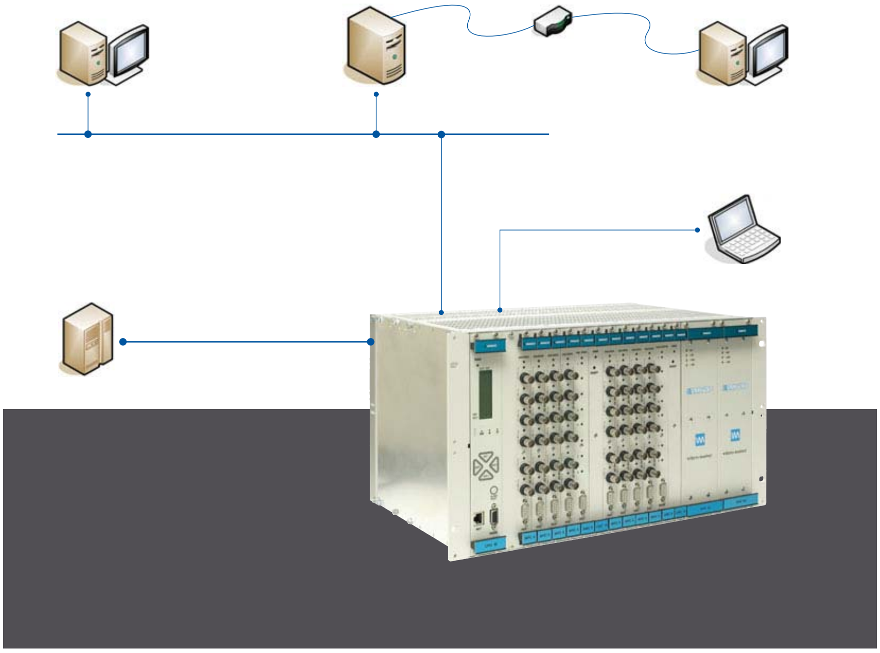
Enabling the Extraordinary
To Fly To Power To Live

MEGGiTT

MANUAL

vibro-meter®

VM600 networking



Document reference MAVM600-NET/E
Edition 10 – February 2021

Meggitt SA, Route de Moncor 4, Case postale, 1701 Fribourg, Switzerland

Tel: +41 26 407 1111
Fax: +41 26 407 1660
energy@ch.meggitt.com
www.meggittsensing.com/energy
www.meggitt.com

REVISION RECORD SHEET

Edition	Date of issue	Written by / modified by	Description	Signature
1	07.10.03	R. Meyer	Original edition	RM
2	31.03.06	N. Parker	Update to reflect new corporate identity	NP
3	13.05.09	S. Trono	Updated for Windows XP. Added AMC8 card support.	ST
4	02.09.2010	Peter Ward	Updated to reflect new improved Modbus support that is available for CPUM cards running firmware version 071 or later. This specifically affects the AMC8 and MPC4 cards and their register definition tables, which have been moved to appendices. Also updated to reflect new PROFINET support that is available for CPUM cards running firmware version 801.	PW
5	24.11.2010	Peter Ward	Minor corrections	PW
6	15.03.2012	Peter Ward	Corrections to Table A-2 on page A-4. Updated in accordance with the latest Meggitt brand guidelines.	PW
7	30.06.2014	Peter Ward	Added additional information on the dynamic host configuration protocol (DHCP). Corrected the default Modbus baud rate in 8.4.2 Communications parameters for the VM600. Updated with the latest networking information, including the use of CPUM Configurator software as supported by CPUM firmware version 075, and terminal sessions (with PuTTY and Midnight Commander). This required the restructuring of the first seven chapters of the manual.	PW
8	08.05.2015	Peter Ward	Updated the table in the Preface to include additional information on CPUM cards (see Affected products). Updated to reflect that the CPUM Configurator software is now included with VM600 MPSx software version 2.7 or later (see 6 CPUM Configurator). Clarified and corrected the introductory text on PuTTY (see 1.3.2.1 PuTTY). Updated incorrect cross-reference links and corrected typos throughout the text.	PW

Edition	Date of issue	Written by / modified by	Description	Signature
9	14.02.2018	Peter Ward	Added a statement that Meggitt SA product certifications and warranties are valid only for products purchased directly from Meggitt SA or an authorised distributor (see IMPORTANT NOTICES). Updated the table in the Preface to include additional information on CPUM cards (see Affected products). Updated to reflect changes to the CPUM Configurator software included with VM600 MPSx 2.7 build 012 or later (see 6 CPUM Configurator). Updated to include CPUM firmware version 081 which adds support for PROFINET (see 2 CPUM card directory structure and configuration files and 10 Setting up a PROFINET connection). Added end-of-life product disposal information (see 11 End-of-life product disposal). Updated the Energy product return procedure and form to be consistent with the Meggitt vibro-meter® website (see 12 Service and support).	PW
10	03.02.2021	Peter Ward	Added a note clarifying that CPUM card support for PROFIBUS uses an additional VM600 "Molex/VME" card to provide the physical PROFIBUS DP interface (see Preface). Updated Figure 10-2, Figure 10-17, Figure 10-18, Figure 10-19, Figure 10-20 and Figure 10-21. Updated to use the latest Meggitt brand identity.	

	Department	Name	Date	Signature
Technical content approved by	Engineering	Igor Karpekin	14.02.2018	IK
	Product Line Management	Michaël Hafner	03.02.2021	MH
Document released by	Technical Publications	Peter Ward	03.02.2021	PW

The duly signed master copy of this page is stored by the Technical publications department of Meggitt SA and can be obtained by writing to Technical publications.

IMPORTANT NOTICES

All statements, technical information, and recommendations in this document which relate to the products supplied by Meggitt vibro-meter® (Meggitt SA) are based on information believed to be reliable, but unless otherwise expressly agreed in writing with Meggitt SA the accuracy or completeness of such data is not guaranteed. Before using this product, you must evaluate it and determine if it is suitable for your intended application. You should also check our website at www.meggittsensing.com/energy for any updates to data sheets, Ex certificates, product drawings, user manuals, service bulletins and/or other instructions affecting the product.

Unless otherwise expressly agreed in writing with Meggitt SA, you assume all risks and liability associated with use of the product. Meggitt SA takes no responsibility for any statements related to the product which are not contained in a current English language Meggitt SA publication, nor for any statements contained in extracts, summaries, translations or any other documents not authored and produced by Meggitt SA.

The certifications and warranties applicable to the products supplied by Meggitt SA are valid only for new products purchased directly from Meggitt SA or from an authorised distributor of Meggitt SA.

Meggitt SA reserves the right to alter any part of this publication without prior notice.

EXPORT CONTROL

The information contained in this document may be subject to export control regulations of the European Community, USA or other countries. Each recipient of this document is responsible for ensuring that the transfer or use of any information contained in this document complies with all relevant export control regulations. ECN N/A.

COPYRIGHT

Published and printed by Meggitt SA in Fribourg, Switzerland.

