINT	SP
FLOATING POINT	KP
FLOATING POINT	KI
FLOATING POINT	KD
FLOATING POINT	BIAS
FLOATING POINT	MAXS
FLOATING POINT	MINS
FLOATING POINT	DB
FLOATING POINT	SO
FLOATING POINT	MAXO
FLOATING POINT	MINO
(FLOATING POINT	UPD
FLOATING POINT	PV
FLOATING POINT	ERR
FLOATING POINT	OUT
FLOATING POINT	PVH
FLOATING POINT	PVL
FLOATING POINT	DVP
FLOATING POINT	DVN
FLOATING POINT	PVDB
FLOATING POINT	DVDB
FLOATING POINT	MAXI
FLOATING POINT	MINI
FLOATING POINT	TIE
OCTET STRING	ADDR (SIZE = 8)
OCTET STRING	DATA (SIZE = 56)

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides samples of mapping PID data types.

Address:	MMS Data Types:
PD10:0	STRUCTURE of type PID
PD10:1,2	ARRAY of two PID STRUCTURES
PD10:0.PE	BOOLEAN
PD10:2.EN	BOOLEAN (applies to all other BOOLEAN members)
PD10:2.EN,5	ARRAY of 5 BOOLEANS (applies to all other BOOLEAN members)
PD10:5.SP	FLOATING POINT
PD10:5.BIAS	FLOATING POINT (applies to all other FLOATING POINT members)
PD10:5.BIAS,4	ARRAY of 4 FLOATING POINTS (applies to all other FLOATING POINT members)
PD10:0.ADDR	OCTET STRING (applies to DATA)

See the section titled *Additional Information on Using Data Types,* later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Message Control (MG) Structure Mappings

The following is the format of a message control structure as viewed by MMS:

```
BOOLEAN         EW
BOOLEAN         CO
BOOLEAN         ER
BOOLEAN         DN
BOOLEAN         ST
BOOLEAN         EN
BOOLEAN         TO
BOOLEAN         NR
INTEGER         ERR (16 bit)
INTEGER         RLEN (16 bits)
INTEGER         DLEN (16 bits)
OCTET STRING    DATA  (SIZE = 104)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides samples of mapping message control data types.

Address:	MMS Data Types:
MG9:0	STRUCTURE of tyoe MESSAGE CONTROL
MG9:1,3	ARRAY of 3 MESSAGE CONTROL STRUCTURES
MG9:0.ER	BOOLEAN
MG9:0.TO,9	ARRAY of 9 BOOLEANS (applies to all other BOOLEAN members).
MG9:0.ERR	INTEGER (applies to all other INTEGER members).
MG9:0.DLEN,5	ARRAY of 5 INTEGERS (applies to all other INTEGER types).
MG9:0.DATA	OCTET STRING

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

BCD Data (D) Mappings

The following table provides samples of mapping BCD data types:

Address:	MMS Data Types:
D9:0	single BCD
D9:0,10	ARRAY of 10 BCDs

Alternate mappings are available, see the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Sequential Function Chart Status (SC) Mappings

The following is the format of an sequential function chart structure as viewed by MMS:

```
BOOLEAN     DN
BOOLEAN     ER
BOOLEAN     OV
BOOLEAN     LS
BOOLEAN     FS
BOOLEAN     SA
INTEGER     PRE (SIZE = 16)
INTEGER     TIM (SIZE = 16)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides samples of mapping sequential function chart data types.

Address:	MMS Data Type:
SC13:0	STRUCTURE of type SFC
SC13:0,2	ARRAY of 2 SFC STRUCTURES
SC13:0.DN	BOOLEAN
SC13:0.ER,4	ARRAY of 4 BOOLEANS (applies to all other BOOLEAN types)
SC13:0.PRE	INTEGER (applies to TIM)
SC13:0.PRE,3	ARRAY of 3 INTEGERS (also applies to TIM)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Token Data (TD) Mappings

The following is the format of a token data structure as viewed by MMS:

```
INTEGER     HI (SIZE = 16)
INTEGER     LO (SIZE = 16)
```

Important: Refer to your PLC–5 Programming Reference Manual for descriptions of the individual fields in data type structures.

The following table provides samples of mapping token data data types:

Address:	MMS Data Type:
TD15:0	STRUCTURE of type TOKEN DATA
TD15:0,10	ARRAY of 10 TOKEN DATA STRUCTURES
TD15:0.HI	INTEGER (also applies to LO.)
TD15:0.HI,5	ARRAY of 5 INTEGERS (applies to LO)

See the section titled *Additional Information on Using Data Types*, later in this chapter, and Appendix A of this manual for more information on mapping MMS data types onto PLC–5 data files.

Additional Information on Using Data Types

The following list contains general information on some of the data types listed in the preceding sections.

- You can access all integer locations with either 16 or 32 bit sizes in an MMS Type Specification. The number shown in parenthesis in the table in Appendix A indicate the default size.
- You can access all integer locations as unsigned integer. To **read** integer locations as unsigned, an MMS Type Specification **is** required. To **write** integer locations as unsigned, an MMS Type Specification is **not** required.
- If you want to access the integer (N) or status (S) sections as binary data (MMS boolean or MMS bitstring), a bit–position specifier (/) must be present in the address. For example, if you want to read bit 0 of N7:0, the ‘‘N7:0/0” must be specified as the address even if an MMS Type Specification stating boolean or bitstring is specified. If there is no bit–position specifier (/) in the address, then binary access is not allowed.
- The maximum integer value that you can write into a BCD location is 9,999. To **read** BCD as an integer, an MMS Type Specification **is** required. To **write** BCD as an integer, an MMS Type Specification is **not** required.
- You must access the ASCII file as a fixed length VisibleString or OctetString. The maximum length of the string is bounded by the negotiated segment size and/or the size of the file.
- You access the string (ST) file as a VisibleString or an OctetString. You can use an indefinite length; the maximum is 82 bytes. To read as an octet string, an MMS Type Specification is required. To write as an octet string, no MMS Type Specification is required.
- You can access MMS arrays of the default type by providing the appropriate MMS Type Specification **or** by specifying ‘‘size” information in the address (for example N7:0,5 for an array of 5 integers). You note this with a comma (,), followed by the size. MMS Type Specification **and** the ‘‘size” information can not both be present, or an error is returned. There are instances where MMS array access is not allowed. They are listed below:
  - arrays of bitstrings are **never** allowed
  - array access into ASCII (‘‘A”) file is not allowed
  - array access of the ‘‘DATA” member of the message structure is not allowed.
  - array access of the ‘‘ADDR” member of the PID structure is not allowed.
  - array access of the ‘‘DATA” member of the PID structure is not allowed.

- arrays of arrays are **never** allowed.

MMS Object Management

This section covers MMS object management operations, including:

- retaining MMS objects in non–volatile memory
- saving MMS objects to, and restoring them from, DOS files
- information on MMS named variables that always exist in the OSI coprocessor

Retaining MMS Objects

You can **retain** the following MMS objects in the OSI coprocessor's non–volatile memory:

- domain
- program invocation
- named variable
- named variable list
- named type

To retain MMS objects in non–volatile memory, you set the *Retain MMS Objects* parameter through your Allen–Bradley MAP Station Manager. This parameter instructs the OSI coprocessor to retain MMS objects on a continuous basis:

If the *Retain MMS Objects* parameter is set to:	**Then:**
No	MMS objects are not being saved. Only default objects will exist if there is a power failure or reset.
Yes	your MMS objects are saved to non–volatile memory continuously. If the coprocessor is reset or power is lost, the MMS objects are retained, and are then restored during powerup processing.

Note that your OSI coprocessor stores the MMS object reference only. No values associated with the object are saved by the OSI coprocessor.

The restored set of MMS objects remains stored until you do one of the following:

- perform the *Use Default MMS Objects* action via the Allen–Bradley MAP Station Manager

Important: When default MMS objects are restored, the OSI coprocessor aborts all connections and resets itself after completing the restoration.

- restore a new set of MMS objects from a file

- remove the power and the battery from the OSI coprocessor, resulting in the loss of non–volatile memory.
- cause the auto clear function to clear non–volatile memory (see the section titled *Switch 3* in Chapter 2 for details of the auto clear function).

For procedures on retaining MMS objects, refer to the Allen–Bradley MAP Station Manager User's Manual (publication 6630–6.5.2). For more information on non–volatile memory, see the section titled *Non–volatile Memory and the Lithium Battery* in Chapter 2.)

Saving MMS Objects to and Restoring Them From a File

You can **save** the MMS objects that are stored in non–volatile memory to a DOS file in your Allen–Bradley MAP Station Manager directory. This is a way of having a copy of your MMS objects, so that if they are lost, you can re–load them to the OSI coprocessor.

When you **restore** the objects from the file, they are loaded to the OSI coprocessor's non–volatile memory (note, that you will overwrite any existing information in non–volatile memory when you restore the objects).

These procedures are called *Save MMS Objects to a File* and *Restore MMS Objects from a File*. See the MAP Station Manager user manual for instructions.

Important: When you restore MMS objects from a file, the OSI coprocessor aborts all connections and resets itself after completing the restoration.

MMS Named Variables That Always Exist in the OSI Coprocessor
Along with the MMS named variables that you define, your OSI coprocessor also supports other named variables that are pre–defined and always exist. These are automatically created when you power–up the OSI coprocessor and exist as long as the OSI coprocessor is in **fully operational mode** (you cannot delete them). The variables are listed below:

- **AB_O5_BATTERY_LOW**: this named variable is of type *boolean* and indicates whether or not the battery has sufficient power to retain non–volatile memory if power is removed from the OSI coprocessor. If the value of this variable is **true**, the battery power is not sufficient to retain memory when power is removed from the OSI coprocessor. The value will be **false** otherwise. You can read this variable using MMS services, but you cannot write it.
- **AB_O5_SWITCH_SETTINGS*:*** This named variable is of type *bit string* and indicates the values of the coprocessor's switch settings. The

value 1 indicates the switch is ON, and the value 0 indicates the switch is OFF. You can read this variable using MMS services, but you cannot write it.

- **AB_PLC5_LAST_EDIT_TIME**: this named variable is of type *generalized time* and indicates the last time the user program was edited. This is a time ‘‘marker” that is updated when you:
  - clear the PLC–5 controller’s memory
  - insert or remove instructions
  - create or delete program files
  - perform set or reset test edits
  - perform assemble edits

  You can read this variable using MMS services, but you cannot write it.
- **AB_PLC5_PROG_CHECKSUM**: This named variable is a 16–bit signed integer and indicates the checksum of the PLC–5 controller user program. This is continuously updated by the PLC–5 controller as the program changes. You can read this variable using MMS services, but you cannot write it. This variable is the same as S:57 in the PLC–5 controller.
- **M_DAYTIME**: this named variable indicated the time of day and is of type *generalized time*. This variable provides direct access to the ‘‘real–time clock” in the status section of the OSI coprocessor. You can read and write this variable using MMS services.

Note that these MMS variables are defined automatically when your coprocessor is powered up and therefore you are not able to erase them from memory.

MMS Security

The OSI coprocessor has a security mechanism that allows you to restrict the access that remote applications have to the following MMS services:

- Rename
- InitiateDownloadSequence
- DeleteDomain
- CreateProgramInvocation
- DeleteProgramInvocation
- Start
- Stop
- Resume
- Reset
- Write
- DefineNamedVariable
- DeleteVariableAccess
- DefineNamedVariableList
- DeleteNamedVariableList
- DefineNamedType

- DeleteNamedType

MMS security acts as a ‘‘master switch” you set either ON or OFF through the MAP Station Manager. With it, you can choose to either give remote applications the access to these 16 services **(maximum privilege)** or no access to these services **(minimum privilege).**

When MMS security is set to:	Then:
OFF (disabled)	MMS security is **off** and all remote MMS applications have access to the 16 services (in addition to all other MMS services).
ON (enabled)	MMS security is **on** and only MMS applications with security level 1 have access to the 16 services (in addition to all other MMS services).

The Allen–Bradley default setting for MMS Security is OFF.

There is a field in each remote application’s LDIB entry that indicates the level of security for that application. The security field is where you assign the remote application one of the two security levels:

If you assign a remote application:	Then, when MMS security is ON, that remote MMS application has:
security level 0	minimum privilege, and therefore has no access to the 16 MMS services.
security level 1	maximum privilege, and therefore has access to the 16 MMS services.

Note that the security field is only relevant when MMS security is ON, when security is OFF, the field is meaningless.

Important: When MMS security is ON, all remote applications connecting to the OSI coprocessor will be given **minimum security** by default. **Maximum security** will be given to only remote applications that meet the following criteria:

- you have an LDIB entry for that remote application
- the LDIB security field is equal to 1 for that remote application
- you have an AP title and AE qualifier entry in the LDIB for that remote application
- the AP title and AE qualifier information sent from that remote application matches what is in the LDIB for that remote application

The MMS security mechanism does not affect the ability of remote MMS applications to connect (establish an association) to the OSI coprocessor, only the **services** supported on that connection.

Refer to the Allen–Bradley MAP Station Manager user manual (publication 6630–6.5.2) for instructions on setting MMS Security and setting Privileges.

Chapter 4

Basic Programming Techniques

Chapter Objectives

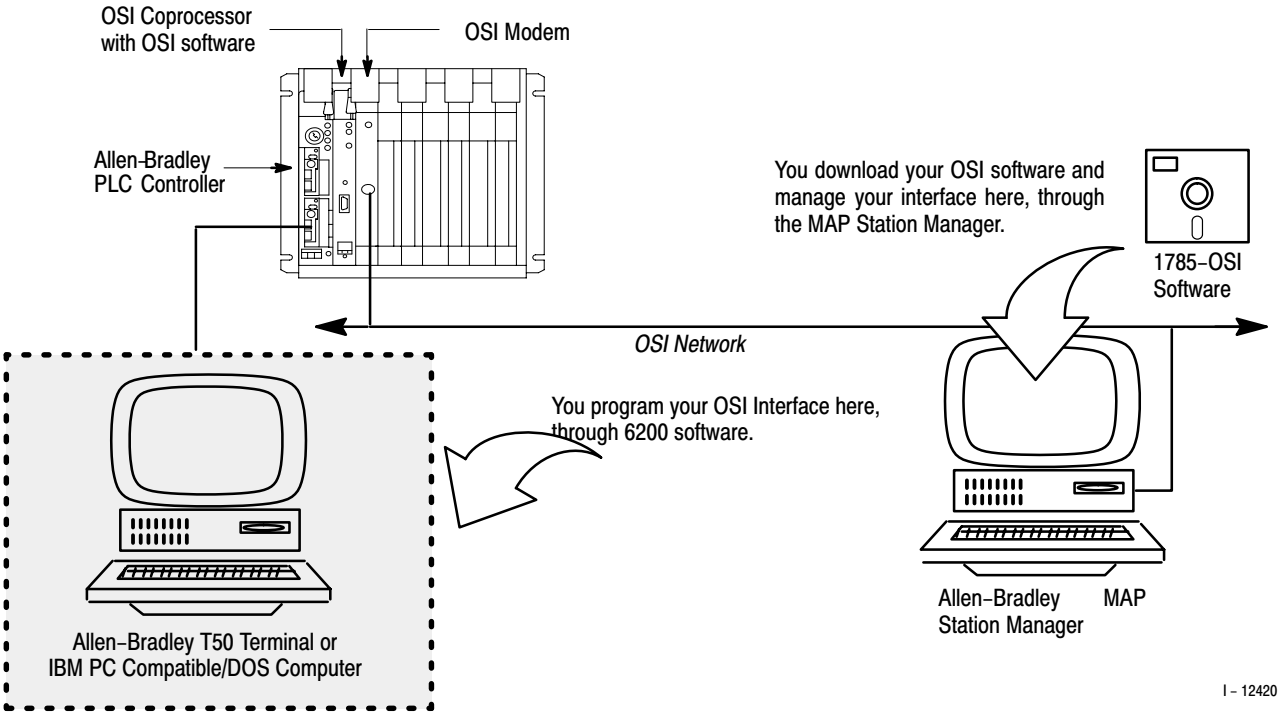
Use the information contained in this chapter to program your OSI coprocessor. This chapter contains the following sections:

- Introduction
- Entering Commands
- Things You Should Know Before You Program
- A Quick Reference Guide to the Commands
- Managing Connections (the OPEN, CLOSE, and ABORT Commands)
- Defining MMS Named Variables (the DEFVAR Command)
- Deleting MMS Named Variables (the DELVAR Command)
- Reading and Writing Data (using the SET Command)

Introduction

You send messages via Allen–Bradley 6200 software to the MAP network through the PLC–5 programming commands (figure 4.1).

Figure 4.1
The 1785–OSI Software Environment



Entering Commands

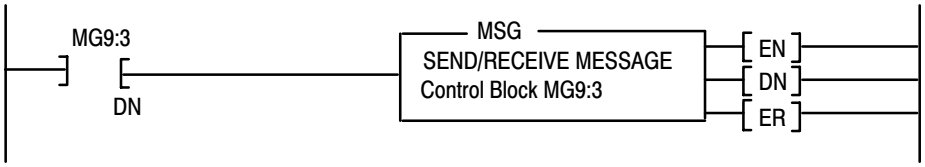
You enter the commands and qualifiers through the PLC–5 programming software (cat. no. 6200–PLC5, version 4.3 or higher) running on either an Allen–Bradley T50 industrial terminal (1784–T50) or a personal computer.

You enter the commands and qualifiers in a command line to create a PLC–5 **message instruction**. The following is an example of a typical command line containing a command and qualifiers:

```
MOVE FROM :C1'TIMER_1' TO 'TIMER_2'
```

Figure 4.2 shows a sample message instruction display of the data from the command line above.

Figure 4.2
Sample Message Instruction



I – 12421

The procedure for using the PLC–5 programming software is listed below. You can exit from this procedure at any time by pressing the [ESC] key.

Important: We assume you are familiar with programming Allen–Bradley PLC controllers using the PLC–5 programming software.

1. Position the cursor on the rung display where you want to insert the message instruction.

2. Press [F10], the *edit* option. A new set of options appears.

3. Press [F4], the *insert rung* option. A new set of options appears.

4. Press [F4], the *insert instruction* option. A new set of options appears.

5. Press [F10], the *others* option. A new set of options appears.

6. Press [F3], the *I/O Message* option. A new set of options appears.

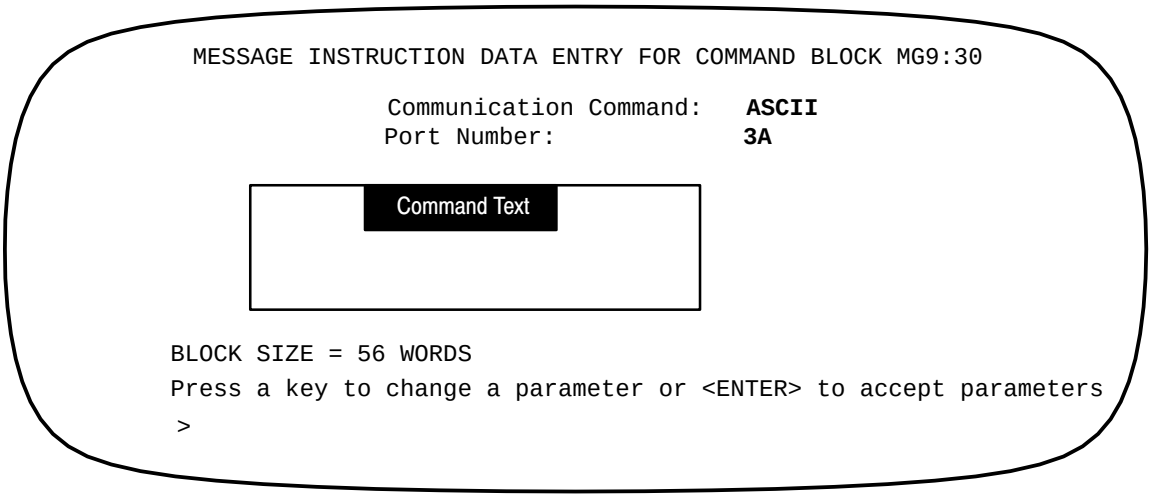
7. Press [F5], the *MSG* option. The following message appears:

```
Enter Message Control Block address >
```

8. Enter a valid MSG address (for example MG9:30), and press [ENTER]. The following message appears:

```
Press a key to change a parameter or <ENTER>
to accept parameters
```

9. Press [F1], the *Command Type* option as many times necessary until you see a screen like:



10. Press [F2]. The following message appears:

```
Enter Port Number.
```

11. At the > prompt, type:

```
3A [ENTER]
```

12. Press [F3] to enter command text. Press [ENTER] three times when complete.

13. Press [F10], the *accept rung* option.

What You Should Know Before You Program

The following sections provide a brief introduction to programming your OSI coprocessor. This includes:

- the Allen–Bradley programming commands
- the qualifiers
- general rules for using MMS named variables
- general rules for using address strings (MMS unnamed variables)
- important information on outstanding network messages

If you are already familiar with these topics, you can skip ahead to the individual command sections later in this chapter.

The Allen–Bradley Commands

The following table contains the programming software commands you can use to program the OSI coprocessor. The table also shows the related MMS service functions for each command.

Command:	Related MMS Service:	Description:
OPEN	Initiate	Initiates a connection with a remote node.
CLOSE	Conclude	Concludes a connection that was opened with a remote node.
ABORT	Abort	Aborts a connection.
DEFVAR	DefineNamedVariable	Defines an MMS named variable.
DELVAR	DeleteVariableAccess	Deletes an MMS named variable.
SET	Read/Write	Transfers data across the MAP network.

There are other commands you can use to program the coprocessor. Refer to Chapter 5, *Additional Programming Techniques,* for more information.

Qualifiers

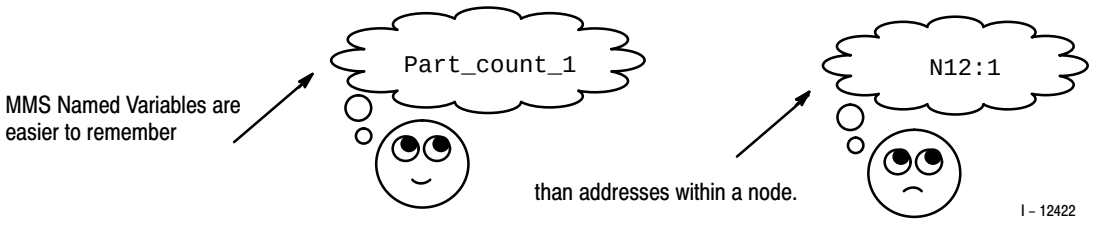
A *qualifier* is a word used to specify the details of data transfer. The qualifiers for PLC–5 programming software are:

This qualifier:	Allows you to specify:
TO	the destination of the data transfer
FROM	the source of the data being transferred

General Rules for Using MMS Named Variables

The MMS protocol allows you to associate a name with an address within the node. These associated names are called MMS named variables. It is easier, for example, to remember *Paint_station_1* than it is to remember *N12:1* (figure 4.3). You can define MMS named variables and use them within command lines when programming.

Figure 4.3
MMS Named Variables are Easier to Remember



You define MMS named variables by using the DEFVAR command (this command is covered in detail in the section titled *Defining MMS Named Variables*, later in this chapter).

There are rules for the using MMS named variables. Both local and remote variables:

- must be in single quotes
- can be up to 32 characters long
- can contain letters A to Z (both upper and lower case)
- can contain an underscore (_) and a dollar sign ($)
- can contain numerals 0 through 9
- cannot **start** with a numeral
- must be in proper symbol format for local/remote device

For example:

```
'I_am_an_MMS_Named_Variable'
```

General Rules for Using Address Strings (MMS Unnamed Variables)

There are rules for the use of address strings (MMS unnamed variables) within PLC–5 message block syntax. Both local and remote address strings:

- must be in double quotes
- can be up to 32 characters long
- must be in proper address format for local/remote device

For example:

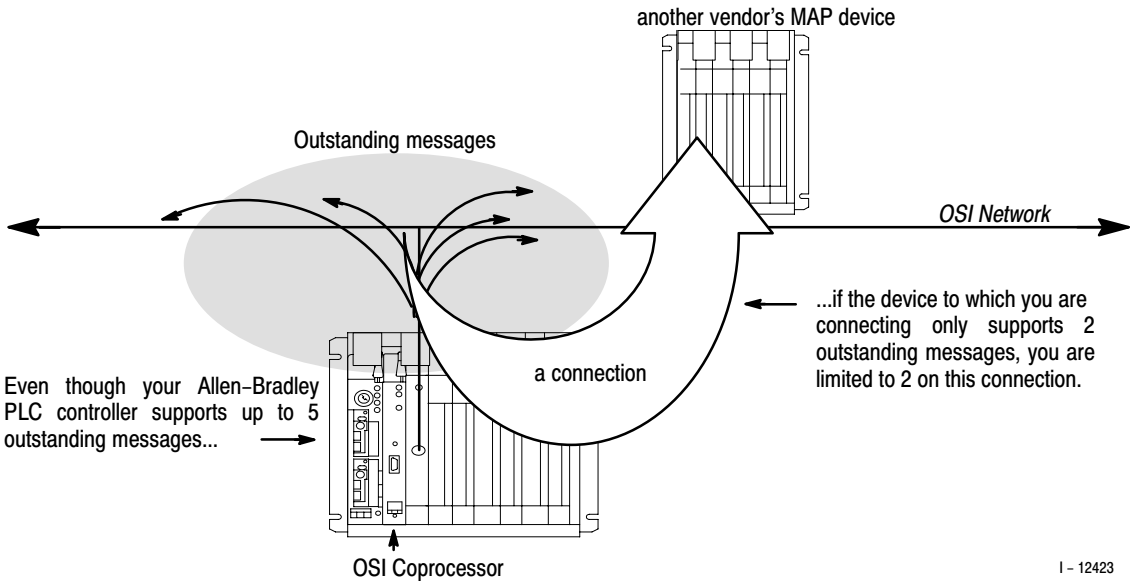
```
"B7:6,10"
```

Important Information on Outstanding Network Messages

You can establish up to 16 concurrent connections from your OSI coprocessor. You are limited in the number of outstanding messages you are allowed to have at one time. By outstanding messages we mean messages that have had no response. You are allowed up to 32 outstanding messages because that is what your PLC–5 controller allows at one time. This limit applies to the **total number of connections** you have. There is a limit of 5 outstanding messages per connection.

Note that even though your PLC controller supports up to 5 outstanding messages per connection, the number you are allowed also depends upon the device to which you are connecting. If, for example, you are connecting to a device that supports only 2 outstanding messages per connection, then you are limited to 2 for that connection (figure 4.4).

Figure 4.4
You are Limited in the Number of Outstanding Messages Allowed



A Quick Reference Guide to the Commands

The following chart provides quick reference material on each of the commands, including syntax rules and examples. Use this guide if you have a thorough understanding of each of the commands. Refer to the individual command sections for detailed information.

For detailed information on:	Refer to the sections located:
OPEN, CLOSE, ABORT, DEFVAR, DELVAR, and SET	later in this chapter
UNINFO, USTAT, and MOVE	in chapter 5

Quick Reference Guide to the 1785–OSI Commands

Command:	Can abbreviate to:	Syntax:	Examples:
OPEN	1 character, upper or lower case	OPEN <connection> TO <AE_name> **Special Note:** AE names can be in either single or double quotes.	open :c16 to 'parts_bin_4' OP :C9 T "PARTS_BIN_14"
CLOSE	1 character, upper or lower case	CLOSE <connection>	CLOSE :c7 clo :c99
ABORT	1 character, upper or lower case	ABORT <connection> **Special Note:** The preferred method for terminating connection is the CLOSE command.	AB :c6 abor :C66
DEFVAR	3 characters, upper or lower case	DEFVAR <connection><remote_symbol> to <remote_address> **Special Note:** MMS named variables must be in single quotes and address strings must be in double quotes.	DEFVAR :C8'WELD_28' TO "T4:16" def :c1'machine_44' to "t9:15"
DELVAR	3 characters, upper or lower case	DELVAR <connection><remote_symbol> **Special Note:** MMS named variables must be in single quotes.	DELVAR :C77'BIN_19' delv :c2'liquid_vat_#22'
SET	1 character, upper or lower case	SET <destination> = <source> **Special Note:** When reading data, the connection identifier directly preceeds the source. When writing data, the connection identifier directly preceeds the destination.	**READING:** set 'station_5' = :c4'parts' SE "T4:3" = :C100'STATION_6' **WRITING:** set :c1"f9:8" = :c'mach_12' S :C77'PLATFORM_6' = "N7:90"
UINFO	2 characters, upper or lower case	UINFO <source> to <connection>	uinfo 'temp_3' to :c35 UI "T4:0.ACC" T :C5
USTAT	2 characters, upper or lower case	USTAT TO <connection>	USTAT TO :C33 ust t :c1
MOVE	1 character, upper or lower case	**READING:** MOVE TO <destination> FROM <connection><source> or MOVE FROM <connection><source> TO <destination> **Special Note:** You can switch the placement of the qualifiers within this command line. As long as you place the <destination> directly after the TO qualifier, and the <connection><source> directly after the FROM qualifier.	**READING:** M T 'TIMER_1' FR :C1'TIMER_4' mov to "n12:1" f :c7'station_99'
		WRITING: MOVE FROM <source> TO <connection><destination> or MOVE TO <connection><destination> FROM <source> **Special Note:** You can switch the placement of the qualifiers within this command line. As long as you place the <connection> <destination> directly after the TO qualifier, and the <source> directly after the FROM qualifier.	**WRITING:** MO FR "N13:90" TO :C13"N16:1" move to :c55'machine_7' fr 'hold'

Managing Connections (The OPEN, CLOSE and ABORT Commands)

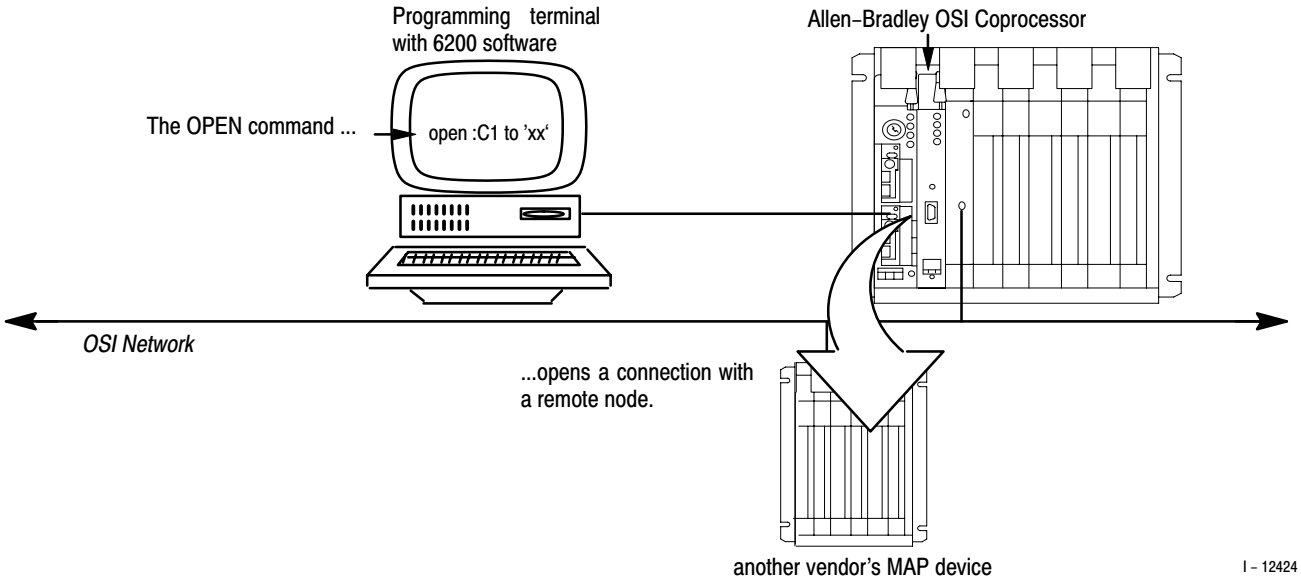
MMS is a connection–oriented set of services. To perform any communication, you must first establish a connection to another device. This section covers the following topics:

- establishing connections with other nodes (OPEN)
- terminating connections with other nodes (CLOSE and ABORT)
- establishing connections to the OSI coprocessor (on connection zero)

Establishing Connections With Other Nodes (OPEN)

The OPEN command establishes a connection between the OSI coprocessor and a remote node on the MAP network (figure 4.5). You **must** establish a connection before attempting any communication or you will receive an error code.

Figure 4.5
Establishing a Connection with the OPEN Command



You can establish up to 16 concurrent connections (outgoing or incoming) from a single OSI coprocessor. You can reserve a particular number of outgoing connections, using the *number of outgoing connection reserved* parameter (see Appendix D, *MMS Parameters*).

Important: You must have an LDIB (Local Directory Information Base) entry for every remote MAP node to which the OSI coprocessor will open a connection. Before establishing a connection, you must add the remote node's addressing information to the LDIB. You edit/add entries to the

LDIB via the Allen–Bradley MAP Station Manager. Refer to the MAP Station Manager User's Manual for more information (publication 6630–6.5.2).

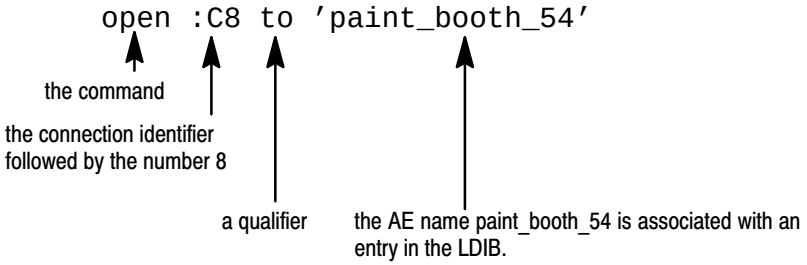
The OPEN command has the following syntax:

```
OPEN <connection> TO <application entity name>
```

The following table defines each part of the OPEN command line.

This:	is:
OPEN	the command that initiates the connection. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999 (you do not have to use them in any particular order).
TO	a qualifier. You can abbreviate as T and use upper or lower case letters.
<application entity name>	the application entity (AE) name of the remote node with which you are establishing a connection. You must match the AE name to an entry in the OSI coprocessor LDIB. The AE name must start with either single or double quotes (and up to 64 characters long). You set up AE names via the A–B MAP Station Manager (see the station manager's user manual, publication 6630–6.5.2, for details).