PREFACE

About this manual

This manual provides reference information on using VM600-rack based monitoring systems, from Meggitt's vibro-meter® product line, in networks.

It is applicable to the following VM600-rack based monitoring and protection systems:

- Machinery protection system (MPS)
- Condition monitoring system (CMS).

About Meggitt and vibro-meter®

Meggitt PLC is a global engineering group, headquartered in the UK, specialising in the design and manufacture of high-performance components and systems for aerospace and energy markets.

The Meggitt facility in Fribourg, Switzerland, operates as the legal entity Meggitt SA (formerly Vibro-Meter SA). vibro-meter® is a product line of Meggitt that applies our core sensing and monitoring technologies to power generation, oil & gas and other industrial markets.

Meggitt SA produces a wide range of vibration, dynamic pressure, proximity, air-gap and other sensors capable of operation in extreme environments, electronic monitoring and protection systems, and innovative software for aerospace and land-based turbomachinery.

vibro-meter® products and solutions have been at the forefront of sensing and monitoring for more than 65 years and help keep machinery and equipment working safely, reliably and efficiently. This includes the VM600-rack based monitoring systems produced for the Meggitt vibro-meter® product line.

To learn more about Meggitt Switzerland, our proud tradition of innovation and excellence, and our solutions for energy markets and applications, visit the Meggitt vibro-meter® website at www.meggittsensing.com/energy

Who should use this manual?

The manual is intended for personnel such as operators of process monitoring/control systems using the VM600-rack based systems.

Operators are assumed to have the necessary technical training in electronics and mechanical engineering (professional certificate/diploma, or equivalent) to enable them to install, program and use the system.

Affected products

The following table provides an overview of the CPUM cards affected by this manual, highlighting some of their main specifications and features.

CPUM ordering number (PNR)	Memory		Operating system	Features					
	Disk On Chip (MB)	Compact Flash (MB)	QNX version	PPP protocol enabled	Modbus	Discrete value coded in analog register	PROFIBUS*	PROFINET	VM600 MPS rack (CPUM) security
200-595-04x-**C**1x	16		4.25						
200-595-061-**C**14	16		6.0.3						
200-595-062-**C**14									
200-595-063-**C**14	32		6.0.3						
200-595-064-**C**14				✓					
200-595-065-**C**14				✓		✓			
200-595-066-**C**14		512	6.0.3	✓		✓			
200-595-067-**C**14		512	6.0.3	✓	✓	✓			
200-595-067-**C**2x		512	6.0.3	✓	✓	✓			
200-595-068-**C**2x				✓	✓	✓			
200-595-070-**C**2x				✓	✓	✓			
200-595-071-**C**2x				✓	✓	✓			
200-595-072-**C**2x				✓	✓	✓			
200-595-073-**C**2x					✓	✓	✓		
200-595-801-**C**2x		512	6.4.1	✓	✓	✓		✓	
200-595-074-**C**2x		512	6.4.1		✓	✓	✓		
200-595-075-**C**2x					✓	✓	✓		
200-595-076-**D**xx		512	6.4.1		✓	✓	✓		
200-595-077-**D**xx					✓	✓	✓		✓
200-595-078-**D**xx					✓	✓	✓		✓
200-595-079-**D**xx					✓	✓	✓		✓
200-595-080-**D**xx		512	6.4.1		✓	✓	✓		✓
200-595-081-**D**xx		512	6.4.1		✓	✓	✓	✓	✓

Notes
See the table notes on page vii.

CPUM ordering number (PNR)	Memory		Operating system	Features					
	Disk On Chip (MB)	Compact Flash (MB)	QNX version	PPP protocol enabled	Modbus	Discrete value coded in analog register	PROFIBUS*	PROFINET	VM600 MPS rack (CPUM) security
200-595-61x-**C**1x 200-595-62x-**C**1x	16		4.25						
200-595-701-**C**14	32		6.0.3	✓					
200-595-703-**C**2x		512	6.0.3	✓	✓	✓			
200-595-724-**C**2x		512	6.0.3		✓	✓	✓		
200-595-72x-**C**2x 200-595-750-**C**2x		512	6.4.1		✓ ✓	✓ ✓	✓ ✓		

Notes

Standard off-the-shelf versions of the CPUM card run “standard” versions of firmware and are typically pre-configured with a “standard” configuration.

Customer specific versions of the CPUM card typically run “special” versions of firmware that support customer-specific functions and/or are pre-configured with a customer-specific configuration. Contact Meggitt SA for additional information.

In the **CPUM ordering number (PNR)** column, the letter **C** indicates the following versions of the CPUM card:

1 Modular CPU card with one Ethernet interface and one serial interface (integral Ethernet and serial communication ports providing 1x RS-232 or RS-485/422 and 1x integral Ethernet controller).

3 Modular CPU card with two Ethernet interfaces and one serial interface (integral Ethernet and serial communication ports, with an additional Ethernet controller module providing redundant Ethernet).

4 Modular CPU card with one Ethernet interface and two serial interfaces (integral Ethernet and serial communication ports, with an additional serial communications module providing redundant RS-485/422).

The change in the CPUM ordering number (PNR) from 200-595-075-C2x to 200-595-076-Dxx corresponds to the CPUM card update in 2015 to use a new PC/104 CPU module (PFM-541I or equivalent) that supports two Ethernet interfaces by default, which required updates to the underlying carrier board. (Previously, the CPUM card was fitted with a CPU module (MSM586SEN or equivalent) that supports one Ethernet interface by default and required an additional Ethernet controller module (MSME104 or equivalent) to be fitted in order to be Ethernet redundant.)

In the **CPUM ordering number (PNR)** column, the letter **D** indicates the following versions of the CPUM card:

3 Ethernet redundant versions of the CPUM card, that is, with two Ethernet interfaces (and two serial interfaces).

5 Serial redundant versions of the CPUM card, that is, with two Ethernet interfaces (and two serial interfaces) and two additional serial interfaces.

For an explanation of the **Discrete value coded in analog register** column, see 8.6.5 Discrete values coded in an analog value register.

*The **PROFIBUS** column indicates CPUM card support for PROFIBUS using an additional VM600 “Molex/VME” card to provide the physical PROFIBUS DP interface.

Structure of the manual

This section gives an overview of the structure of the document and the information contained within it. Some information has been deliberately repeated in different sections of the document to minimise cross-referencing and to facilitate understanding through reiteration.

The chapters are presented in a logical order. You should read those that are most relevant to you and then keep the document at hand for future reference.

The structure of the document is as follows:

Chapter 1 VM600 networking introduction and overview

Provides an introduction to working with networked VM600 racks, including communication options and recommendations. Some background information on computer (Ethernet) networks, terms and principles is also provided.

Chapter 2 CPUM card directory structure and configuration files

Provides an introduction to the different files that define the configuration and operation of the CPUM card and VM600 rack, and how to work with these files.