For example:



Here are more examples of using the OPEN command:

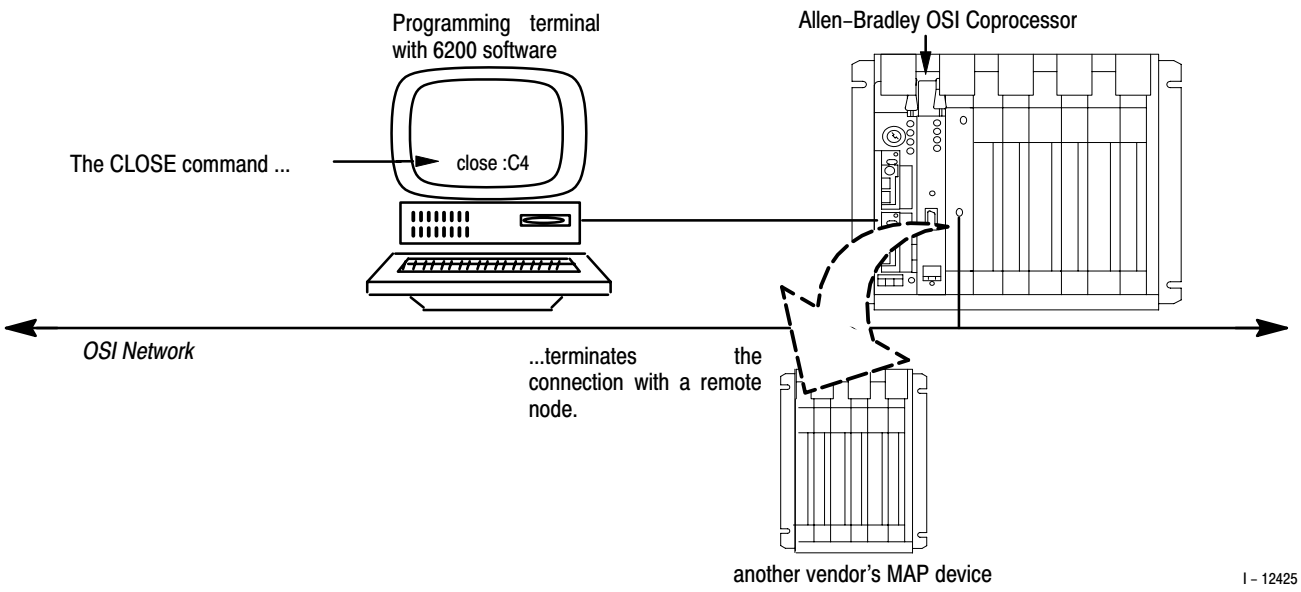
```
OPEN :C16 TO 'STATION_33'

OP :C999 t 'wheel_10'
```

Terminating Connections With Other Nodes (CLOSE and ABORT)

The **CLOSE command** gracefully terminates the connection with a remote node on your MAP network. You can close only the connections you established from your local PLC–5 controller (figure 4.6).

Figure 4.6
Terminating a Connection with the CLOSE Command



The CLOSE command has the following syntax:

```
CLOSE <connection>
```

The following table defines each part of the CLOSE command line.

This:	is:
CLOSE	the command that gracefully terminates the connection. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999).

For example:

```
CLOSE :C78
```

the command

the connection identifier followed by the number 78

Here are more examples of using the CLOSE command:

```
close :c4
```

```
c :c1
```

The **ABORT command** *abruptly* terminates the connection between the OSI coprocessor and a remote node on the MAP network. You can abort only the connections that you established from your local PLC–5 controller. If there are problems with a connection, you would use the abort command.

Important: The preferred method for simply terminating connections is the CLOSE command. See the preceding section for details.

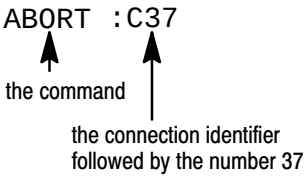
The ABORT command has the following syntax:

```
ABORT <connection>
```

The following table defines each part of the ABORT command line.

This:	is:
ABORT	the command that abruptly terminates the connection. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999.

For example:



Here are more examples of using the ABORT command:

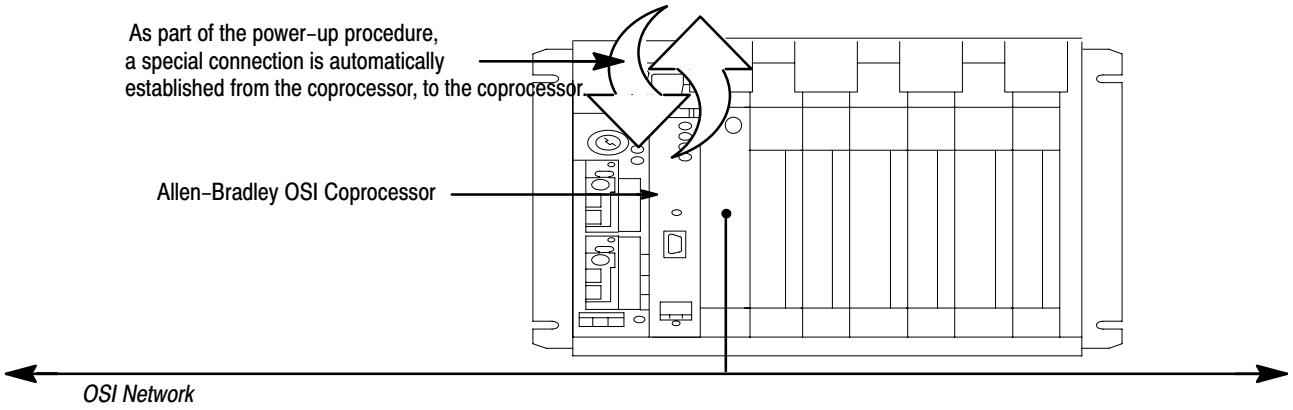
```
AB :C199
```

```
abort :c66
```

Connection Zero

There is a special connection that is opened automatically during the coprocessor's power–up procedure (figure 4.7). This is **connection zero**, denoted as **:C0**.

Figure 4.7
Establishing a Connection to the OSI Coprocessor with :C0



This connection is always available to the coprocessor operator and allows you to manage your local MMS named variables without having to open or close connections. With connection zero you also avoid using any of the 16 connections available for establishing connections to remote devices. You cannot close or abort connection zero.

You can do the following to your local OSI coprocessor on connection zero:

- define MMS named variables (using the DEFVAR command)
- delete MMS named variables (using the DELVAR command)
- transfer data (using the SET or MOVE command)

Refer to each of the DEFVAR, DELVAR, and SET command sections later in this chapter for special instructions on using connection zero. For information on the MOVE command, refer to Chapter 5.

Defining MMS Named Variables (The DEFVAR Command)

Within MMS, you must define a variable so that application programs can access it. The DEFVAR command defines MMS named variables in a VMD. Refer to the section titled *What You Should Know Before You Program*, earlier in this chapter, for guidelines on using MMS named variables.

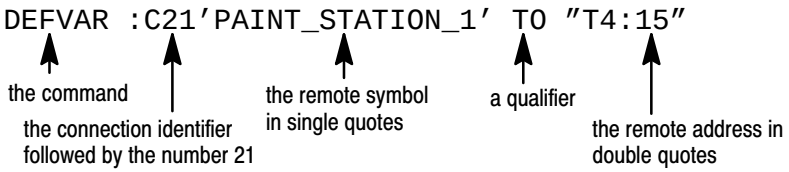
The DEFVAR command has the following syntax:

```
DEFVAR <connection><remote_symbol> TO <remote_address>
```

The following table defines each part of the DEFVAR command line.

This:	Is:
DEFVAR	the command that defines MMS named variables in a VMD. You can abbreviate to three characters, upper or lower case. Leave at least one space between each field in the command line.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999).
<remote_symbol>	the name you are assigning to the remote address. This must be in single quotes and directly follow the connection identifier with no spaces between them.
TO	the qualifier that specifies the location of the address for which you are defining a variable. You can abbreviate as T and use upper or lower case letters.
<remote_address>	the address you are assigning to the name. This must be in double quotes.

For example:



Here are more examples of using the DEFVAR command:

```
def :c66'remote_robot_13' t "T4:22"

DEFV :C5'MACHINE_7' TO "T4:9"
```

You can also define an MMS named variable in your OSI coprocessor that represents and address in your PLC–5 processor. You use the DEFVAR command with connection zero.

For example:

```
DEFVAR :C0'MY_VARIABLE' TO "T4:4,10"
```

See the section titled *Connection Zero,* earlier in this chapter for more information on connection zero.

Deleting MMS Named Variables (The DELVAR Command)

Use the DELVAR command to delete MMS named variables.

The DELVAR command has the following syntax:

```
DELVAR <connection><remote_symbol>
```

The following table defines each part of the DELVAR command line.

This:	Is:
DELVAR	the command that requests the deletion of an MMS named variable. You can abbreviate to three characters and use either upper or lower case letters. Leave at least one space between each field in the command line.
<connection>	the communication link between two applications. This is designated by the connection identifier :C followed by an integer from 1 to 9999. This must directly follow the DELVAR command, followed by the remote symbol.
<remote_symbol>	the MMS variable name you are deleting. This must be in single quotes and directly follow the connection identifier with no spaces separating them.

Here are some examples of using the DELVAR command:

```
DELVAR  :C88'TANK_TEMPERATURE'
```

```
DEL  :C7'station_14'
```

To delete MMS named variable within the coprocessor itself, use connection zero.

For example:

```
DELVAR :C0'MY_VARIABLE'
```

See the section titled *Connection Zero,* earlier in this chapter for more information on connection zero.

Reading and Writing Data (Using the SET Command)

The SET command transfers (reads or writes) an element, a contiguous block of elements, or an MMS structure to or from a node on the MAP network.

Keep in mind that the syntax for the SET command to read is different from using it to write. The placement of the connection identifier (:C) determines whether the command line is a read or a write:

If you are:	The connection identifier directly precedes:
reading data	the source
writing data	the destination

Each case is covered in the following sections.

Using the SET Command to Read Data

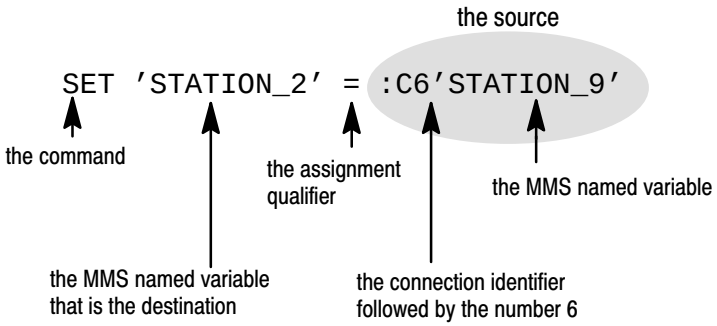
The syntax for using the SET command to **read** is:

```
SET <destination> = <source>
```

The following table defines each part of this SET command line.

This:	Is:
SET	the command that requests the data transfer. The destination directly follows the SET command. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
<destination>	the location where the requested information is going to be placed. This could be an MMS named variable (in single quotes) or an address (in double quotes). The destination always follows the SET command.
=	an assignment qualifier. This always follows the destination and precedes the source.
<source>	the location where the requested information is stored. This could be an MMS named variable (in single quotes) or an address (in double quotes). The location of the source also includes a connection identifier (:C). The source must follow the assignment qualifier (=).

For example:


the source
SET 'STATION_2' = :C6'STATION_9'
the command
the assignment qualifier
the MMS named variable
the MMS named variable that is the destination
the connection identifier followed by the number 6

Here are more examples of using SET to read:

set ’vat_#18’ = :c3’fluids’

SE ”t4:43” = :C78’timer_44’

SET ’plank_50’ = :c6”n7:99”

Using the SET Command to Write Data

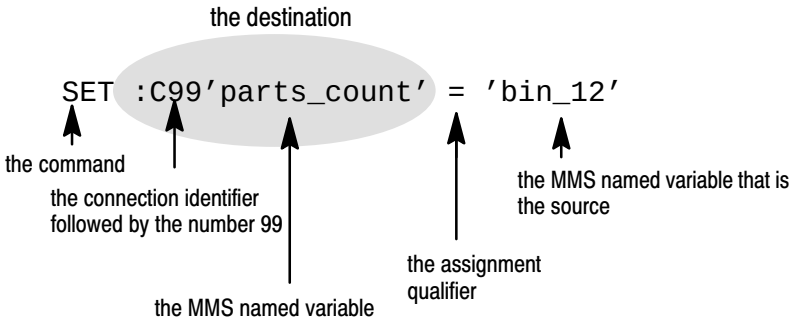
The syntax for using the SET command to **write** is:

```
SET <destination> = <source>
```

The following table defines each part of this SET command line.

This:	is:
SET	the command that requests the data transfer. The destination directly follows the SET command. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
<destination>	the location where the requested information is going to be placed. This could be an MMS named variable (in single quotes) or an address (in double quotes). The location of the destination also includes a connection identifier (:C). The destination always follows the SET command.
=	an assignment qualifier. This always follows the destination and precedes the source.
<source>	the location where the requested information is stored. This could be an MMS named variable (in single quotes) or an address (in double quotes). The source must follow the assignment qualifier (=).

For example:



Here are more examples of using SET to write:

set :c8’vat_#18’ = ’fluids’

SE :C78”t4:43” = ’timer_44’

SET :c6’plank_50’ = ”n7:99”

Using SET with Connection Zero
You can also transfer data within your local PLC–5 controller using the SET command. You use the command in conjunction with connection zero (:C0). The rules apply the same way for reading and writing on connection zero as discussed in the two previous sections. For example:

```
SET :C0’MY_VARIABLE’ = ”B7:6”
```

As noted in the preceding section, the placement of the connection identifier determines whether you are **reading** or **writing.** On connection zero, however, reading and writing data are essentially the same, because you are reading/writing to/from the local PLC–5 controller. With that in mind, we could have written the example above as:

```
SET ’MY_VARIABLE’ =  :C0”B7:6”
```

Chapter 5

Additional and Advanced Programming Techniques

Chapter Objectives

This chapter contains additional programming techniques. In it, we cover the following topics:

- sending unsolicited variable information (the UINFO command)
- sending unsolicited status information (the USTAT command)
- reading and writing data (using the MOVE command)
- obtaining status on a connection (using CSTAT)
- specifying the data type of an MMS named variable (using DTYPE)
- defining scopes of MMS named variables (using DEFVAR)
- deleting MMS named variables of a particular scope (using DELVAR)
- specifying the scope of an MMS named variable within the SET and MOVE command lines
- specifying the scope of an MMS named variable within the UINFO command line

Note that we consider many of these topics as advanced programming techniques. We assume you have thorough knowledge of the MMS protocol. Refer to Chapter 4 of this manual for basic programming information.

Sending Unsolicited Variable Information (The UINFO Command)

You can send unsolicited variable status using the UINFO command. This command has the following syntax:

```
UINFO <source> TO <connection>
```

The following table defines each part of the UINFO command line.

Where:	Is:
UINFO	the command that specifies sending an unsolicited report to a connection. You can abbreviate to two characters and use upper or lower case letters. Leave at least one space in between each field in the command line.
<source>	the location where the requested information is stored. This must be a local MMS named variable or local address, followed by the TO qualifier.
TO	the qualifier that specifies where the information is being sent; you can abbreviated as T.
<connection>	the communication link established between one application and another. This will be the connection identifier :C followed by an integer from 1 to 9999.

Here are examples of using the UINFO command:

```
UINFO  'TEMPERATURE_4'  TO  :C25

UI  "T4:0.ACC"  TO  :C601
```

The related MMS service for UINFO is InformationReport.

Sending Unsolicited Status Information (The USTAT Command)

You can send unsolicited status information using the USTAT command (this will be the same status as in the MMS Status Response in server mode). This command has the following syntax:

```
USTAT  TO  <connection>
```

The following table defines each part of the USTAT command line.

Where:	Is:
USTAT	the command that specifies sending status to a connection. You can abbreviate to two characters and use upper or lower case letters. Leave at least one space between each field in the command line.
TO	the qualifier that specifies where the status will be sent; you can abbreviate as T. You must specify the TO qualifier, and it must directly follow the USTAT command.
<connection>	the communication link established between one application and another. This will be the connection identifier :C followed by an integer from 1 to 9999.

Here are examples of using the USTAT command:

```
USTAT  TO  :C33

US   TO  :C9
```

The related MMS service for USTAT is UnsolicitedStatus.

Reading and Writing Data (Using the MOVE Command)

Like the SET command, the MOVE command transfers (reads or writes) an element, a contiguous block of elements, or an MMS structure to or from a node on the MAP network. With MOVE, however, you also need to use the TO and FROM qualifiers. We recommend using SET, simply because it is shorter to type in (see Chapter 4 for information on the SET command).

The syntax for using the MOVE command to read is different from using it to write. The following sections address each case individually in the following order:

- using the MOVE command to read data
- using the MOVE command to write data
- using the MOVE command read/write data from/to your local PLC–5 controller (on :C0)

The related MMS service for MOVE is Read/Write.