Chapter 3 Serial communications with a VM600 rack

Describes how to establish serial communications with a VM600 rack (CPUM card) and perform basic tasks. For example:

- Default serial communication parameters.
- How to identify the firmware version of a CPUM card.
- How to discover the IP address of a CPUM card.

Chapter 4 Setting up an Ethernet connection

Provides information on how to configure a computer to communicate with a VM600 rack (CPUM card) over an Ethernet link. For example:

- Default Ethernet communication (IP address) parameters.
- How to configure a computer for communications with a CPUM card.
- How to test communications.
- Primary and secondary CPUM card Ethernet communications.

Chapter 5 Ethernet communications with a VM600 rack

Describes how to establish Ethernet (TCP/IP) communications with a VM600 rack (CPUM card) and perform basic tasks. For example:

- How to establish Ethernet communications with a CPUM card.
- How to check the IP address of a CPUM card.
- How to change the IP address of a CPUM card and edit other configuration files.
- How to add a default gateway to a CPUM card.

Chapter 6 CPUM Configurator

Describes how to use the CPUM Configurator software to configure and manage a VM600 rack over an Ethernet link. For example:

- How to change the IP address of a CPUM card.
- How to transfer and work with configuration files.

Chapter 7 Terminal emulation

Describes how to use a terminal emulation session, using PuTTY and Midnight Commander, to configure and manage a VM600 rack over an Ethernet or serial link. For example:

- How to change the IP address of a CPUM card.
- How to transfer and work with configuration files.

Chapter 8 Setting up a Modbus connection (CPUM firmware version 067 or earlier)

Describes the Modbus RTU and Modbus TCP protocols. Provides detailed information on reading discrete inputs and registers from the VM600 rack.

Chapter 9 Setting up a Modbus connection (CPUM firmware version 071 or later)

Provides detailed additional information on the functionality added by improvements to the Modbus software interface.

Chapter 10 Setting up a PROFINET connection (CPUM firmware version 081 or later)

Provides a basic description of PROFINET and detailed information on how to configure a computer running SIMATIC STEP7 to communicate with a VM600 using PROFINET.

Chapter 11 End-of-life product disposal

Provides information and contact details concerning the environmentally friendly disposal of electrical/electronic equipment at the end of its useful life.

Chapter 12 Service and support

Provides contact details for technical queries and information concerning the repair and return of equipment.

Energy product return procedure Allows the user to indicate problems observed on a module/unit, thus enabling Meggitt customer support to repair the equipment as quickly as possible.

Energy customer feedback form Allows the user to provide us with valuable feedback on our documentation.

NOTE: When using CPUM firmware version 071 or later to implement Modbus communications, it is important to read both **Chapter 8** and **Chapter 9** because 9 Setting up a Modbus connection (CPUM firmware version 071 or later) builds on the information provided in **Chapter 8**.

NOTE: When using CPUM firmware version 081 or later to implement PROFINET communications, it is important to also read **Chapter 8** and **Chapter 9** because 10 Setting up a PROFINET connection (CPUM firmware version 081 or later) builds on the information provided in **Chapter 8** and **Chapter 9**.

Related publications and documentation

For further information on the use of a VM600 machinery monitoring and/or protection system, refer to the following Meggitt vibro-meter® documentation:

- *VM600 condition monitoring system (CMS) software manual – volumes 1 and 2* (document reference MACMS-SW/E)
- *VM600 condition monitoring system (CMS) quick start manual* (document reference MACMS-QS/E)
- *MPCC configuration software for VM600 machinery protection card* (document reference MAMPCC/E)
- *MPS1 configuration software for VM600 machinery protection systems* (document reference MAMPS1-SW/E)
- *MPS2 configuration software for VM600 machinery protection systems* (document reference MAMPS2-SW/E).

Refer also to the following hardware manuals:

- *VM600 condition monitoring system (CMS) hardware manual* (document reference MACMS-HW/E)
- *VM600 machinery protection system (MPS) – hardware manual (standard version)* (document reference MAMPS-HW/E)
- *VM600 machinery protection system (MPS) – hardware manual (CSA version)* (document reference MAMPS-HW/E-CSA).

NOTE: To ensure that the latest version of documentation is being used, visit the Meggitt vibro-meter® Energy website at www.meggittsensing.com/energy and check for any updates. Alternatively, contact your local Meggitt representative.

Abbreviations

The following table defines the abbreviations found in this manual as well as in associated Meggitt vibro-meter® documentation.

Term	Definition
ADC	Analog to digital converter
bps	bits per second
CMS	Condition monitoring system
CRC	Cyclic redundancy check
DCS	Distributed control system
DNS	Domain name system
FSD	Full scale deflection
FTP	File transfer protocol
GSDML	Generic station description markup language
IP	Internet protocol
LAN	Local area network
MB	Megabytes
Mbps	Megabits per second
MPS	Machinery protection system
NIC	Network interface card (also known as a network adapter)
PLC	Programmable logic controller
PLL	Phase-locked loop
PPP	Point-to-point protocol
RAS	Remote access service
RTU	(Modbus) Remote terminal unit
SCADA	Supervisory control and data acquisition
SIMATIC	Automation software and system developed by Siemens
SLIP	Serial line internet protocol
STP	Shielded twisted pair
TCP/IP	Transmission control protocol / internet protocol
UTP	Unshielded twisted pair
VT100	A video terminal made by Digital Equipment Coropration (DEC) that became the de facto standard for terminal emulators
WAN	Wide area network
XML	eXtensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

1 VM600 NETWORKING INTRODUCTION AND OVERVIEW

1.1 Communication possibilities

Several possibilities exist for communicating with a VM600 rack. These are listed below and summarised in Figure 1-1:

NOTE: The numbers in parentheses (…) refer to the elements of Figure 1-1.

- **Ethernet / TCP/IP** (1a and 1b)

  Allows configuration of a rack using the VM600 MPS software (MPS1 and MPS2) or VM600 CMS software, from the Meggitt vibro-meter® product line.

- **Ethernet / CPUM Configurator** (1c)

  Allows communication with a rack in order to configure network adapters (IP addressing) and modify configurations such as Modbus, NTP and/or XMx16.

  Communication is supported by a CPUM card.

  See 5 Ethernet communications with a VM600 rack and 6 CPUM Configurator for additional information.

- **Ethernet / Terminal emulation** (1d)

  Allows communication with a rack in order to configure network adapters (IP addressing) and modify configurations such as Modbus, NTP and/or XMx16.

  Communication is supported by a CPUM card.

  See 5 Ethernet communications with a VM600 rack and 7 Terminal emulation for additional information.

  NOTE: See 4 Setting up an Ethernet connection for additional information.

- **Ethernet / Modbus TCP** (2)

  Allows a distributed control system (DCS) to read discrete inputs and registers.

  Communication is supported by an IOCN card.

  See 8 Setting up a Modbus connection (CPUM firmware version 067 or earlier) and 9 Setting up a Modbus connection (CPUM firmware version 071 or later) for additional information.

- **Industrial Ethernet / PROFINET** (3)

  Allows a distributed control system (DCS) to read discrete inputs and registers.

  Communication is supported by an IOCN card.

  See 10 Setting up a PROFINET connection (CPUM firmware version 081 or later) for additional information.

- **RS-232, RS-485/422 / Modbus RTU** (4)

  Allows a distributed control system (DCS) to read discrete inputs and registers.

  Communication is supported by an IOCN card.

  See 8 Setting up a Modbus connection (CPUM firmware version 067 or earlier) and 9 Setting up a Modbus connection (CPUM firmware version 071 or later) for additional information.

- **RS-232 / Terminal emulation** (5)

  Allows communication with a rack in order to configure network adapters (IP addressing) and modify configurations such as Modbus, NTP and/or XMx16.

  Communication is supported by a CPUM card.

  See 3 Serial communications with a VM600 rack and 7 Terminal emulation for additional information.

- **RS-232 / Proprietary protocol** (6)

  Allows configuration of a stand-alone rack using the VM600 MPS1 software, from the Meggitt vibro-meter® product line.

  Communication is supported directly by a MPC4 or an AMC8 card, so it is not strictly VM600 networking and will not be considered further in this manual.

  NOTE: Refer to the *VM600 MPS1 configuration software for machinery protection systems software manual* for additional information.

- **RS-232 / PPP (Point-to-point protocol)**

  Allows configuration of a rack using the VM600 MPS software (MPS1 and MPS2) or CMS software, from the Meggitt vibro-meter® product line.

  Communication is supported by a CPUM card.

  NOTE: The use of the point-to-point protocol (PPP) for VM600 networking is deprecated and has not been supported since CPUM firmware version 072.
  Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

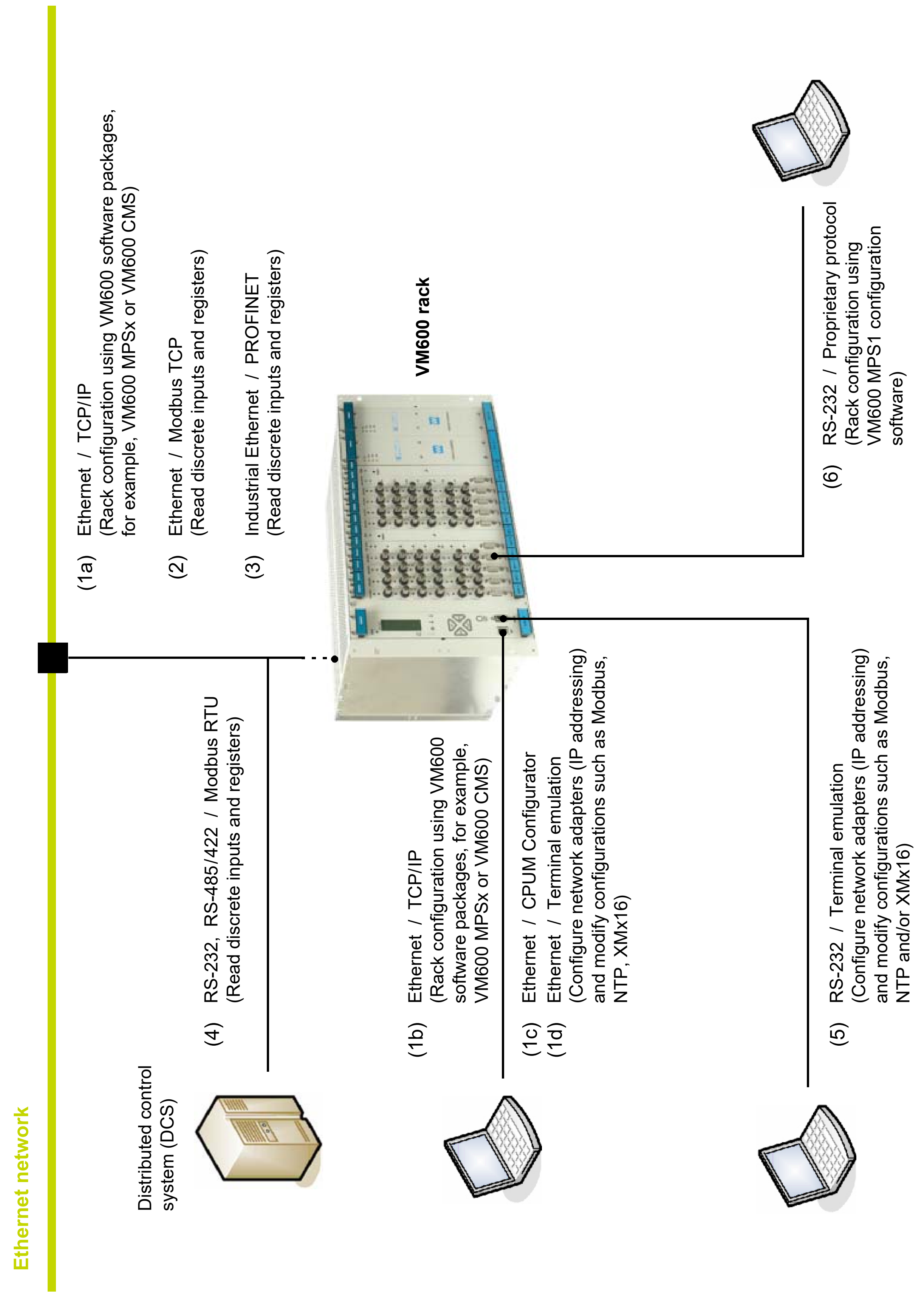

Ethernet network
Distributed control system (DCS)
(4) RS-232, RS-485/422 / Modbus RTU (Read discrete inputs and registers)
(1b) Ethernet / TCP/IP (Rack configuration using VM600 software packages, for example, VM600 MPSx or VM600 CMS)
(1c) Ethernet / CPUM Configurator
(1d) Ethernet / Terminal emulation (Configure network adapters (IP addressing) and modify configurations such as Modbus, NTP, XMx16)
(5) RS-232 / Terminal emulation (Configure network adapters (IP addressing) and modify configurations such as Modbus, NTP and/or XMx16)
(1a) Ethernet / TCP/IP (Rack configuration using VM600 software packages, for example, VM600 MPSx or VM600 CMS)
(2) Ethernet / Modbus TCP (Read discrete inputs and registers)
(3) Industrial Ethernet / PROFINET (Read discrete inputs and registers)
VM600 rack
(6) RS-232 / Proprietary protocol (Rack configuration using VM600 MPS1 configuration software)

Figure 1-1: Overview of communications possibilities

1.2 Communication recommendations

As explained in 1.1 Communication possibilities, several possibilities exist for communicating with a VM600 rack. However, not all communication possibilities are supported by all CPUM cards, as shown in Table 1-1.