Using the MOVE Command to Read Data

The syntax for using MOVE to read data is:

```
MOVE TO <destination> FROM <connection><source>
```

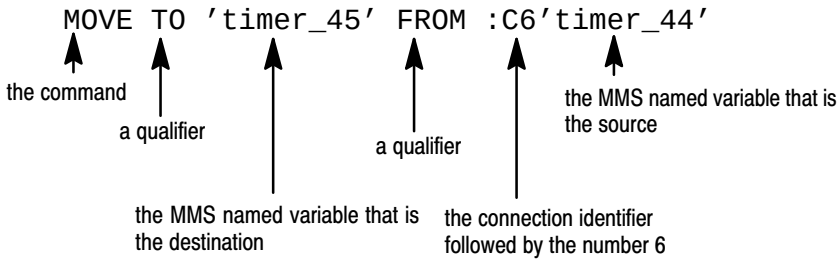
You can also switch the placement of the qualifiers within the command line:

```
MOVE FROM <connection><source> TO <destination>
```

The following table defines each part of the MOVE command line.

This:	Is:
MOVE	the command that requests the data transfer. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
TO	the qualifier that allows you specify the destination of the data transfer. You can abbreviate as T and use upper or lower case letters.
<destination>	the location where the requested information is going to be placed. This could be a local MMS named variable (in single quotes) or a local address (in double quotes).
FROM	the qualifier that allows you to specify the source of the data transfer. You can abbreviate to one character and use upper or lower case letters.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999).
<source>	the location where the requested information is stored. This could be a remote MMS named variable (in single quotes) or a remote address (in double quotes). A connection identifier (:C) must precede the source, and both must directly follow the FROM qualifier.

For example:



The important thing to remember about the syntax for using the MOVE command to read is the placement of the source and destination. You **must** place the <destination> directly after the TO qualifier, and the <connection><source> directly after the FROM qualifier.

Here are more examples of using the MOVE command to read:

```
MOVE FROM :C9'PLATFORM_3' TO 'BAY_44'

m t "n12:1" f :c34'station_16'

mov fr C:7"n34:2" to "n12:9"
```

Using the MOVE Command to Write Data

The syntax for using MOVE to write data is:

```
MOVE FROM <source> TO <connection><destination>
```

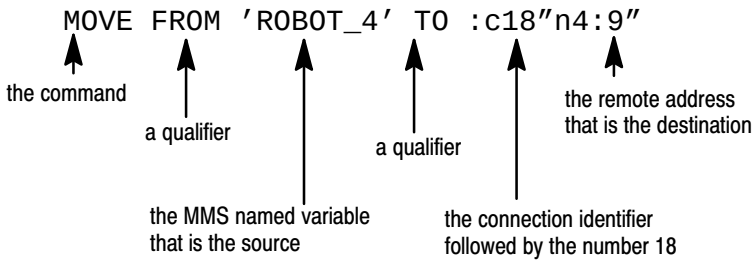
You can also switch the placement of the qualifiers within the command line:

```
MOVE TO <connection><destination> FROM <source>
```

The following table defines each part of the MOVE command line.

This:	Is:
MOVE	the command that requests the data transfer. You can abbreviate to one character, upper or lower case. Leave at least one space between each field in the command line.
FROM	the qualifier that allows you to specify the source of the data transfer. You can abbreviate to one character and use upper or lower case letters.
<source>	the location where the requested information is stored. This could be a local MMS named variable (in single quotes) or a local address (in double quotes).
TO	the qualifier that allows you specify the destination of the data transfer. You can abbreviate as T and use upper or lower case letters.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 0 to 9999.
<destination>	the location where the requested information is going to be placed. This could be a remote MMS named variable (in single quotes) or a remote address (in double quotes). A connection identifier (:C) must precede the destination, and both must directly follow the FROM qualifier.

For example:



The important thing to remember about the syntax for using the MOVE command to write is the placement of the source and destination. You **must** place the <connection><destination> directly following the TO qualifier, and the <source> directly following the FROM qualifier.

Here are more examples of using the MOVE command to write:

```
MO FR "N15:90" TO :C13"N16:1"

move to :c5'machine_19' fr 'hold_44'

M T :C78"N67:10" FROM 'WELDER_10'
```

Using MOVE to Transfer Data Within Your Local PLC–5 Controller

You can also transfer data within you local PLC–5 controller using the MOVE command. You use the command in conjunction with connection zero (:C0).

The MOVE command to **read** on connection zero has the following syntax:

```
MOVE FROM :C0<source> TO <destination>
```

The MOVE command to **write** on connection zero has the following syntax:

```
MOVE FROM <source> TO :C0<destination>
```

(You can also switch the placement of the qualifiers within both of the command lines).

For example:

```
MOVE FR :C0'MY_VARIABLE' TO "B7:6"
```

As noted in the previous section, the placement of :C0 determines weather you are **reading** or **writing,** but on connection zero reading and writing data are essentially the same. You are reading/writing to/from the local OSI coprocessor. With that in mind, we could have used the example above as:

```
MOVE FR 'MY_VARIABLE' TO :C0"B7:6"
```

Obtaining Status on a Connection (using CSTAT)

If you use the optional CSTAT parameter when you open a connection, you can later determine the status of that connection. You do this by checking the value of a bit in the database (1 = open, 0 = closed).

The syntax for using CSTAT is:

```
OPEN <connection> to <AE name> CSTAT 'local_bit_address'
```

the optional parameter that allows you to determine if the connection is open (1) or closed (0).

The local address; this can be an MMS named variable (in single quotes) or an address (in double quotes).

MUST be a single bit (boolean), refer to Appendix A for more information on data types.

See Chapter 4 for details on the OPEN command.

How does CSTAT work?
When you use CSTAT in an OPEN command line, the OSI coprocessor uses the specified bit in the PLC data table to report status for that connection. When the coprocessor sends the initiate request to the remote node, it sets the specified bit to zero:

If the connection:	Then:
is successfully made	the OSI coprocessor sets the bit to 1
is not made or is unsuccessful	the bit remains 0
is broken (by either the local or remote device)	the OSI coprocessor sets the bit to 0

You can query this bit in your rung ladder logic to determine the current state of the connection.

Important: The OSI coprocessor writes to this bit only when there is a change in the state of the connection. You must determine that nothing else in the system is modifying the bit.

Specifying the Data Type of an MMS Named Variable (using DTYPE)

When you use the DEFVAR command to define an MMS named variable, you are also specifying the data type for which that variable will be used (see Chapter 4 for more information on DEFVAR). You specify the data type in one of two ways:

- using the DTYPE parameter with the DEFVAR command line to specify the data type.
- by simply defining your MMS variable using the DEFVAR command and not using the DTYPE parameter, in which case the **default** data type is assumed.
- specifying the data type of an MMS named variable

The syntax for using the DTYPE parameter with the DEFVAR command is:

```
DEFVAR <connection><remote_symbol> TO
 <remote_address> DTYPE <local_address>
```

The following table defines each part of this command line:

This:	Is:
DEFVAR	the command that defines MMS named variables in a VMD. You can abbreviate to three characters, upper or lower case. Leave at least one space between each field in the command line.
<connection>	the communication link established between two points. You label connections using the connection symbol :C followed by an integer from 1 to 9999).
<remote_symbol>	the name you are assigning to the remote address. This must be in single quotes and directly follow the connection identifier with no spaces between them.
TO	the qualifier that specifies the location of the address for which you are defining a variable. You can abbreviate as T and use upper or lower case letters.
<remote_address>	the address you are assigning to the name. This must be in double quotes.
DTYPE	a parameter you place within the DEFVAR command line when you want to specify the data type for which the variable will be used. Must be followed by the local address. You can abbreviate to one character, upper or lower case.
<local_address>	the type of address the MMS named variable will be associated with. The data type of the data at this address is the data type of the MMS named variable. This directly follows the DTYPE parameter and is in double quotes.

Directly after the DTYPE parameter, you add a local address. The data type of the data at the local address will be the data type of the MMS named variable.

If you **do not** specify the data type, the default data type for the remote address that you have entered is assumed. For example, the following shows how to define the variable **painter_6** for the BCD (binary coded decimal) address type **D11:7,2**:

```
DEFVAR :C1'painter_6' to "D11:7,2"
```

The MMS named variable painter_6 will be associated with the BCD address D11:7,2. Because we did not specify a data type here, the default data type associated with the remote address D11:7,2 is assumed. In our example we used a **BCD** address, which has a **BCD** default data type. When information is written to or read from that address in the future, the information will be in the BCD data type. We could have, however, defined a different data type for that information.

If we had wanted the information at the remote BCD address D11:7,2 to be in **integer**, we could use the DTYPE parameter to define it as such. For example:

```
DEFVAR :C1'painter_6' to "D11:7,2" dt "N7:0,2"
```

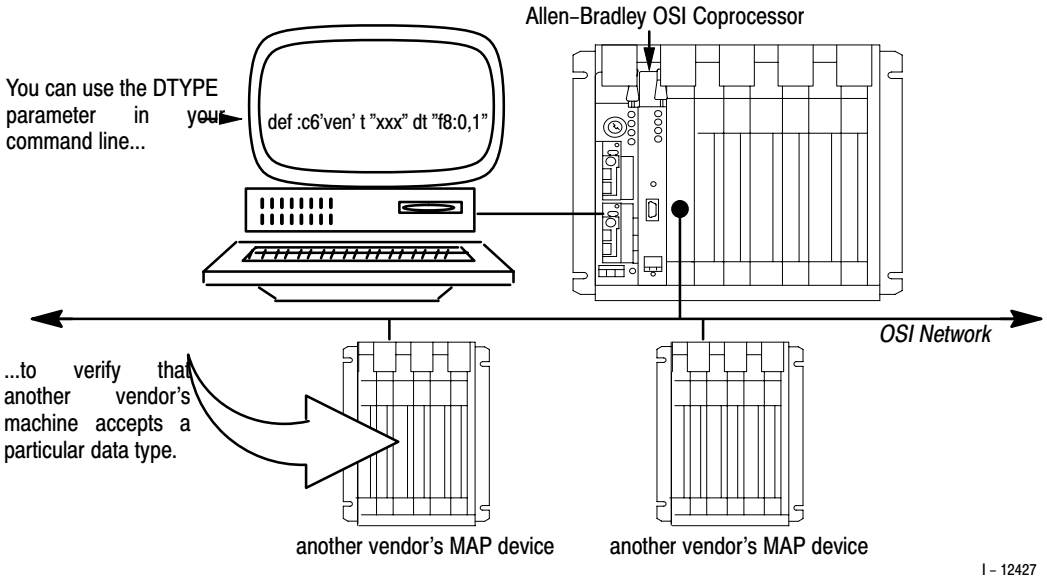
Notice that in the command line, we used the DTYPE parameter (abbreviated as dt), directly followed by an integer address. The MMS protocol knows that we are requesting the data type for information going into and out of the address D11:7,2 be in integer form.

It is the type of data at the address you enter directly after the DTYPE parameter that designates the data type. This address can be any valid PLC–5 address string, but need not exist.

Why would you specify a data type?
If the remote address for which you are defining an MMS named variable is of a type that is acceptable to your applications, you do not need to worry about specifying a data type. However, if you know that you want the information that goes to and from the remote address to be of a certain data type that is different than that of the remote address, you can specify that data type at the time you define the variable.

Another reason you may want to use the data type parameter is if you are defining a variable in another vendor's machine and you want to make sure it will accept the data type you want to use. You specify the data type in the DEFVAR command line and verify that the machine accepts that data type (figure 5.1).

Figure 5.1
Verifying Another Machine Accepts the Data Type You Want to Use



For example:

```
def :c6'other_vendor_var' t "xxxx" dt "f8:0,12"
```

In the example above, we are defining the variable as address "xxxx". With the added DTYPE parameter, we are specifying that we want an **array** of 12 floating points to be used as the data type. If the data type is **not** acceptable, we will receive an error message right away. If we had not used DTYPE here, we would have had to use the MMS named variable within a MOVE command line before knowing the data type was unacceptable.

Defining the Scope of an MMS Named Variable (using DEFVAR)

When you define an MMS named variable using the DEFVAR command (see chapter 4), you also define the scope of that variable. A variable can have one of three scope types:

- VMD–specific
- application association (AA)
- domain

Important: Domain variable scope is only supported by the OSI coprocessor as a **client,** not a **server.** Therefore, you can specify all three domain types in client–type applications, but only VMD and AA scope will be accepted by server applications. For example, you can define MMS named variables with VMD–specific and AA scope in the OSI coprocessor, but **not** domain scope. You can, however, define MMS

named variables with domain scope for other vendor's devices (if supported by those devices).

Defining VMD–specific Scope

VMD–specific will probably be the most widely used scope type for MMS named variables in your application and is probably the only one you need to use. You can specify VMD–specific scope in two ways:

- by placing /vmd/ just before the MMS named variable within the single quotes in the DEFVAR command line.
- NOT specifying any scope type. If no scope is specified, VMD scope is assumed.

The following example shows how to specify VMD–specific scope:

```
DEFVAR :C8'/vmd/BOX_6' to "F24:7"
```

The MMS named variable is BOX_6 and is in single quotes. Here, we are specifying VMD–specific scope, with VMD within slash marks (/) and within the single quotes.

Because VMD–specific scope type, we could have accomplished the same result if we had written the above command line as:

```
DEFVAR :C8'BOX_6' to "F24:7"
```

Since we did not specify a scope type, MMS assumes VMD–specific scope.

Defining Application Association (AA) Scope

You can define MMS named variable with AA scope. MMS named variables with application association scope are variables that are defined during the period of time that a particular connection is in use. These variables are automatically deleted when the connection is closed.

Important: When you define MMS named variables with application association scope, you should keep a record of them to ensure you specify their scopes correctly in future applications. After defining the variables' scopes, you must specify scopes correctly within command lines or you will receive an error code.

You specify AA scope by placing /AA/ before the MMS named variable and within the single quotes. You can enter AA in upper or lower case letters, and abbreviate to one character.

The following example shows how to specify AA–specific scope:

```
DEFVAR :C78'/AA/TIMER' TO "N7:0"
```

The MMS named variable is TIMER and is in single quotes. By placing /AA/ just before the variable, we are specifying application association scope.

Defining Domain Scope

You can define MMS named variables with domain scope. You should keep a record of them to ensure you specify their scopes correctly in future applications. After defining the variables' scopes, you must specify the scopes correctly within command lines or you will receive an error code.

Important: Domain variable scope is only supported by the OSI coprocessor as a **client,** not a **server.**

You specify domain scope by placing /domain, followed by the domain name, and another /. All of this precedes the MMS named variable and is within the single quotes.

The following example shows how to specify domain–specific scope.

```
DEFVAR :C1'/DOMAIN:PAINT/LEVEL' TO "abc123"
```

The MMS named variable is LEVEL, DOMAIN is the scope, and PAINT is the domain name, and all are within single quotes.

Variable address in another vendor's MAP device.

In the example above, DOMAIN is the scope and PAINT is the domain name. A colon must separate the domain name and the word DOMAIN. Note that we show the sample address "abc123" to designate the address of another vendor's MAP device, because specifying domain scope is only supported by your OSI coprocessor as a client, not as a server.

Rules for Specifying Domain–specific Scopes
When specifying domain scope, keep the following rules in mind:

- you can abbreviate DOMAIN to one or more characters, upper or lower case
- a colon must separate DOMAIN (or its abbreviation) and the domain name, with **no spaces** between them
- you **must** specify the domain name
- the domain name can be up to 32 characters long and can contain the following characters: A to Z (either upper or lower case), numerals 0 through 9, an underscore (_), and a dollar sign ($). It cannot start with a numeral.

Deleting MMS Named Variable of a Particular Scope (Using DELVAR)

Within the DELVAR command line, you have the opportunity to specify the scope of the variable you are deleting. If you do not specify the scope of the MMS named variable you are deleting, it is assumed that variable has VMD–specific scope.

Important: Domain variable scope is only supported by the OSI coprocessor as a **client,** not a **server.** Therefore, you can specify all three domain types in client–type applications, but only VMD and AA scope will be accepted by server applications. For example, you can define/delete MMS named variables with VMD–specific and AA scope in the OSI coprocessor, but **not** domain scope. You can, however, define/delete MMS named variables with domain scope for other vendors' devices (if supported by those devices).

Important: If you have defined an MMS named variable with either application association (AA) or domain scope, you **must** specify its scope within the DELVAR command line (see previous section). By not entering a scope, it is assumed the variable has VMD–specific scope.

If the variable you are deleting has, for example, application association scope (AA), you must specify it within the command line, for example:

```
DELVAR  :C9'/AA/COUNTER_6'
```

If we had **not** specified **AA** scope for the variable COUNTER_6, then it would be assumed that it had VMD–specific scope. If there was no variable called COUNTER_6 with VMD–specific scope, no variable would be deleted. If there was another variable called COUNTER_6 that **had** VMD–specific scope, that variable would have been deleted.

The following is an example of deleting a variable with **domain** scope:

```
DELVAR :C1'/DOMAIN:PAINT/LEVEL'
```

The same rules apply for deleting domain–scope variable that apply for defining them. For the rules, see the section titled *Rules for Specifying Domain–specific Scopes,* earlier in this chapter.

Specifying MMS Named Variable Scope Within SET and MOVE

Within the SET and the MOVE command lines, you have the opportunity to specify the scope of an MMS named variable. If you do not specify the scope of the MMS named variable, it is assumed that variable has VMD–specific scope.

You can specify each of the scope types:

- VMD–specific
- application association (AA)
- domain

The same rules for specifying scopes within the SET command line apply to specifying them with the MOVE command line. The following sections provide examples of the SET command first. We show examples of the MOVE command after that.

Important: Domain variable scope is only supported by the OSI coprocessor as a **client,** not a **server.** Therefore, you can specify all three domain types in client–type applications, but only VMD and AA scope will be accepted by server applications. For example, you can define (and therefore transfer within SET) MMS named variables with VMD–specific and AA scope in the OSI coprocessor, but **not** domain scope. You can, however, define (and transfer) MMS named variables with domain scope for other vendors' devices (if supported by those devices).

Specifying VMD–specific Scope Within the SET Command Line

You can transfer MMS named variables of VMD–specific scope within the SET command line.

You can specify VMD scope in two ways:

- by placing /vmd/ just before the MMS named variable within the single quotes in the SET command line.
- NOT specifying any scope type. If no scope is specified, VMD scope is assumed.

The following example shows how to specify VMD–specific scope:

```
SET 'PARTS' = :C8'/VMD/BOX_1'
```

The MMS named variable is BOX_1 and is in single quotes. Here, we are specifying VMD–specific scope, with VMD within slash marks (/) and within the single quotes.

If you **do not** specify the scope of an MMS named variable, it automatically defaults to VMD–specific scope. With that in mind, we could have accomplished the same result if we had used the above command line as:

```
SET 'PARTS' = :C8'BOX_1'
```

Since we did not specifically specify a scope type, MMS assumes VMD–specific scope.

If you are using an MMS named variable in a command line that was defined with a particular scope, you need to use that scope correctly. For instance, in the previous example, we did not specify the scope of BOX_1, so it was assumed to have VMD–specific scope. But if BOX_1 had been previously defined with AA scope, we would have received an error code.

Specifying AA Scope Within the SET Command Line

You specify AA scope by placing /AA/ before the MMS named variable and within the single quotes. You can enter AA in upper or lower case letters, and abbreviate to one character.

The following example shows how to specify AA–specific scope:

```
set :C51'/aa/timer' = "N9:0"
```

The MMS named variable is TIMER and is in single quotes. By placing /AA/ just before the variable, we are specifying application association scope.

Specifying Domain Scope Within the SET Command Line

You specify domain scope by placing /domain, followed by the domain name, and another /. All of this precedes the MMS named variable and is within the single quotes.

Rules for Specifying Domain–specific Scopes
When specifying domain scope, keep the following rules in mind:

- you can abbreviate DOMAIN to one or more characters, upper or lower case
- a colon must separate DOMAIN (or its abbreviation) and the domain name, with **no spaces** between them
- you **must** specify the domain name
- the domain name can be up to 32 characters long and can contain the following characters: A to Z (either upper or lower case), numerals 0

through 9, an underscore (_), and a dollar sign ($). It cannot start with a numeral.

The following example shows how to specify domain–specific scope:

```
set :C100'/DOMAIN:PAINT/LEVEL' = '/aa/station_9'
```

The MMS named variable is LEVEL and is in single quotes. DOMAIN is the scope and PAINT is the domain name. A colon must separate the domain name and the word DOMAIN (or an abbreviation).

Specifying Scope Types Within the MOVE Command Line

The same rules for specifying scopes within the SET command line apply to specifying them with the MOVE command line. Refer to the previous section for details. The following sections provide examples of specifying scopes within the MOVE command line.

Specifying VMD–specific Scope Within the MOVE Command Line
This is an example of specifying VMD–specific scope:

```
MOVE TO 'PARTS' FROM :C8'/VMD/BOX_1'
```

The MMS named variable is BOX_1 and is in single quotes. Here, we are specifying VMD–specific scope, with VMD within slash marks (/) and within the single quotes.

If you **do not** specify the scope of an MMS named variable, it automatically defaults to VMD–specific scope. With that in mind, we could have accomplished the same result if we had used the above command line as:

```
MOVE TO 'PARTS' FROM :C8'BOX_1'
```

Since we did not specifically specify a scope type, MMS assumes VMD–specific scope.

If you are using an MMS named variable in a command line that was defined with a particular scope, you need to use that scope correctly. For instance, in the previous example, we did not specify the scope of BOX_1, so it was assumed to have VMD–specific scope. But if BOX_1 had been previously defined with AA scope, we would have received an error code.

Specifying AA–specific Scope Within the MOVE Command Line
This is an example of specifying AA–specific scope:

```
move from :C51'/aa/timer' t "N9:0"
```

The MMS named variable is TIMER and is in single quotes. By placing /AA/ just before the variable, we are specifying application association scope.

Specifying Domain–specific Scope Within the MOVE Command Line
This is an example of specifying domain–specific scope:

```
mo to :C100'/DOMAIN:PAINT/LEVEL' fr '/aa/station_9'
```

The MMS named variable is LEVEL and is in single quotes. DOMAIN is the scope and PAINT is the domain name. A colon must separate the domain name and the word DOMAIN (or an abbreviation).

Here are examples for specifying MMS named variables within the MOVE command line:

```
move from :c11'robot_4' to '/aa/robot_17'

m t 'machine_67' f :c8'/dom:P5/transfer_4'

move f :C90'/vmd/press_19' to '/aa/press_1'
```

When you are defining MMS named variables and their scopes, it is a good idea to keep a record of the variables and their associated scope types. If you specify the wrong domain for a given MMS named variable, you will receive an error code.

Specifying MMS Named Variable Scope Within UINFO

You can specify the scope of the variable within the UINFO command line. To do this, enter the scope within the slash marks (/) and within the MMS named variable's single quotes. For example:

```
UINFO '/AA/MACHINE_44' TO :C7
```

The same rules apply for specifying scopes within the DEFVAR, DELVAR and SET commands as they do for specifying scopes in the UINFO command. Refer to the previous sections of this chapter for more details.

Appendix **A**

Mapping MMS Data Types onto PLC–5/40, –5/60 File Types

Appendix Contents

The following table contains the MMS data types and PLC–5/40, –5/60 file types for you to use as a reference when mapping the two types during your application programming. Refer to Chapter 2 for format examples of the PLC Data Table File Types mapped as ‘‘MMS Structures” (as far as MMS is concerned) and additional information related to data types.

In the following table, we use these symbols:

- the letter **D** to indicate the ‘‘default” data type for the respective PLC–5/40, –5/60 file type. The default data type is the type that is returned in the response to a *GetVariableAccessAttributes* service request on a variable in that field.
- **asterisks (*)** to indicate alternate data types supported for each PLC–5/40, –5/60 file type
- **numbers in parenthesis** to indicate the default size

Any file that has a supported type of BitString, also has a supported type of Boolean when accessed by a single bit address. Note that MMS arrays can also be used/obtained when accessing the PLC controller memory.

PLC–5 Data Table File Types:	MMS Data Types								
	Boolean:	Bit String:	Integer:	BCD:	Unsigned Integer:	Floating Point:	Visible String:	Octet String:	MMS Structure:
Binary (B)	D	D							
Output (O)	D	D							
Input (I)	D	D							
Status (S)	*	*	D(16)		*				
Signed Word (N)	*	*	D(16)		*				
Float (F)						D			
BCD (D)			*	D	*				
ASCII (A)							D	*	
String (ST)							D(–82)	*(–82)	
Timer (T)									D
T.DN	D								
T.TT	D								
T.EN	D								
T.PRE			D(16)		*				

T.ACC			D(16)		*				

PLC–5 Data Table File Types:	MMS Data Types								
	Boolean:	Bit String:	Integer:	BCD:	Unsigned Integer:	Floating Point:	Visible String:	Octet String:	MMS Structure:
Counter (C)									D
C.UN	D								
C.OV	D								
C.DN	D								
C.CD	D								
C.CU	D								
C.PRE			D(16)		*				
C.ACC			D(16)		*				
Control (R)									D
R.FD	D								
R.IN	D								
R.UL	D								
R.ER	D								
R.EM	D								
R.DN	D								
R.DU	D								
R.EN	D								
R.LEN			D(16)		*				
R.POS			D(16)		*				
PID (PD)									D
PD.PE									
PD.MO	D								
PD.CA	D								
PD.SWM	D								
PD.DO	D								
PD.PVT	D								
PD.CL	D								
PD.CT	D								
PD.EN	D								
PD.PVHA	D								
PD.PVLA	D								
PD.DVPA	D								
PD.DVNA	D								
PD.EWD	D								
PD.OLH	D								

PLC–5 Data Table File Types:	MMS Data Types								
	Boolean:	Bit String:	Integer:	BCD:	Unsigned Integer:	Floating Point:	Visible String:	Octet String:	MMS Structure:
PD.OLL	D								
PD.SPOR	D								
PD.INI	D								
PD.SP						D			
PD.KP						D			
PD.KI						D			
PD.KD						D			
PD.BIAS						D			
PD.MAXS						D			
PD.MINS						D			
PD.DB						D			
PD.SO						D			
PD.MAXO						D			
PD.MINO						D			
PD.UPD						D			
PD.PV						D			
PD.ERR						D			
PD.OUT						D			
PD.PVH						D			
PD.PVL						D			
PD.DVP						D			
PD.DVN						D			
PD.PVDB						D			
PD.DVDB						D			
PD.MAXI						D			
PD.MINI						D			
PD.TIE						D			
PD.ADDR								D(8)	
PD.DATA								D(56)	
Message (MG)									D
MG.ER	D								
MG.DN	D								
MG.EW	D								
MG.CO	D								
MG.ST	D								
MG.EN	D								
MG.TO	D								

PLC–5 Data Table File Types:	MMS Data Types								
	Boolean:	Bit String:	Integer:	BCD:	Unsigned Integer:	Floating Point:	Visible String:	Octet String:	MMS Structure:
MG.NR	D								
MG.ERR			D(16)		*				
MG.RLEN			D(16)		*				
MG.DLEN			D(16)		*				
MG.DATA								D(104)	
Block Transfer Control (BT)									D
BT.RW	D								
BT.TO	D								
BT.NR	D								
BT.EW	D								
BT.CO	D								
BT.ER	D								
BT.DN	D								
BT.ST	D								
BT.EN	D								
BT.RLEN			D(16)		*				
BT.DLEN			D(16)		*				
BT.FILE			D(16)		*				
BT.ELEM			D(16)		*				
BT.RGS			D(16)		*				
SFC Status (SC)									D
SC.DN	D								
SC.ER	D								
SC.OV	D								
SC.LS	D								
SC.FS	D								
SC.SA	D								
SC.PRE			D(16)		*				
SC.TIM			D(16)		*				
Token Data (TD)									D
TD.HI			D(16)		*				
TD.LO			D(16)		*				

Appendix B

Error Codes

Appendix Contents

This appendix contains a list of error codes that can be returned to a MSG instruction given to the OSI interface for execution.

Decimal:	Hex:	Description:
257	0101	The requested command is inappropriate for the state of the connection.
259	0103	CLOSE (conclude) not supported by remote device.
260	0104	Invalid or duplicate invoke ID – indicates network performance problem.
263	0107	Service not supported by remote device.
264	0108	Parameter invalid or not supported by remote device, or error in the MMS named variable specification.
265	0109	The number of service requests outstanding exceeds the negotiated maximum.
266	010a	Outgoing PDU is larger than negotiated maximum.
268	010c	Data value out of range at local node – check validity of data in data table.
269	010d	Insufficient memory resources at local node to complete service.
271	010f	Connection ID already in use.
272	0110	Local service request timeout – connection aborted.
273	0111	Cannot CLOSE this connection – Request(s) outstanding.
274	0112	Connection could not be made.
275	0113	Requested service not available on connection 0 or unable to delete named variable on connection 0.
276	0114	No connection available – all in use.
277	0115	Connection unknown to local node or is lost.
278	0116	Communications temporarily disabled from local node.
280	0118	Parameter support not negotiated – see ISO 9506 Parameter CBB.
295	0127	Access to local node temporarily unavailable.
298	012a	UINFO – named object non–existent on local node.
302	012e	UINFO – Cannot access unnamed variable on local node.
306	0132	Defined variable has an undefined reference attribute. This is a permanent error.
307	0133	An attempt to access the variable on the local node has failed due to a hardware fault.
308	0134	The requested variable is temporarily unavailable on the local node.
309	0135	The local MMS client has insufficient privilege to request this operation.
310	0136	The variable does not exist on the local node.
311	0137	The local unnamed variable is invalid because the format of the address is out of range.
312	0138	An inappropriate or unsupported type is specified for a local variable.
313	0139	The data type of the local variable is inconsistent with the data type of the remote variable or with the service requested.

314	013a	The local variable is specified with inconsistent attributes.
315	013b	The local variable is not defined to allow the requested access.
316	013c	The local unnamed variable does not exist.
Decimal:	**Hex:**	**Description:**
317	013d	The local named variable does not exist.
318	013e	The local named variable has invalid address reference.
319	013f	Variable access temporarily disabled on the local node.
327	0147	Invalid number of parameters in MSG string.
328	0148	Command keyword in MSG string not found or invalid.
329	0149	MSG string syntax error – invalid or misspelled parameter, or unmatched or missing quotes.
330	014a	Parameter position invalid, parameter missing, or duplicate parameter in MSG string.
331	014b	Missing or misspelled TO or FROM in MSG string.
332	014c	Invalid connection identifier in MSG string.
334	014e	Invalid Application Entity Name in MSG string.
335	014f	Invalid Application Entity Name length in MSG string.
336	0150	Named variable in MSG string has invalid length.
337	0151	Local variable specification missing from MSG string.
338	0152	Remote variable specification missing from MSG string.
339	0153	Missing or misplaced connection identifier in MSG string.
340	0154	Misplaced DTYPE keyword in MSG string.
341	0155	Missing or misplaced type specification in MSG string.
342	0156	Keyword in MSG string has too few characters.
343	0157	MSG string syntax error – type specification invalid.
344	0158	MSG string syntax error – missing / or : delimiter in scope specification.
345	0159	Invalid or misspelled scope keyword in MSG string.
346	015A	The scope keyword length in MSG string is invalid.
347	015B	The ASCII MSG command string too long.
348	015c	Missing or misplaced equal (=) sign.
349	014d	Misplaced CSTAT parameter.
350	015e	CSTAT bit address missing.
351	015f	CSTAT named variable must have VMD scope.
356	0164	The remote node cannot execute the requested service in its present state.
357	0165	The requested service would change the state of the remote node, in conflict with the current state of the remote node.
358	0166	The requested service cannot be executed due to an operational problem in the remote node.
359	0167	The load data transmitted is inconsistent at the remote node and cannot be used.
360	0168	There is no state machine at the remote node associated with the state machine ID.
366	016e	General application reference problem at remote node.
367	016f	The application referenced is currently unreachable at remote node.
368	0170	The connection to the specified application at the remote node was lost before the service could be completed.

369	0171	The specified application reference is invalid at the remote node.
370	0172	The specified application does not support the desired application context at the remote node.
376	0178	General object definition problem at remote node.
Decimal:	**Hex:**	**Description:**
377	0179	The object does not exist at the remote node.
378	017a	The address format is incorrect or address is out of range at remote node. Applies only to unnamed variables when the CBB VADR parameter is selected.
379	017b	The type specified for the variable is inappropriate or unsupported at the remote node.
380	017c	The remote node determined that the specified type is inconsistent with the service or referenced variable.
381	017d	The defined object already exists at the remote node.
382	017e	The remote node is specified with inconsistent attributes.
386	0182	The resources requested by this service are not available for assignment at the remote node.
387	0183	The memory requested by this service is not available at the remote node.
388	0184	The CPU resources to support maintenance of states are not available at remote node.
389	0185	The storage for additional file data at remote node is lost.
390	0186	One or more capabilities are insufficient at remote node.
391	0187	One or more capabilities are unknown at remote node.
396	018c	Problem with service primitives at remote node.
397	018d	The remote node determined that the sequence of service primitives is invalid.
398	018e	Current state of the remote object does not permit a response for this service from the remote node.
399	018f	PDU size is larger than the negotiated size at the remote node.
400	0190	The remote node determined that the file name to continue after cannot be a member of the group of files indicated in the file specification.
401	0191	Current constraints on an object prevent the execution of this service at the remote node.
406	0196	Service was preempted due to a cancel or other unspecified problem at the remote node.
407	0197	Service was cancelled due to user defined timeout at the remote node.
408	0198	Service was cancelled by the remote node in order to prevent a deadlock.
409	0199	Service was cancelled by the remote node.
416	01a0	General time–resolution problem at remote node.
417	01a1	Requested time resolution is not supportable at remote node.
426	01aa	The remote node determined that service to an object was incorrectly specified.
427	01ab	The variable is not defined to allow the requested access at the remote node.
428	01ac	The variable does not exist at the remote node.
429	01ad	The remote MMS client has insufficient privilege to request this operation.
430	01ae	The variable at the remote node has an undefined reference attribute. This is a permanent error for access to this variable.
436	01b4	General problem with initiate service at remote node.
437	01b5	The common version number does not exist at remote node for the desired communication.
438	01b6	Maximum PDU size proposed to the remote node is too small for the desired communication.
439	01b7	Max Services Outstanding Calling parameter proposed to the remote node is too small for the desired communication.