Table 1-1: CPUM card communication and configuration

Communication possibilities	**CPUM firmware version**			
	072 or earlier	**073**	**074**	**075 or later**
Supported by the CPUM card (that is, included in the CPUM firmware)				
Serial (RS-232)	✓	✓	✓	✓
PPP (RS-232)	✓			
Telnet (TCP/IP)	✓	✓	✓	✓
Midnight Commander file manager			✓	✓
vi text editor	✓	✓	✓	✓
Configuration files and file structure	Depends on the version of the firmware running on the CPUM card. See 2 CPUM card directory structure and configuration files.			
Supported by the CPUM card (that is, compatible with the CPUM firmware)				
CPUM Configurator				✓

In general, the following practices are recommended as the preferred way of working with VM600 racks:

- **VM600 rack containing a CPUM card running firmware version 075 or later**

  If connected to a VM600 rack via Ethernet (TCP/IP), use the CPUM Configurator software to work with the CPUM card and configure the rack.

  If the CPUM Configurator software is not available, establish a Telnet connection to the VM600 rack with the PuTTY terminal emulator program and use the Midnight Commander file manager to configure the CPUM card.

  An FTP connection can also be established to the VM600 rack with FileZilla Client in order to download the configuration files so that they can be edited locally on a computer, then uploaded to the CPUM card.

  Alternatively, establish a Telnet connection to the VM600 rack with any terminal emulator and use the vi text editor to configure the CPUM card, as described in the *VM600 networking manual* (edition 6 or earlier).

- **VM600 rack containing a CPUM card running firmware version 074 or earlier**

  If connected to a VM600 rack via Ethernet (TCP/IP), establish a Telnet connection to the VM600 rack with the PuTTY terminal emulator program and use the Midnight Commander file manager to work with the CPUM card and configure the rack.

  An FTP connection can also be established to the VM600 rack with FileZilla Client in order to download the configuration files so that they can be edited locally on the host computer, then uploaded to the CPUM card.

Alternatively, establish a Telnet connection to the VM600 rack with any terminal emulator and use the vi text editor to configure the CPUM card, as described in the *VM600 networking manual* (edition 6 or earlier).

In order to use the CPUM Configurator software or a Telnet connection to a VM600 rack, the IP address of the CPUM card must be known.
If the IP address of the CPUM card is not known, then a serial (RS-232) connection to the card can be used in order to obtain the card's IP address. See 3.4 Discovering the IP address of a CPUM card.

1.3 Communication software and protocols

This section provides a brief overview of the software and protocols that can be used to help configure and manage VM600 racks, notably:

- For a VM600 rack containing a CPUM card running firmware version 075 or later, CPUM Configurator is the recommended tool.
- For a VM600 rack containing a CPUM card running firmware version 074, PuTTY, Midnight Commander and/or FileZilla Client are the recommended tools.
- For a VM600 rack containing a CPUM card running firmware version 073 or earlier, PuTTY, vi and/or FileZilla Client are the recommended tools.

1.3.1 CPUM Configurator

CPUM Configurator is a program that communicates with a CPUM card in a VM600 rack over an Ethernet (TCP/IP) link. Basically, it is software that provides a graphical user interface for a Telnet session between a CPUM Configurator (Telnet client) and a CPUM card (Telnet server), and is used primarily for configuring CPUM cards and managing VM600 racks over Ethernet links.

The CPUM Configurator software is included with VM600 MPSx software version 2.7 or later and is copied to the computer as part of the VM600 MPSx software installation process. It can also be obtained from Meggitt customer support (see 12.1 Contacting us).

NOTE: CPUM Configurator is compatible with CPUM cards running firmware version 075 or later (see Table 1-1).

In order to use the CPUM Configurator software to configure a VM600 rack, the IP address of the CPUM card in the rack must be known.

1.3.2 Telnet

Telnet is a network protocol used on the Internet or a LAN to provide a bidirectional text-oriented communications using a virtual terminal connection. It is a client-server protocol and is typically used to establish a connection to TCP port number 23.

Telnet is typically used to provide access to a command-line interface (usually, of an operating system) on a remote host. Most network equipment and operating systems with a TCP/IP stack support a Telnet service for remote configuration.

From a VM600 networking perspective, terminal emulator programs such as PuTTY or HyperTerminal are Telnet clients that allow connection to the Telnet server running ("listening") on the CPUM card.

NOTE: HyperTerminal is the terminal emulator program that was provided with Microsoft® Windows® XP or earlier. However, HyperTerminal was deprecated and is no longer included with Windows operating systems such as Windows Vista, Windows 7 and Windows 8.

1.3.2.1 PuTTY

PuTTY is a terminal emulator program that supports network protocols such as Telnet. A PuTTY session can be configured as a Telnet client in order to communicate with a CPUM card (Telnet server).

PuTTY also supports serial communications and can be configured to communicate directly with a VM600 card, such as a CPUM or an MPC4, via the RS-232 connector on the card's front panel.

PuTTY is free and open-source software, and is available from here:

- http://www.putty.org

In order to use PuTTY to communicate with a VM600 rack over an Ethernet communications link, the IP address of the CPUM card in the rack must be known (see 5 Ethernet communications with a VM600 rack). The IP address can be discovered using PuTTY and a serial communications link (see 3 Serial communications with a VM600 rack).

Other terminal emulation software such as HyperTerminal (or a VT100-compatible terminal) can be used instead of PuTTY.

1.3.2.2 FileZilla Client

FileZilla Client is an FTP client program that supports network protocols such as the file transfer protocol (FTP) and the secure file transfer protocol (SFTP). A FileZilla Client session is an FTP client that can communicate with a CPUM card (FTP server).

FileZilla Client is free and open-source software, and is available from here:

- https://filezilla-project.org

In order to use FileZilla Client to help configure a VM600 rack, the IP address of the CPUM card in the rack must be known.

Other FTP client software can be used instead of FileZilla Client.

1.3.2.3 Notepad++

Notepad++ is a text editor program that supports the editing of text files. Unlike Notepad, Windows built-in text editor, Notepad++ supports tabbed editing, which allows working with multiple open files in a single window.

Notepad++ is free and open-source software, and is available from here:

- http://notepad-plus-plus.org

Other text editor programs can be used instead of Notepad++.

1.3.3 Midnight Commander

Midnight Commander is a file manager program with a text user interface that is used to display a file system and allow file management operations such as selection, copying, displaying and editing to be performed.

Midnight Commander is included in CPUM firmware version 075 or later in order to provide an easy way to work with a CPUM's configuration files. The Midnight Commander command (`mc`) can be used in a terminal emulation session to manually provide all of the operations that are automatically supported by CPUM Configurator.

As Midnight Commander is included in the firmware running on the CPUM card (see Table 1-1), vi is the only alternative that can be used to edit configuration files directly on the card (see 1.3.4 vi editor). However, an FTP client can be used to download the configuration files so that they can be edited locally on a computer, then uploaded to the CPUM card.

1.3.4 vi editor

vi is a text editor program that supports the editing of text files.

vi is included in all versions of CPUM firmware in order to provide an easy way to work with a CPUM's configuration files. The vi command (`vi`) can be used in a terminal emulation session to manually edit configuration files directly on a CPUM card.