440	01b8	Max Services Outstanding Called parameter proposed to the remote node is too small for the desired communication.
Decimal:	**Hex:**	**Description:**
441	01b9	The remote node determined that a necessary parameter CBB is not present in the proposed list.
442	01ba	The Proposed Data Structure Nesting Level is too small for the desired communication to the remote node.
443	01bb	The remote node determined that a necessary service CBB is missing from the proposed list.
446	01be	General problem with conclude service at remote node.
447	01bf	The remote node cannot conclude because responses are not generated for all confirmed service requests, or because upload state machine exists.
456	01c8	General problem with cancel request at remote node.
457	01c9	The attempted cancel service could not be completed at remote node due to invoke ID mismatch.
458	01ca	The cancel service cannot be performed at remote node according to service requirements.
486	01e6	The variable at the remote node has an undefined reference attribute. This is a permanent error for access to this variable.
487	01e7	Cannot access the variable at the remote node due to a hardware fault.
488	01e8	The requested variable is temporarily unavailable at the remote node.
489	01e9	The remote MMS client has insufficient privilege to request this operation.
490	01ea	The variable does not exist at the remote node.
491	01eb	The remote variable is invalid because the format of its address is incorrect or the address is out of range.
492	01ec	An inappropriate or unsupported type is specified for a variable at the remote node.
493	01ed	The remote node determined that the type specified is inconsistent with the service requested or with the referenced variable.
494	01ee	The remote node determined that the object is specified with inconsistent attributes.
495	01ef	The remote variable is not defined to allow the requested access.
496	01f0	The requested variable does not exist at the remote node.
506	01fa	The local node determined that the communication protocol is invalid. This indicates an inter–operability problem –– please contact Allen–Bradley.
507	01fb	The remote node determined that the communication protocol is invalid. This indicates an inter–operability problem –– please contact Allen–Bradley.

Appendix C

Protocol Implementation Conformance Statement (PICS)

Appendix Contents

To implement MMS protocol, certain conformance requirements must be met. These requirements cover different areas that deal with aspects of communicating on the network via MMS. There are four areas that make up the PICS:

- implementation and system information
- which service Conformance Building Block (CBBs) are supported
- which parameter CBBs are supported (and their values)
- local implementation values

The following sections provide PICS information for your Allen–Bradley PLC–5 802.4 MAP/OSI Coprocessor.

PICS Part 1: Implementation Information

The following table lists the implementation values for the your OSI coprocessor:

Implementation Information:	Value:
Implementation's vendor Name	Allen–Bradley Company
Implementation's Model Name	PLC–5 MAP/OSI Interface
Implementation's Revision Identifier	(see note 1, below)
Machine Name(s) and Version Number(s)	PLC–5/40 PLC–5/60 PLC–5/40L PLC–5/60L PLC–5/30 PLC–5/20 PLC–5/11
Operating System(s)	N/A
MMS Abstract Syntax	Yes
MMS Version Number Supported	0 and 1
MMS Companion Standard Abstract Syntaxes	None
MMS Companion Standard Version Number Supported	None
Calling MMS–user (indicate "Yes" or "No")	Yes
Called MMS–user (indicate "Yes" or "No")	Yes
List of Standardized Names	M_DAYTIME

Note for PICS Part 1

Note 1: The Revision Identifier will change with each new PLC–5 MAP/OSI Interface Software update and will reflect the version of the software.

PICS Part 2: Service CBBs

The following is the MMS Service CBB (Conformance Building Block) table, which indicates whether or not the MMS implementation fulfills the server requirements, the client requirements, or both when operating in the abstract syntax (as defined in Part 2 of ISO/IEC 9506, clause 19). The right–hand column of the following table indicates if the CBB is supported for ‘‘server”, ‘‘client”, or ‘‘both”. If the MMS implementation does not fulfill the server or client requirements for the service or CBB, then the column shall be left blank.

Service Conformance Building Blocks:	Server, Client, or Both:
Initiate	Both
Conclude	Both
Cancel	Server
Status	Server
Unsolicited Status	Server
GetNameList	Server
Identify	Server
Rename	Server
GetCapabilityList	Server
InitiateDownloadSequence	Server
DownloadSegment	Server
TerminateDownloadSequence	Server
InitiateUploadSequence	Server
UploadSegment	Server
TerminateUploadSequence	Server
RequestDomainDownload	Server
RequestDomainUpload	Server
LoadDomainContent	Server
StoreDomainContent	Server
DeleteDomain	Server
GetDomainAttributes	Server
CreateProgramInvocation	Server
DeleteProgramInvocation	Server
Start	Server
Stop	Server

Service Conformance Building Blocks:	Server, Client, or Both:
Resume	Server
Reset	Server
Kill	
GetProgramInvocationAttributes	Server
Read	Both
Write	Both
InformationReport	Server
GetVariableAccessAttributes	Server
DefineNamedVariable	Both
DefineScatteredAccess	
GetScatteredAccessAttributes	
DeleteVariableAccess	Both
DefineNamedVariableList	Server
GetNamedVariableListAttributes	Server
DeleteNamedVariableList	Server
DefineNamedType	Server
GetNamedTypeAttributes	Server
DeleteNamedType	Server
Input	
Output	
TakeControl	
RelinquishControl	
DefineSemaphore	
DeleteSemaphore	
ReportSemaphoreStatus	
ReportPoolSemaphoreStatus	
ReportSemaphoreEntryStatus	
AttachToSemaphore	
DefineEventCondition	
DeleteEventCondition	
GetEventConditionAttributes	
ReprotEventConditionStatus	
AlterEventConditionMonitoring	
TriggerEvent	
DefineEventAction	
DeleteEventAction	
GetEventActionAttributes	
ReportEventActionStatus	
DefineEventEnrollment	

Service Conformance Building Blocks:	Server, Client, or Both:
DeleteEventEnrollment	
AlterEventEnrollment	
ReportEventEnrollmentStatus	
GetEventEnrollmentAttributes	
AcknowledgeEventNotification	
AttachToEventCondition	
EventNotification	
GetAlarmSummary	
GetAlarmEnrollmentSummary	
ReadJournal	
WriteJournal	
InitializeJournal	
CreateJournal	
DeleteJournal	
ReportJournalStatus	
ObtainFile	
FileOpen	
FileRead	
FileClose	
FileRename	
FileDelete	
FileDirectory	

PICS Part 3: Parameter CBBs

The following table indicates what MMS Parameter Conformance Building Blocks are supported for the PLC–5 802.4 MAP/OSI Coprocessor. There is a ‘‘yes” in the *Supported?* column if the parameter CBB is supported by your OSI coprocessor, and blank if it is not.

Parameter Conformance Building Blocks:	Supported? (Value)
STR1	Yes
STR2	Yes
NEST (>=0. Give integer value.)	2
VNAM	Yes
VADR	Yes
VALT	
VSCA	
Parameter Conformance Building Blocks:	**Supported? (Value)**

TPY	Yes
VLIS	Yes
REAL	
AKEC	
CEI	

PICS Part 4: Local Implementation Values

The following table provides the Local Implementation Values for your PLC–5 802.4 MAP/OSI Coprocessor.

Description:	**Value:**
Range of values for floating point numbers	(see Note 1)
Supported values of the floating point exponent width	8
Supported values of the floating point format width	32
Range of values for signed integer	(see Note 2)
Range of values for unsigned integer	(see Note 3)
Maximum length for BIT STRING in bits	(see Note 4)
Maximum length for OCTET STRING in octets	(see Note 4)
Address formats for VADR horizontal CBB	Symbolic (see Note 5)
Maximum input Time Out in seconds	N/A
Level of support for time	(see Note 6)
Granularity of time in milliseconds	(see Note 7)
Uninterruptible access to variable	(see Note 8)
Priority processing for semaphores	N/A
Capabilities of VMD	(see Note 9)
Local Detail	(see Note 10)
File Name Syntax	N/A
Range of Maximum Services Outstanding Calling	1 to 5
Range of Maximum Services Outstanding Called	1 to 5
Execution Argument	(see Note 11)
Additional Code in Error Type	(see Note 12)
Additional Detail in Error Type	(see Note 13)
Method for Extended Derivation of Status Information	(see Note 14)
Description:	**Value:**
Local Detail Calling/Called	(see Note 15)

Load Data Format	(see Note 16)
Maximum number of Upload State Machines	(see Note 17)

Notes for PICS Part 4

Note 1: The PLC–5 controller supports all valid normalized single–precision floating point values, as described in IEEE 754.

Note 2: All integers in the PLC–5 controller can be accessed as signed integers within the range of –32768 (–(2**15) to 32767 ((2**15)–1) decimal.

Note 3: All integers in the PLC–5 can be accessed as unsigned integers within the range of 0 to 32767 ((2**15)–1) decimal.

Note 4: There are no pre–determined maximum lengths for strings. Maximum length is only constrained by the negotiated Local Detail parameter of the Initiate Service (see Note 15).

Note 5: Map the PLC–5 logical ASCII addressing into the Symbolic Address choice of the Address parameter. For logical ASCII address format, see PLC–5 Programmable Controller Programming Reference Manual.

Note 6: The date and time of day are supported and maintained in the PLC–5 controller. Access is through the MMS standardized variable M_DAYTIME. The Time Sequence Identifier is not supported.

Note 7: Granularity of the time of day is to 1000 milliseconds (1 second).

Note 8: Each 16–bit (word) access to the PLC–5 controller is uninterruptible.

Note 9: The defined VMD capability strings indicate the PLC–5 processor model, series, revision, and user memory size (total). These capabilities are provided for informational use and are not referenced by any MMS domain.

Note 10: The Local Detail parameter in a Status service response and in an Unsolicited Status request is a bit string which contains the processor status word and major fault word of the PLC–5 controller. The following is a description of each bit in the bitstring when set (to 1):

Bit Number:	Description:
0	Bad RAM checksum at powerup.
1	Processor is in Run mode.

2	Processor is in Test mode.
3	Processor is in Program Mode.
Bit Number:	**Description:**
4	Currently burning EEPROM.
5	Downloading in process.
6	Test edits enabled.
7	Mode Switch in REMOTE.
8	Forces enabled.
9	Forces present.
10	EEPROM successfully burned.
11	Performing online programming.
12	Processor is in debug mode.
13	User program checksum done.
14	Last program scan.
15	First program scan.
16	Bad user program memory.
17	Illegal operand address.
18	Programming error.
19	Function chart error.
20	Duplicate labels found.
21	Power protection fault.
22	Peripheral fault.
23	User generated fault.
24	Watchdog timer fault.
25	Bad system configuration.
26	Hardware fault.
27	MCP file does not exist or is not ladder or SFC.
28	PII file does not exist or is not ladder.
29	STI program does not exist or is not ladder.
30	Fault program does not exist or is not ladder.
31	Faulted program does not exist or is not ladder.

Note 11: The PLC–5 OSI Interface uses the Execution Argument parameter in the MMS Start service to determine whether to put the PLC–5 controller into ‘‘RUN” or ‘‘TEST” mode. To put the PLC–5 controller into ‘‘RUN” mode, use the MMS Start service or Resume service with the Execution Argument parameter equal to ‘‘RUN” or ‘‘run”. To put the PLC–5 controller into ‘‘TEST” mode, use the MMS Start or Resume service with the Execution Argument parameter equal to ‘‘TEST” or ‘‘test”. When no Execution Argument is specified, the Start and Resume services will use the default Execution Argument. The default Execution

Argument will be ‘‘RUN” at powerup of the PLC–5 OSI Interface. Anytime after powerup, the default Execution Argument will reflect the last (most recent) mode of the PLC–5 processor. The character string form of the Execution Argument parameter is the only form supported.

Note 12: The PLC–5 OSI Interface will return various Additional Code values when errors are detected during processing domain management services. These are in addition to the MMS Error Class and Error Code returned. The domain management Additional Code values are as listed In the following table.

Decimal:	Hex:	Value:
511	1FF	Upload terminated prematurely.
512	200	Download data segment smaller than minimum size.
513	201	Invalid image header in download data.
514	202	Invalid file type in download data.
515	203	Invalid data in program file of download data.
516	204	Download data image too big for processor type.

Note 13: The Additional Detail parameter in the Error Type is used only to indicate when memory resources have been exhausted in the PLC–5 OSI Interface.

Note 14: There is no extended derivation of status information performed.

Note 15: The local Detail Calling and Local Detail Called parameters are used as described in the NIST Stable Implementation Agreements for Open Systems Interconnect Protocols, Part 20. The maximum PDU size supported is 1800 octets.

Note 16: Load Data Format must be in the MMS Octet String format and contain a logical snapshot of the PLC–5 controller image (all program and data table files).

Note 17: The maximum number of upload state machines at any given point in time is 16.

Appendix D

The Communication Layers' Attributes

Appendix Contents

This appendix provides information on the communication layer attributes associated with your OSI coprocessor. It contains the following sections:

- Introduction to Attributes
- Definition of Default Settings
- Frequently Used Acronyms
- System–Layer Attributes
- System–Load Attributes
- MMS Attributes
- ACSE (Association Control Service Element)–Layer Attributes
- Presentation–Layer Attributes
- Session–Layer Attributes
- Transport–Layer Attributes
- Network–Layer Attributes
- LLC (Logical Link Control)–Layer Counters
- MAC (Media Access Control)–Layer Attributes
- RS–232 Port Parameters

Introduction to Attributes

Each OSI communication layer has attributes (characteristics) that in some way control or help to control the communication process. There are four different categories of attributes:

- **parameters** are variables that govern the operation of the module. The parameters are given a certain value to control the communication process. There are two basic types of parameters:
  - *Read–only* parameters allow you to view the parameter values.
  - *Read/write* parameters allow you to view and manipulate the parameter values.
- **statuses** are read–only parameters (associated with the system layer only).
- **counters** serve simply as a tally of the number of times a certain event occurs. The counters provide a status of the module's operation. Some counters have *thresholds*, which are flags (or limits) to the number of times the counter will increment. You are allowed to set some thresholds, others have pre–determined settings. You can clear most counters.
- **actions** cause the module to change its state or mode.

Not every layer has all four types of attributes.

ATTENTION: The OSI coprocessor is shipped with its attributes pre–set to Allen–Bradley default settings. Some of these should **not** be changed because it can cause severe communication problems. For the attributes that you can change, you should do so from the Allen–Bradley MAP Station Manager. For details, refer to the Allen–Bradley MAP Station Manager User's Manual (publication 6630–6.5.2).

Definition of Defaults

In some of the following tables, we refer to **default settings.** The two types of defaults are defined below:

- **Allen–Bradley communication defaults:** These are the values used for the MAC–layer and RS–232 port configuration parameters whenever the OSI coprocessor's non–volatile memory is re–initialized.
- **user communication defaults:** You have the ability to edit the MAC layer and RS–232 port parameters via the Allen–Bradley MAP Station Manager. If you do, those values are the *user communication defaults.* These are the values the OSI coprocessor uses in place of the Allen–Bradley communication defaults (unless the coprocessor's switch 2 is set to ON, see Chapter 2 for details on switch 2).

Frequently Used Acronyms

We use a variety of acronyms in the following tables. You may see:

This acronym:	Stands for:
ESH	end system hello
LD	loadable device
PDU	protocol data unit
NPDU	network (layer) protocol data unit
SPDU	session (layer) protocol data unit
TPDU	transport (layer) protocol data unit
SAP	service access point
LSAP	link (layer) service access point
SSAP	session (layer) service access point

System–Layer Attributes

The System–Layer attributes control the communication between the OSI coprocessor and the Allen–Bradley MAP Station Manager.

System–Layer Parameters:	Description:
Printf_option	When the Allen–Bradley MAP Station Manager terminal is connected directly to the RS–232 port of the OSI coprocessor, this parameter provides the option of having all diagnostic information messages that come out from the module "translated" such that they can be understood by the Manager.
Printf_enable	This enables the parameter shown above. Dictates whether or not diagnostic information messages can go out of the RS–232 port.
Pass Through Option	Reserved for future use.

System–Layer Status:	Description:
Current Mode	The current operating mode of the OSI coprocessor (Partially or Fully Operational).
Battery	A value of OK indicates that the OSI coprocessor battery has sufficient power to retain non–volatile memory.
Firmware Revision:	
Revision	The revision number of your OSI coprocessor firmware.
Software Revision:	
Revision	The revision number of your OSI coprocessor software.
Switch settings:	This reflects the condition of the OSI coprocessor's four switches (see list below).
Mode After Reset	This switch determines which mode, either Fully or Partially Operational, the OSI coprocessor attempts to transition to after a system reset or powerup. A value of OFF is interpreted as a command to Go Fully Operational. A value of ON is interpreted as a command to Go Partially Operational.
Communication Defaults	The OSI coprocessor contains both Allen–Bradley and user default values. The Allen–Bradley defaults are stored in read–only memory and the user defaults are stored in non–volatile read/write memory. This switch determines which default setting to apply after a system reset or powerup. A value of OFF is interpreted as a command to use the user defaults. A value of ON is interpreted as a command to use the Allen–Bradley defaults. See the preceding sections titled *Definition of Defaults*, for description of user and Allen–Bradley defaults.
Auto Clear	Allows you to specify whether or not the contents of non–volatile memory are to be erased (cleared) when the coprocessor is detached from the PLC–5 controller (and re–attached to a different PLC–5 controller). See Chapter 2 of this manual for the four events that must occur for the auto clear function to work.
Reserved	Reserved and should always be OFF.
Image ID:	
File name	The filename is the name of the disk file on the Allen–Bradley MAP Station Manager that contains the loadable image (software program) that was loaded into this OSI coprocessor.
Image type	The image type is an integer that represents the type of loadable image in the OSI coprocessor. This will always be the number 2.

System–Layer Status:	Description:
Date	This is the creation date of the image (software program) currently loaded into this OSI coprocessor.
Time	This is the creation time of the image (software program) currently loaded into this OSI coprocessor.

System–Layer Actions:	Description:
Edit LDIB	Allows you to access the LDIB for editing.
Configure Node RS–232 Port	Allows you to configure the parameters associated with the coprocessor's RS–232 port.
Write Entire Device Configuration to a File	Allows you to store (write to a file) the configuration information associated with a device (node).
Restore Entire Device Configuration from a File	Allows you to restore the configuration information associated with a device (node).
Node Reset	Conditionally initiates the reset process for this OSI coprocessor.
Go Partially Operational	Puts the OSI coprocessor into partially operational mode.
Go Fully Operational	Puts the OSI coprocessor into fully operation mode.

System–Load Attributes

The following tables list the System–Load parameters.

System–Load Parameters:	Description:
LD Status	Specifies whether or not loading is currently enabled or disabled.
LD Minimum Block Delay	The minimum amount of time that this loadable device (the coprocessor) requires between successive Load Data PDUs. The loadable device reports this value to the load server (the MAP Station Manager) within the LoadRequest PDU. This provides a measure of flow control within the loadable device.
LD Wait Load Response Timeout	The amount of time the loadable device will wait for a LoadResponsePDU after it has sent a LoadRequest PDU. This should take into consideration worst case network delay time. This should also take into consideration the fact that the load server may wait a period of time before sending the LoadResponsePDU.
LD Load Request Retry	The number of times the LoadRequest PDU will be retried if timeouts are occurring. When this counter is reached, the loadable device machine is aborted and enters the INACTIVE state.

System-Load Parameters:	Description:
LD Wait Data Timeout	The amount of time the loadable device will wait for LoadData or GroupStatusRequest PDUs before timing out. This should also take into consideration worst case network and inter-node processing delays.
LD PDU Retry	The number of times PDUs other than the LoadRequest PDU will be retried if timeouts are occurring. When this counter is exhausted, the loadable device machine is aborted and enters the INACTIVE state.
LD Block Size	The maximum number of data bytes in the LoadData PDU.

System-Load Actions:	Description:
Perform a System Load	Initiates a system load (initial program load).

MMS Attributes

The following table lists the MMS attributes for the OSI coprocessor.

MMS Parameters:	Description:
Retain MMS Objects	Specifies that all MMS domains, program invocations, named variables, named variable lists, and name types will be saved in non-volatile memory.
MMS Security Option	Turns the master MMS security mechanism ON and OFF.
MMS Client Timeout	Number of seconds the client will wait for a response from a server before it errors the message instruction and aborts the connection (range 1 to 120 seconds, default is 60 seconds).
MMS Number of Outgoing Connections Reserved	Number of MMS connections reserved for use by the rung ladder program.
MMS Minor Version Number	Specifies the version of MMS that is in use: IS or DIS.

MMS Counters:	Description:
MMS Maximum Number of Connections	Total number of MMS connections supported by the OSI coprocessor.
MMS Current Number of Outgoing Connections	The number of currently open connections the local system has initiated.
MMS Current Number of Incoming Connections	The number of currently open connections a remote system(s) has opened to the local system.
MMS Number of Reject PDUs Sent	The total number of MMS Reject PDUs the local system has sent since its last powerup or reset.
MMS Number of Reject PDUs Received	The total number of MMS Reject PDUs the local system has received since its last powerup or reset.
MMS Number of Aborts Sent	The total number of Abort requests the local system has sent since its last powerup or reset.
MMS Number of Aborts Received	The total number of Abort indications the local system has received since its last powerup or reset.

MMS Actions:	Description:
Use Default MMS Objects	This clears the MMS objects saved in non–volatile memory. Only the default domain, program invocation, and pre–defined named variables will exist after the OSI coprocessor resets. **Important:** The OSI coprocessor aborts all connections and resets itself after completing this action.
Save MMS Objects to a File	This saves (to a DOS file on the station manager) a copy of all MMS objects.
Restore MMS Objects from a File	This restores (from a DOS file on the station manager) a copy of all MMS objects you saved within that file. **Important:** The OSI coprocessor aborts all connections and resets itself after completing this action.

ACSE–Layer Attributes

The ACSE (Association Control Service Element)–Layer attributes are listed in the following tables.

ACSE–Layer Parameters:	Description:
ACPM Reject Rcv. Limit	A threshold level on the ACPM Reject Received counter.
ACPM Reject Sent Limit	A threshold level on the ACPM Reject Sent counter.
ACPM Abort Rcv. Limit	A threshold level on the ACPM Abort Received counter.
ACPM Abort Sent Limit	A threshold level on the ACPM Abort Sent counter.
Assoc. Pres. Abort Limit	A threshold level on the A–P Aborts counter.
Assoc. Pres. Reject Limit	A threshold level on the A–Assoc. Presentation Reject counter.

ACSE–Layer Counters:	Description:
ACPM Reject Received	The number of times an association has been rejected by the peer ACPM.
ACPM Reject Sent	The number of times the local ACSE rejected an association.
Aggregate Association Reject Received	The number of times an association reject is received by this ACSE.
Aggregate Association Reject Sent	The number of times this ACSE rejects an association for any reason.
ACPM Abort Received	The number of times an A–ABORT.indication invoked by the peer ACPM is received by this ACSE entity.
ACPM Abort Sent	The number of times this ACSE initiates an abort request to the peer ACSE.
Assoc. Pres. Aborts	The number of times an abort is invoked by the underlying presentation layer.
Assoc. Pres. Reject	The number of times an association has been rejected by the peer user.

Presentation–Layer Attributes

The following table lists the Presentation–Layer attributes for your OSI coprocessor.

Presentation–Layer Parameters:	Description:
CPR Rcv. Transient Grp Limit	A threshold level on the CPR Rcv. Transient Grp counter.
CPR Rcv. Permanent Grp Limit	A threshold on the CPR Rcv. Permanent Grp counter.
CPR Sent Permanent Grp	A threshold level on the CPR Send Permanent Grp counter.
APR Sent Protocol Error Grp Limit	A threshold level on the APR Sent Protocol Error Grp counter.

Presentation–Layer Counters:	Description:
CPR Received No Reason Count	The number of presentation connection rejections received by this presentation entity when no particular reason is given.
CPR Sent No Reason Count	The number of presentation connection rejections sent by this presentation entity when no particular reason is given.
CPR Rcv. Transient Grp Count	The number of presentation connection rejections received by this presentation entity due to transient circumstances.
CPR Rcv. Permanent Grp Count	The number of presentation connection rejections received by this presentation entity due to permanent circumstances.
CPR Send Permanent Grp Count	The number of presentation connection rejections sent by this presentation entity due to permanent circumstances.
APR Sent No Reason Count	The number of provider aborts sent by this presentation entity with no reason given.
APR Sent Protocol Error Grp Count	The number of provider aborts sent by the presentation entity due to a protocol error.

Session–Layer Attributes

The following table lists the Session–Layer Attributes for the OSI coprocessor.

Session–Layer Parameters:	Description:
Refuse SPDU Receive Permanent Limit	A threshold level on the Refuse SPDU Received Permanent counter.
Refuse SPDU Sent Permanent Limit	A threshold level on the Refuse SPDU Sent Permanent counter.
Refuse SPDU Received Temporary Limit	A threshold level on the Refuse SPDU Received Temporary counter.
Abort Sent Protocol Error Limit	A threshold level on the Abort Sent Protocol Error counter.

Session–Layer Counters:	Description:
Refuse SPDU Received No Reason	The number of times this session entity received a S–CONNECT.confirm (reject) event with no reason given.

Session–Layer Counters:	Description:
Refuse SPDU Sent No Reason	The number of times this session entity sent a S–CONNECT.response (reject) event with a provider reason member indicating no reason given.
Refuse SPDU Received Permanent	The number of times this session entity received a S–CONNECT.confirm (reject) event with a provider reason member indicating SSAP indentifier unknown or proposed protocol version(s) not supported.
Refuse SPDU Sent Permanent	The number of times this session entity sent a S–CONNECT.response (reject) event with a provider reason member indicating SSAP indentifier unknown or proposed protocol version(s) not supported.
Refuse SPDU Received Temporary	A count of the number of times this session entity received a S–CONNECT.confirm (reject) event with a provider reason member indicating session service user (presentation layer) is not attached to the SSAP or congestion at connection time.
Abort SPDU Received No Reason	The number of times this session entity received a S–P–ABORT.request event with a reason indicating unspecified.
Abort SPDU Sent No Reason	The number of times this session entity sent a S–P–ABORT.request event with a reason indicating unspecified.
Abort Sent Protocol Error	The number of times this session entity sent a S–P ABORT.request event with a reason indicating protocol errors.
Current session buffer usage	This is a count of the number of buffers the session layer is using.
Highest level of session buffer usage	This is a measure of the maximum value that the counter listed above has reached.

Transport–Layer Attributes

The following table lists the Transport–Layer attributes for the OSI coprocessor.

Transport–Layer Parameters:	Description:
Inactivity Time	An integer that specifies the maximum time that may pass before a transmitted TPDU is received by the peer transport entity. If a TPDU is not received within this time, transport closes the connection.
Local Retransmission Time	An integer that specifies the maximum time that this transport entity will wait for an acknowledgement of a previously transmitted TPDU.
Maximum Number of Retransmissions	An integer that specifies the maximum number of times that this transport entity will retransmit a TPDU without receiving an acknowledgement (assuming the TPDU requires an acknowledgement).
Connect Request Congestion Limit	A threshold level on the Connect Request Congestion counter.
Connect Request Configuration Err. Det. Limit	A threshold level on the Connect Request Configuration Error counter.
Connect Request Refused Configuration Error Limit	A threshold level on the Connect Request Refused Configuration Error counter.

Transport-Layer Parameters:	Description:
Connect Request Protocol Error Detected Limit	A threshold level on the Connect Request Protocol Error counter.
Unsuccessful Connect Request Limit	A threshold level on the Unsuccessful Connect Request counter.
Detected TPDU Protocol Error Limit	A threshold level on the Detected TPDU Protocol Error counter.
Refused TPDU Protocol Error Limit	A threshold level on the Refused TPDU Protocol Error counter.
Discard TPDU Checksum Fail Limit	A threshold level on Checksum counter.
Timeout Limit	A threshold level on the Timeout counter.
Window Time	An integer that specifies the maximum time elapsed before this transport entity will transmit updated window information.
Checksum	An integer that specifies the usage of the optional checksum in a TPDU header. A value of 0 is interpreted as Checksum **disabled**. A value of 1 is interpreted as Checksum **enabled**.
Local Acknowledge Delay Time	An integer that specifies the maximum time (in milliseconds) that this transport entity will wait to acknowledge a received TPDU.
Maximum TPDU Size	This specifies the maximum TPDU size that this transport entity can support. This number will be sent out on connect request TPDU's by this transport entity.

Transport-Layer Counters:	Description:
Connect Request Congestion	The number of times a connection request (CR TPDU) is refused due to congestion.
Conn. Req. Err. Config. Detected	The number of transport connection rejections sent by this transport entity due to a configuration error.
Conn. Req. Refused Conf. Err.	The number of transport connection rejections received by this transport due to a configuration error.
Conn. Req. Prot. Err. Detected	The number of transport connection rejections sent by this transport entity due to a protocol error.
Unsuccessful Connect Request	The number of transport connection rejections sent by this transport entity due to an unsuccessful connect request.
Detected TPDU Protocol Error	The number of invalid TPDUs (except for connect request TPDUs) received by this transport entity.
Refused TPDU Protocol Error	The number of disconnect request and error TPDUs received by this transport entity in response to any TPDU that is not a connect request TPDU.
Discarded TPDU Checksum Failure	The number of TPDUs discarded to a checksum error.
Timeout	The number of times the transport layer times out when attempting to transmit a TPDU. A timeout error occurs when the peer transport layer does not acknowledge a previously transmitted TPDU.
TPDU Sent	The number of times a TPDU is sent.
TPDU Received	The number of times a TPDU is received.
TPDU Retransmission	The number of TPDU retransmissions due to loss or error.
Advertizable Credit Reduced to Zero	The number of times a transport entity sends an AKTPDU reducing the credit allocation to zero.

Transport–Layer Counters:	Description:
Open Connections	The number of connections open at the transport layer.
Current free transport buffers	This is a count of the number of buffers that are available for use by the transport layer.
Lowest level of free transport buffer	This is the minimum value that the counter listed above has reached.

Network–Layer Attributes

The following table lists the Network–Layer attributes for the OSI coprocessor.

Network–Layer Parameters:	Description:
Lifetime	An integer that specifies the maximum time that may elapse before a transmitted NPDU is delivered to its destination.
Discard NPDU General Limit	A threshold level on the Discard NPDU General counter.
Discard NPDU Congestion Limit	A threshold level on the Discard NPDU Congestion counter.
Discard NPDU Address Limit	A threshold level on the Discard NPDU Address counter.
Discard NPDU Lifetime Exceeded Limit	A threshold level on the Discard NPDU Lifetime Exceeded counter.
Discard NPDU Unsupported Option Limit	A threshold level on the Discard NPDU Unsupported Option counter.
Discard NPDU Reassembly Limit	A threshold level on the Discard NPDU Reassembly counter.
Checksum	An integer that specifies the usage of the optional checksum in a NPDU header. A value of 0 is interpreted as Checksum **disabled** A value of 1 is interpreted as Checksum **enabled.**
Configuration Timer	This timer determines how often an End System Hello (ESH) PDU is sent from this network entity. The units are in seconds.
Redirection Timer	This timer value is placed into a Redirect (RD) PDU as the hold timer, to ensure that bad or unused routing information is not stored forever (units are in seconds).

Network–Layer Counters:	Description:
Discard NPDU General	A count of the number of NPDU's discarded for "general" reasons. General reasons include an unspecified reason, protocol procedure error, incorrect checksum, header syntax error, incomplete NPDU received, and duplicate option specified.
Discard NPDU Congestion	A count of the number of NPDU's discarded because of media congestion.
Discard NPDU Address	A count of the number of NPDU's discarded because of an unknown or inaccessible address.
Discard NPDU Lifetime Exceeded	A count of the number of NPDU's discarded because its lifetime expired.
Discard NPDU Unsupported Option	A count of the number of NPDU's discarded because it is not fully supported.
Discard NPDU Reassembly	A count of the number of NPDU's discarded because of some reassembly problem.

Network-Layer Counters:	Description:
NPDU Received	A count of the number of NPDU's received by this network entity.
NPDU Sent	A count of the number of NPDU's sent by this network entity.
Byte Sent	A count of the number of bytes sent by this network entity.
Byte Received	A count of the number of bytes received by this network entity

LLC-Layer Counters

The following table lists the LLC (Logical Link Control)–Layer counters for the OSI coprocessor.

LLC-Layer Counters:	Description:
Number of Messages Received	The number of Link Layer messages (packets) received by this entity
Number of Messages Sent	The number of Link Layer messages (packets) sent out by this entity.
Number of Received Errors	The number of Link Layer messages received that contained some form of error such as a CRC error.
Number of Send Errors	The number of Link Layer messages that could not be sent due to some form of error such as a CRC error.
Number of Rcv Buffer Overflows	The number of received Link Layer messages that could not be accepted because of local buffer congestion problems.
Test Commands Received	The number of LLC Type 1 TEST commands received by this entity.
Test Responses Sent	The number of LLC Type 1 TEST responses (to previous TEST requests) sent out by this entity.
XID Commands Received	The number of LLC Type 1 XID commands received by this entity.
XID Responses Sent	The number of LLC Type 1 XID responses (to previous XID requests) sent out by this entity.
Frames Discarded – Inactive LSAP	A count of the number of received Link Layer messages that could not be accepted because the Link Layer Service Access Point (LSAP) was not activated.
Frames Discarded – Illegal PDU Type	A count of the number of received Link Layer messages that could not be accepted because the command type was illegal or not supported by this entity.

MAC Layer Attributes

The MAC (Media Access Control)–Layer attributes control the way in which information is sent out onto the network. The OSI coprocessor's MAC–Layer attributes are listed in the following tables, along with a brief description and the default settings for each.

ATTENTION: Your OSI coprocessor is shipped with its MAC–Layer parameters pre–set to their Allen–Bradley default values. Changing these values may cause severe communication problems. Refer to the Allen–Bradley MAP Station Manager User's Manual (publication 6630–6.5.2) for details on changing them.

ATTENTION: Each device on your MAP 802.4 token–passing network must have the same slot time. Failure to set the slot time correctly may severely affect the operation of the network.