As vi is included in the firmware running on the CPUM card (see Table 1-1), Midnight Commander is the only alternative that can be used to edit configuration files directly on the card (see 1.3.3 Midnight Commander). However, an FTP client can be used to download the configuration files so that they can be edited locally on a computer, then uploaded to the CPUM card.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information on the vi text editor.

1.4 Some networking terms and definitions

This section provides a brief overview of networking terms and definitions. It is intended for users who are unfamiliar with this field.

additional information can be found in the literature or on numerous web sites.

1.4.1 Network types

Two principal types are outlined below (see also Figure 1-2).

1.4.1.1 Local area networks

Local area networks (LANs) have the following characteristics:

- Usually limited to short distances.
- Owned by the organisation that uses it.
- Usually employs solid cable, though wireless LANs are increasingly common.
- Transmission rates tend to be in the range of 10 Mbps to 1000 Mbps.

1.4.1.2 Wide area networks

Wide area networks (WANs) have the following characteristics:

- Cover extremely large areas.
- Usually owned by major communications companies.
- Systems often connected to a WAN through public networks such as the telephone system.
- The most common WAN protocol is TCP/IP.
- The Internet is an example of a WAN.

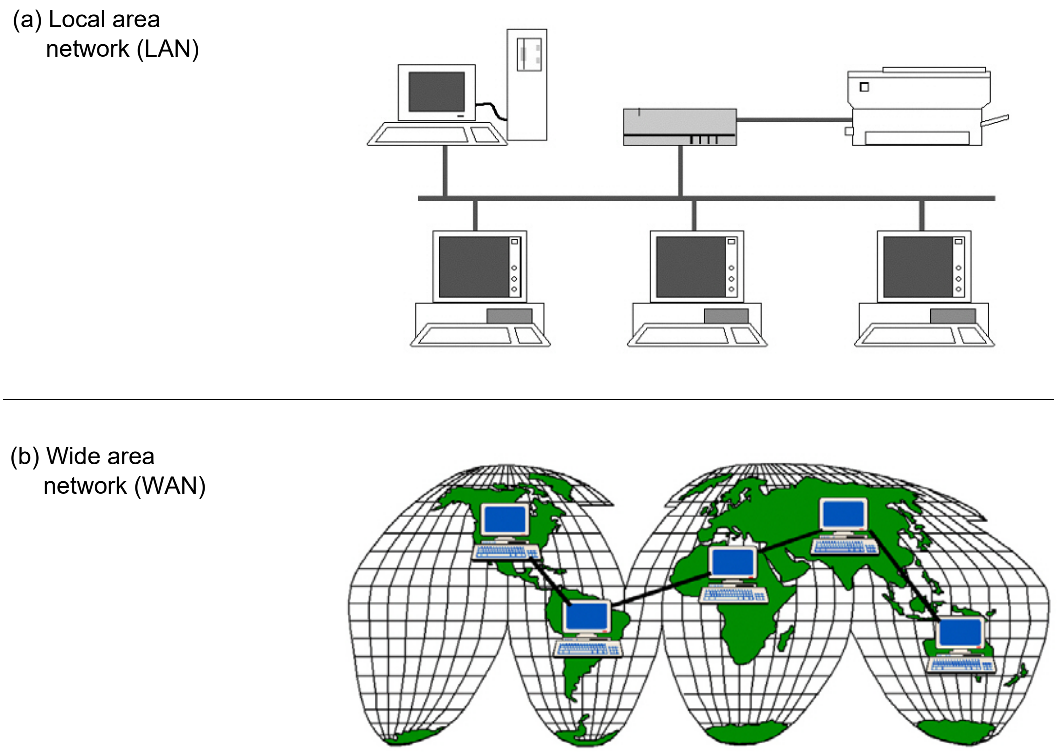


Figure 1-2: Network types (LAN and WAN)

1.4.2 Connectivity within networks

The following devices may be found in networks (see also Figure 1-3):

1- **Hubs**
A hub is a common connection point for devices in a network. Hubs are commonly used to connect segments of a LAN. A hub contains multiple ports. When a packet arrives at one port, it is copied to the other ports so that all segments of the LAN can see all packets.
The availability of low-priced network switches has largely made hubs obsolete but they are still seen in older installations and more specialised applications.

2- **Network switches**
A network switch is a network device that cross-connects clients, servers and other network devices. Also known as a frame switch, stand-alone LAN switches are common in all Ethernet networks. A four-port switch is typically built into wired and wireless routers for homes and small offices.

3- **Routers**
A router is a network device that forwards packets from one network to another. Based on internal routing tables, routers read each incoming packet and decide how best to forward it.
Most routers in homes and small offices do nothing more than direct Web, email and other Internet transactions from the local network to the cable or DSL modem, which is connected to an internet service provider and then to the Internet.
In larger companies, routers are also used to separate local area networks (LANs) into sub networks (or subnets) for network management reasons. For example, in order to balance traffic within workgroups or to filter traffic for security purposes.

4- **Gateways**
In computer networking, a gateway is a node on a network that serves as an access point to another network. That is, a it acts as a go-between (entry/exit point) for networks or subnets using the same protocols. (A router can be considered a special type of gateway.)
The default gateway is a node on a network that is used when an IP address does not match any other routes in the routing table. That is, a default gateway typically connects internal networks and outside networks (for example, the Internet). See **5.5 Gateways** and **5.6 Adding a default gateway to a CPUM card** for additional information.

NOTE: A network gateway, also known as a protocol translation/mapping convertor, is a different device that converts packets from one protocol to another. That is, a network gateway connects networks with different network protocol technologies by performing the required protocol conversions.

5- **Firewalls**
A firewall allows or blocks traffic in and out of a private network or a user's computer. Firewalls are widely used to give users secure access to the Internet as well as to separate a company's public Web server from its internal network. Firewalls are also used to keep internal network segments secure. For example, the accounting network might be vulnerable to snooping from within an organisation.
In homes, a personal firewall is typically software that is installed on the user's computer. In larger organisations, a firewall can be software in a router or server, or a stand-alone machine. It can be as simple as a single router that filters out unwanted packets, or it may comprise a combination of routers and servers each performing a specific type of firewall processing.