MAC Layer Parameter:	**Description:**	Default Value:
MAC address	The IEEE 802.4 address of this node	802.4 default values are created to be globally unique. This value is defined in PROM.
Group Address 1 through 8	A logical OR is performed on these group addresses to obtain the group address mask. This mask is used to determine if a frame specifying a group address should be accepted by this node.	(1) 09 00 2B 00 00 04 (2) 41 5F 42 49 50 4C (3) 01 00 00 00 00 00 (4) 01 00 00 00 00 00 (5) 01 00 00 00 00 00 (6) 01 00 00 00 00 00 (7) 01 00 00 00 00 00 (8) 01 00 00 00 00 00
High priority token hold time	Specifies the maximum amount of time (in octets) the node can transmit data before passing the token.	FFFF (Hex) 65535 (Decimal)
Priority 0 token hold time	Maximum amount of time (in octets) this station can transmit from this queue.	FFFF (Hex) 65535 (Decimal)
Priority 2 token hold time	Maximum amount of time (in octets) this station can transmit from this queue.	FFFF (Hex) 65535 (Decimal)
Priority 4 token hold time	Maximum amount of time (in octets) this station can transmit from this queue.	FFFF (Hex) 65535 (Decimal)
Target rotation time for ring maintenance	In conjunction with maximum inter solicit count and ring maintenance timer initial value, this parameter controls the frequency at which a node solicits successors.	1FFFFF (Hex) 2097151 (Decimal)
Ring maintenance initial value	Used in determining when a station solicits.	1FFFFF (Hex) 2097151 (Decimal)

MAC Layer Parameter:	**Description:**	Default Value:
Maximum inter–solicit count	How often the station opens response window.	40 (Hex) 64 (Decimal)
Slot time	Maximum time (in octets) the node will wait for a response. All nodes on the network must have the same slot time.	50 (Hex) 80 (Decimal)
Maximum PDU size	Maximum size of PDU.	1FEB (Hex) 8171 (Decimal)
Maximum non–RWR retry limit	Maximum retries of a non–RWR message.	3 (Decimal)
In Ring Desired	When set to zero, the node will only "go online" to transmit.	1

MAC–Layer Counters:	**Description:**
Successor	The source address of the node to which this node will pass the token.
Predecessor	The source address of the node which passed the token to this node.
Last token rotation time	A measure of the time in octets of the last complete token pass as seen by this node.
Frame checksum errors	Number of IEEE 802.4 CRC errors.
Number of tokens heard	Number of detected tokens being passed.
Number of who follows query	Number of times this station has had to look for a new successor.
Token pass failures	Number of times this station has failed to pass the token.
Number of successors	Number of times this station failed to find a successor.
Unexpected frames	A possible duplicate token situation.
Claim tokens	Number of claim tokens found.
Received frames too long	Number of frames this station has received that are greater than 8K bytes.
Modem errors	Number of local physical errors from modem.
E–Bit errors	Number of frames with checksum errors detected by a remodulator.
Non–silence	Number of non–silence periods received from modem.
Frame fragments	Number of frames seen with a start delimiter and no end delimiter.
Solicit Any	Number of times frame soliciting all potential successors has been sent.

MAC–Layer Actions:	**Description:**
Go off line	The node removes itself from the network.
Go on line	The node will attempt to enter the network on the next available opportunity.

RS–232 Port Parameters

The following table lists the RS–232 Port parameters for your OSI coprocessor

RS–232 Parameters:	Possible Values:	Default Value:
Baud Rate	300, 1200, 2400, 4800, 9600	9600
Data Bits	7 or 8	8
Parity	odd, even, or none	None
Stop bits	1	1

Appendix E

OSI Layer Management

Appendix Contents

Each of the OSI layers has a layer management entity (LME) that:

- is responsible for specific layer management information
- can pass that management information to the Network Management Agent (NMA) via each layer's attributes (see Appendix D for details on layer attributes).

You can view only the following type of attributes:

- attributes that can be retrieved and set
- the events (if any) that can be detected by a LME
- the actions (if any) that can take place within a layer

In the following tables:

- all listed attributes are readable (you can retrieve with a GET operation)
- we abbreviate ''attribute I.D." as Attrid
- a ''yes" in the *settable* column indicate you can write to that attribute, and a dash (___) in that column indicates you **cannot** write to it
- if you can write to an attribute, the *Set value or range* column indicates the range of values to which you can set that attribute

For more information on specific Layer Management Attributes, refer to the MAP 3.0 specifications.

ACSE Attributes

The following table lists the ACSE attributes.

ACSE Attribute:	Description:	Attrid:	Settable	Set value or range:
ACSE Profile	status containing info about this ACSE instance	0	____	
ACPM Reject received	counter for assoc rejects received from peer ACSE with **NULL** reason, or no reason specified	1	____	____
ACPM Reject Received Threshold	threshold for above counter (i.e. counter with attrid = 1)	2	YES	0...64K
ACPM Reject sent	counter for assoc rejects sent to peer ACSE with **NULL** reason, or no reason specified	3	____	____

ACSE Attribute:	Description:	Attrid:	Settable	Set value or range:
ACPM Rejects Sent Threshold	threshold for above counter (i.e. counter with attrid = 3)	4	YES	0...64K
Aggregate Association Rejects Received	counter of assoc rejects received for peer ACSE with *any* reason specified	5	____	____
Aggregate Association Rejects Sent	counter of assoc rejects sent to peer ACSE with *any* reason specified	6	____	____
ACPM Aborts Received	counter of aborts received from the peer ACSE Protocol Machine (ACPM)	7	____	____
ACPM Aborts Received Threshold	threshold for above counter (i.e. counter with attrid = 7)	8	YES	0...64K
ACPM Aborts Sent	counter of aborts sent to the peer ACSE Protocol Machine (ACPM)	9	____	____
ACPM Aborts Sent Threshold	threshold for above counter (i.e. counter with attrid = 9)	10	YES	0...64K
Assoc. Pres. Aborts	counter of aborts invoked by underlying Presentation protocol layer	11	____	____
Assoc. Pres. Aborts Threshold	threshold for above counter (i.e. counter with attrid = 11)	12	YES	0...64K
Aggregate Association Rejects Sent	counter of assoc rejects received from Presentation service provider	13	____	____
Aggregate Association Rejects Sent Threshold	threshold for above counter (i.e. counter with attrid = 13)	14	YES	0...64K

Events

The ACSE LME issues an event when any of the counter attributes listed in the table above reaches its corresponding threshold.

Actions

There are no actions defined for ACSE.

Presentation Attributes

The following table lists the Presentation Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
Presentation Profile	status containing info about the Presentation layer	0	____	
APR PPDU received no reason	counter of Pres connect rejects received with no reason	1	____	____
APR PPDU sent no reason	counter of Pres connect rejects sent with no reason	2	____	____
APR PPDU received transient group	counter of Pres connect rejects received with reason indication temp congestion or PSAP not available	3	____	____
APR PPDU received transient group threshold	threshold for above counter (i.e. counter with attrid = 3)	4	YES	0...64K
APR PPDU received permanent group	counter of Pres connect rejects received with reason indication Pres address unknown, local limit exceeded, default context not supported, or User data unreadable	5	____	____
APR PPDU received permanent group threshold	threshold for above counter (i.e. counter with attrid = 5)	6	____	____
APR PPDU sent permanent group	counter of Pres connect rejects sent with reason indicating Pres address unknown, local limit exceeded, default context not supported, or User data unreadable	7	____	____
APR PPDU sent permanent group threshold	threshold for above counter (i.e. counter with attrid = 7)	8	YES	0...64K
APR PDU sent no reason	counter of Pres provider aborts sent with no reason	9	____	____
APR PDU sent protocol error group	counter of Pres provider aborts sent with a reason indicating protocol error	10	____	____
APR PDU sent protocol error group threshold	threshold for above counter (i.e. counter with attrid = 10)	11	____	____

Events

The Presentation LME issues an event when any of the counter attributes listed in the table above reaches its corresponding threshold.

Actions

There are no actions defined for Presentation.

Session Attributes

The following table lists the Session Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
session profile	status containing info about this Session layer	0	____	
Refuse SPDU Received No Reason	count of Session connect rejects received with no reason	1	____	____
Refuse SPDU Sent No Reason	counter of Session connect rejects sent with no reason	2	____	____
Refused SPDU Received Permanent	counter of Session connect rejects received with reason indication Sess selector unknown or unsupported protocol version	3	____	____
Refused SPDU Received Permanent Threshold	threshold or above counter (i.e. counter with attrid = 3)	4	YES	0...64K
Refused SPDU Sent Permanent	counter of Session connect rejects sent with reason indicating Sess selector unknown or unsupported protocol version	5	____	____
Refused SPDU Sent Permanent Threshold	threshold for above counter (i.e. counter with attrid = 5)	6	YES	0...64K
Refused SPDU Received Temporary	counter of Session connect rejects received with reason indication Sess service user not attached to SSAP or congestion at connection time	7	____	____
Refused SPDU Received Temporary Threshold	threshold for above counter (i.e. counter with attrid = 7)	8	YES	0...64K
Abort SPDU Received No Reason	counter of Session aborts received with no reason	9	____	____
Abort SPDU Sent No Reason	counter of Session aborts sent with no reason	10	____	____
Abort Sent Protocol Error	counter of Session aborts received with reason indicating protocol error	11	____	____
Abort Sent Protocol Error Threshold	threshold for above counter (i.e. counter with attrid = 11)	12	YES	0...64K

Events

The Session LME issues an event when any of the counter attributes listed in the table above reaches its corresponding threshold.

Actions

There are no actions defined for Session.

Transport Attributes

The following table lists the Transport Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
Transport Resource Profile	status containing info about the Transport layer	0	____	
Inactivity Time	characteristic specifying the max time that can pass since a TPDU was received	2	YES	0...65,535
Local Retransmission Time	characteristic specifying the max time that can pass without receiving an ACK, resulting in a TPDU retransmission	3	YES	0..65,535
Maximum Number of Transmissions	characteristic specifying the max number of times a TPDU will be retransmitted without receiving an ACK	4	YES	0...65,535
Connect Request Congestion	counter of connect request refusals due to congestion	5	____	____
Connect Request Congestion Threshold	threshold of above counter (i.e. counter with attrid = 5)	6	YES	0...65,535
Connect Request configuration Err. Det. Limit	counter of Transport connect rejects sent due to a configuration error	7	____	____
Connect Request configuration Err. Det. Limit Threshold	threshold of above counter (i.e. counter with attrid = 7)	8	YES	0...65,535
Connect Request Refused Configuration Error Limit	counter of Transport connect rejects received due to a configuration error	9	____	____
Connect Request Refused Configuration Error Limit Threshold	threshold of above counter (i.e. counter with attrid = 9)	29	YES	0...65,535
Connect Request Protocol Error Detected Limit	counter of Transport connect rejects sent due to protocol error	11	____	____
Connect Request Protocol Error Detected Limit Threshold	threshold of above counter (i.e. counter with attrid = 11)	30	YES	0...65,535
Unsuccessful Connect Request Limit	counter of Transport connect rejects sent due to unsuccessful connect request	13	____	____
Unsuccessful Connect Request Limit Threshold	threshold for above counter (i.e. counter with attrid = 13)	14	YES	0...65,535
Detected TPDU Protocol Error Limit	counter of invalid TPDUs received	15	____	____
Detected TPDU Protocol Error Limit Threshold	threshold for above counter (i.e. counter with attrid = 15)	31	YES	0...65,535
Refused TPDU Protocol Error	counter of disconnect requests and error TPDUs received which are NOT in response to connect reqs	17	____	____

Attribute:	Description:	Attrid :	Settable:	Set value or range:
Refused TPDU Protocol Error Threshold	threshold for above counter (i.e. counter with attrid = 17)	18	YES	0...65,535
Discard TPDU Checksum Fail Limit	counter of TPDUs discarded due to checksum error	19	____	____
Discard TPDU Checksum Fail Limit Threshold	threshold for above counter (i.e. counter with attrid = 19)	20	YES	0...65,535
Timeout	counter of number of times Transport times out trying to transmitt a TPDU; a timeout occurs when a required ACK is not received after max transmit tries	21	____	____
Timeout Limit	threshold for above counter (i.e. counter with attrid = 21)	22	YES	0...65,535
Window Time	characteristic specifying max time passed be for Transport transmit updated window info	23	YES	0...65,535
TPDU Sent	counter of the number of TPDUs transmitted	24	____	____
TPDU Received	counter of the number of TPDUs received	25	____	____
TPDU Retransmission	counter of the number of TPDUs retransmitted	26	____	____
Advertizable Credit Reduced to Zero	counter of the number of times Transport sets the credit member of a Data ACK TPDU to 0	27	____	____
Open Connections	status specifying number of currently open Transport connections	28	NO	

Events

The Transport LME issues an event when any of the counter attributes listed in the table above reaches its corresponding threshold.

Actions

There are no actions defined for Transport.

Network Attributes

The following lists the Network Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
Network Profile	status containing info about the Network layer	0	____	
Lifetime	characteristic specifying the max time that can pass before a NPDU reaches its destination	1	YES	0...65,535
Discard NPDU General	counter of times NPDU is discarded due to "general" reasons	2		____
Discard NPDU General Limit	threshold for above counter (i.e. counter with attrid = 2)	3	YES	0...65,535
Discard NPDU Congestion	counter of times NPDU is discarded due to media congestion	4	____	____
Discard NPDU Congestion Limit	threshold for above counter (i.e. counter with attrid = 4)	5	YES	0...65,535
Discard NPDU Address	counter of times NPDU is discarded due to unknown or inaccessible address	6	____	____
Discard NPDU Address Limit	threshold for above counter (i.e. counter with attrid = 6)	7	YES	0...65,535
Discard NPDU Lifetime Exceeded	counter of times NPDU is discarded due to lifetime expiration	8	____	____
Discard NPDU Lifetime Exceeded Limit	threshold for above counter (i.e. counter with attrid = 8)	9	YES	0...65,535
Discard NPDU Unsupported Option	counter of times NPDU is discarded because it is not fully supported	10	____	____
Discard NPDU Unsupported Option	threshold for above counter (i.e. counter with attrid = 10)	11	YES	0...65,535
Discard NPDU Reassembly	counter of NPDUs discarded due to reassembly problem	12	____	____
Discard NPDU Reassembly Limit	threshold of above counter (i.e. counter with attrid = 12)	13	YES	0...65,535
NPDU Received	counter of NPDUs received	14	____	____
NPDU Sent	counter of NPDUs sent	15	____	____
Byte Sent	counter of bytes sent	16	____	____
Byte Received	counter of bytes received	17	____	____
Checksum	characteristic specifying if the Network checksum algorithm is enabled or not	18	YES	1 (ENABLE) 0 (DISABLE)
Routing Table	characteristic specifying a routing table entry	19	NO	

Events

The Network LME issues an event when any of the counter attributes listed in the previous table reaches its corresponding threshold.

Actions

There are currently two ACTIONS supported by the Network LME.

1. **Add routing table entry(s)**-This action adds one or more entries to the routing table. Existing entries with the same NSAP address and quality of service are overwritten.

2. **Delete routing table entry(s)**-This action deletes one or more entries from the routing table.

LLC Type 1 Attributes

The following table lists the LLC Type 1 Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
Test Commands Received	counter of TEST commands received from the MAC sublayer	0	___	___
Test Response Sent	counter of TEST responses sent to the MAC sublayer	1	___	___

Events

There are no events defined for LLC Type 1.

Actions

There are no actions defined for LLC Type 1.

MAC Attributes

The following table lists the MAC Attributes.

Attribute:	Description:	Attrid:	Settable:	Set value or range:
MAC Profile	status containing info about the MAC 802.4 sublayer	0	____	
Group Address 1 through 8	characteristic specifying the local node's 2 or 6 byte MAC address	1	YES	Assigned Value
Slot Time	characteristic specifying max time the MAC station will wait for an ACK from another station	2	YES	See MAC spec
Maximum inter–solicit count	characteristic specifying max times a token passes through before a response window opens	3	YES	See MAC spec
Maximum non–RWR retry limit	characteristic specifying max time a frame is retransmitted without receiving a response	4	YES	0...7
High priority token hold time	characteristic specifying max time a station can transmit at an access class	5	YES	See MAC spec
Priority 4 token hold time	characteristic specifying max time taken for a token of access class 4 to rotate around logical ring	6	YES	See MAC spec
Priority 2 token hold time	characteristic specifying max time taken for a token of access class 2 to rotate around logical ring	7	YES	See MAC spec
Priority 0 token hold time	characteristic specifying max time taken for a token of access class 0 to rotate around logical ring	8	YES	See MAC spec
Target rotation time for ring maintenance	characteristic specifying ring maintenance timer	9	YES	See MAC spec
Ring maintenance initial value	characteristic specifying ring maintenance timer initial value upon entry into the ring	10	YES	See MAC spec
In Ring Desired	characteristic specifying if station should be participant in the ring or not	11	YES	1 (TRUE) or 2 (FALSE)
Number of who follows query	counter of number of times **who follows** frame has been transmitted	12	____	____
Token pass failures	counter of number of times token-pass timer has expired	13	____	____
solicit any	counter of times frame soliciting all potential successors has been sent	14	____	____
Number of successors	counter of times no successor has been found	15	____	____
Unexpected frames	counter of unexpected frames received	16	____	____

Attribute:	Description:	Attrid:	Settable:	Set value or range:
Claim tokens	counter of times a claim token is successfully transmitted	17	____	____
Modem errors	counter of local physical errors from modem	19	____	____

Events

The MAC 802.4 issues an event to notify the NM Agent that it has detected another station with the same MAC address.

Actions

There are no actions defined for the MAC 802.4 LME.

ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

Allen-Bradley has been helping its customers improve productivity and quality for 90 years. A-B designs, manufactures and supports a broad range of control and automation products worldwide. They include logic processors, power and motion control devices, man-machine interfaces and sensors. Allen-Bradley is a subsidiary of Rockwell International, one of the world's leading technology companies.

With major offices worldwide.

Algeria • Argentina • Australia • Austria • Bahrain • Belgium • Brazil • Bulgaria • Canada • Chile • China, PRC • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark • Ecuador • Egypt • El Salvador • Finland • France • Germany • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Israel • Italy • Jamaica • Japan • Jordan • Korea • Kuwait • Lebanon • Malaysia • Mexico • New Zealand • Norway • Oman • Pakistan • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania • Russia-CIS • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic • Spain • Switzerland • Taiwan • Thailand • The Netherlands • Turkey • United Arab Emirates • United Kingdom • United States • Uruguay • Venezuela • Yugoslavia

World Headquarters, Allen-Bradley, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000 Fax: (1) 414 382-4444

Publication 1785-6.5.6 August 1992
Supersedes 1785-6.5.6 March 1991

PN 404636102