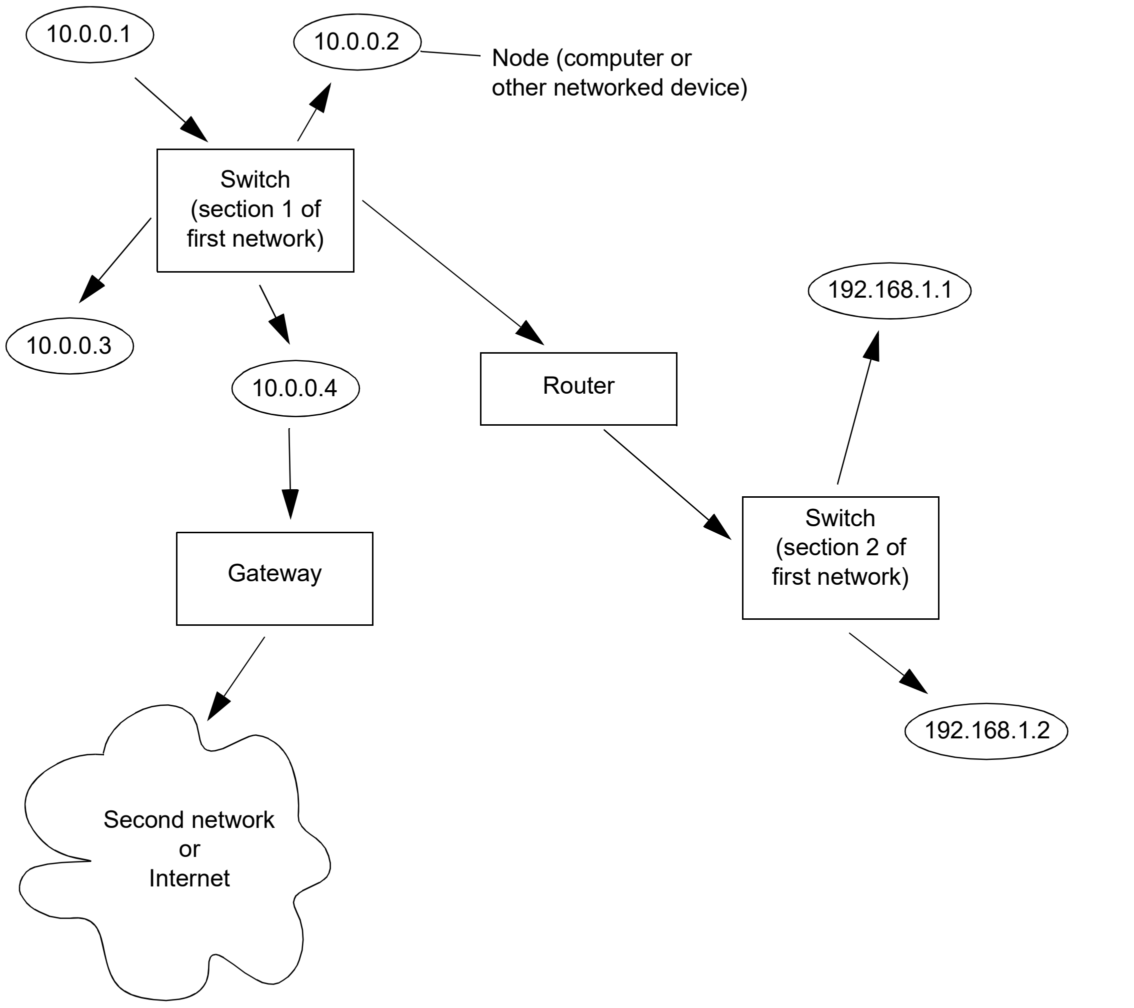

10.0.0.1
10.0.0.2
Node (computer or other networked device)
Switch (section 1 of first network)
10.0.0.3
10.0.0.4
Router
192.168.1.1
Gateway
Switch (section 2 of first network)
192.168.1.2
Second network or Internet

Figure 1-3: Network devices

1.4.3 Ethernet

Ethernet provides a method for high-bandwidth communication between devices connected on a LAN. At present, it is the most commonly used computer networking technology.

The original Ethernet specification served as a basis for the IEEE 802.3 standard. This designates both physical transmission media and the method of transmission along the media.

Over 20 variations of Ethernet are presently available. The ones most applicable to the VM600 system are:

1- 10BASE-T

- Operates at up to 10 Mbps and uses baseband transmission methods
- Uses the least expensive cable type, such as shielded twisted pair (STP) or unshielded twisted pair (UTP). (UTP is the most common)
- Maximum cable segment length is 100 meters
- Cables use 8P8C modular connectors (often called RJ45 connectors)

- Star or star-bus topologies are created, with systems connected to each other using hubs.
- Easy to expand.

2- 100BASE-T (also known as Fast Ethernet)

- Operates at up to 100 Mbps and uses baseband transmission methods
- Variants depending on physical transmission media include:

  100BASE-TX (2 pairs of high-quality twisted pairs, allowing full duplex transmission)

  100BASE-T4 (4 pairs of normal-quality twisted pairs)

  100BASE-FX (2 multi-mode fiber optic cables)
- Star topology, therefore requires a hub
- Has effectively replaced 10BASE-T.

3- 1000BASE-T (also known as gigabit Ethernet)

- Operates at up to 1000 Mbps (1 Gbps) and uses baseband transmission methods
- Variants depending on physical transmission media include:

  1000BASE-T (CAT5, CAT5e, CAT6 or CAT7 twisted pair cabling)

  1000BASE-TX (CAT6 or CAT7 twisted pair cabling)

  1000BASE-LX (multi-mode or single-mode fiber optic cables)
- Used mainly for network backbones or server-to-server connections (clustering)
- Gradually migrating down to replace 100BASE-T.

As well as supporting more complicated LAN based network topologies, Ethernet communication for a VM600 can be as simple as a crossover cable used to connect the VM600 rack directly to a host computer. See 4 Setting up an Ethernet connection for additional information.

1.4.4 Network protocols

Important network protocols for a VM600 system are:

- TCP/IP
- DHCP.

1.4.4.1 The TCP/IP protocol

- The most widely used protocol suite in the world (used for the Internet).
- Most major network operating systems (NOSs) support the use of TCP/IP.
- Originally designed for WAN use, now commonly used for LANs as well.
- Routable protocol that offers true internetworking and interoperability between disparate NOSs.
- Dynamic host configuration protocol (DHCP) allows automatic IP address assignment (see 1.4.4.2 The dynamic host configuration protocol).
- Provides full internetwork routing support.

1.4.4.2 The dynamic host configuration protocol

The dynamic host configuration protocol (DHCP) is a network configuration protocol for IP-based networks that simplifies the allocation of IP addresses to network devices.

Using DHCP on a network means that a network administrator does not have to manually assign, and the user does not have to manually enter, an IP address for each device on the network.

NOTE: Contact your IT department or network administrator for information on whether a DHCP server is available on your network.

Instead, the network uses a DHCP server that has been assigned a range (pool) of available IP addresses by a network administrator. When a new device running a DHCP client (such as a VM600 card or VibroSmart® distributed monitoring system (DMS) module) joins the network, the DHCP server will assign (loan) an IP address to the device from its pool. When the device leaves the network, the device will release the IP address, for reuse by the DHCP server. In order to do this is, the DHCP server maintains a record of its IP address pool – which addresses are in use and which are still free to be used (and reused).

It is quite typical for a network to use a combination of manually assigned (static) and DHCP assigned (dynamic) network addresses for its attached devices. However, good network administration is necessary to ensure that an IP address that has already been allocated to a DHCP server is not manually assigned (reused) elsewhere in the network.

NOTE: It is always recommended to contact your IT department or network administrator for information before manually assigning a static IP address to a device on a network.

1.4.5 IP addressing

1.4.5.1 Basic rules

- Each computer on a network must have a unique host address.
- Each network must have a unique network address.
- Each IP address must be unique to the rest of the network.
- Never use an IP address that has not been assigned to you if you are going to be connected to a network, such as a corporate network or the Internet.

1.4.5.2 IP addressing and address classes

IP addressing uses a 32-bit address (that is, four 8-bit bytes) with two parts:

- Network number.
- Host number.

Each byte of an IP address is converted into a decimal number between 1 and 255 and the bytes of the address are separated by periods (dots). This is known as dot-decimal notation. For example, 10.10.56.56 is an IP address.

Three common classes of IP address exist (see also Figure 1-4):

- Class A

  Supports approximately 16 million hosts on each of 127 networks.
  Address format: Network.Host.Host.Host

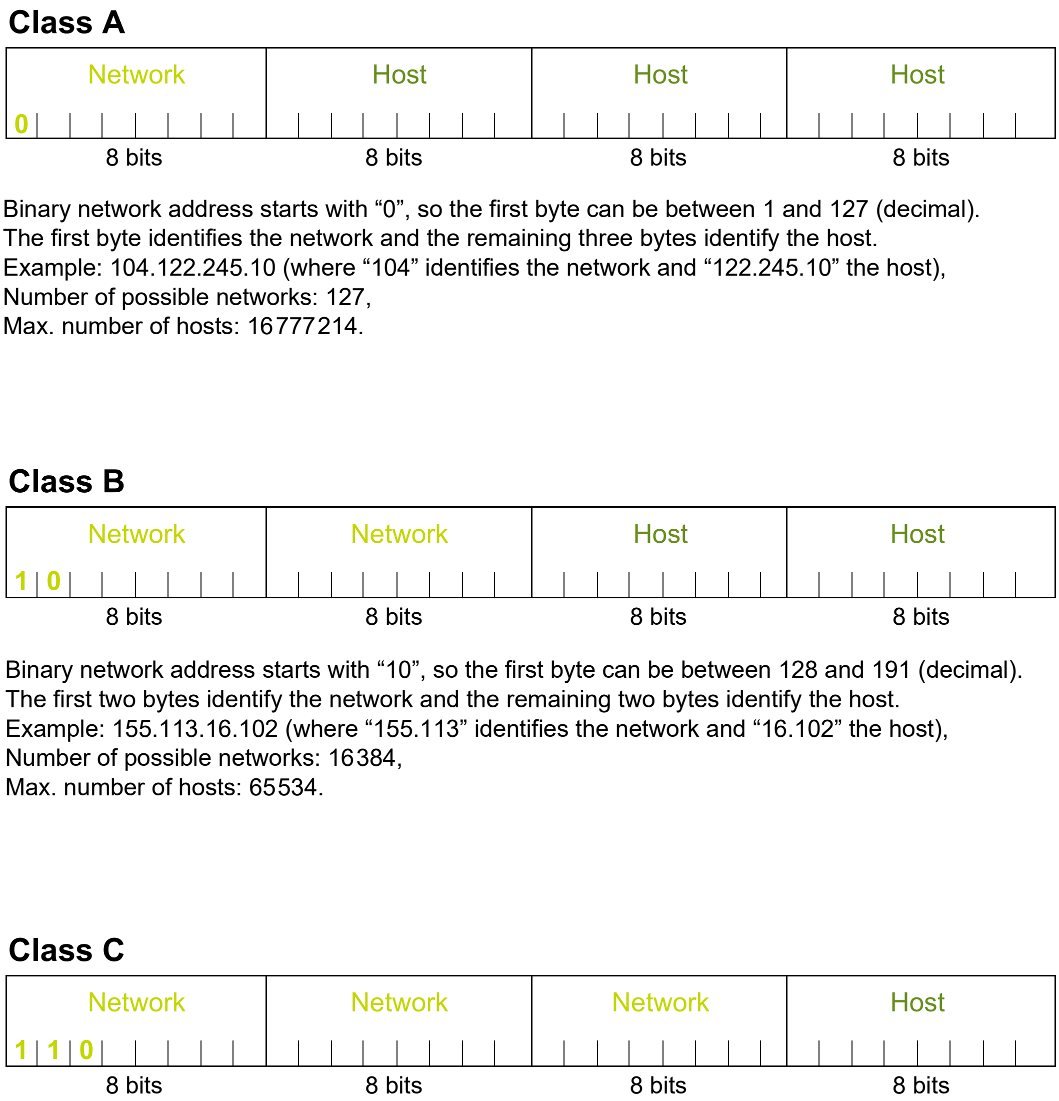
- Class B

  Supports approximately 65000 hosts on each of approximately 16000 networks.
  Address format: Network.Network.Host.Host

- Class C

  Supports 254 hosts on each of approximately 2 million networks.
  Address format: Network.Network.Network.Host

For information on the default IP addresses used by a CPUM card, see 4.1 Default IP address parameters.



Binary network address starts with “0”, so the first byte can be between 1 and 127 (decimal).
The first byte identifies the network and the remaining three bytes identify the host.
Example: 104.122.245.10 (where “104” identifies the network and “122.245.10” the host),
Number of possible networks: 127,
Max. number of hosts: 16777214.

Binary network address starts with “10”, so the first byte can be between 128 and 191 (decimal).
The first two bytes identify the network and the remaining two bytes identify the host.
Example: 155.113.16.102 (where “155.113” identifies the network and “16.102” the host),
Number of possible networks: 16384,
Max. number of hosts: 65534.

Binary network address starts with “110”, so the first byte can be between 192 and 223 (decimal).
The first three bytes identify the network and the remaining byte identifies the host.
Example: 210.222.126.55 (where “210.222.126” identifies the network and “55” the host),
Number of possible networks: 2097152,
Max. number of hosts: 254.

Note: Although 256 host addresses should be theoretically possible in this example, the addresses 0 (0b00000000 in binary) and 255 (0b11111111 in binary) are reserved.

Figure 1-4: Overview of IP address classes

1.4.5.3 Subnets and subnet masks

A subnet (subnetwork) is a logically visible subdivision of an IP network. Subnets allow you to subdivide a network into smaller, more manageable units. Benefits of subnetting include:

- Reduced network traffic by selecting the path any information is sent along.
- Optimised network performance.
- Simplified management.
- Facilitates spanning of large geographical distances.

To use a subnet, you must provide a subnet mask (or netmask), which determines the size of a subnet and the end points on the subnet. The reason it's called a subnet mask is that it literally masks out the host bits and leaves only the network address that begins the subnet. Once you know the beginning of the subnet and how big it is, you can determine the end of the subnet, which is the broadcast address.

Due to the network mask, the first subnet address is even and the last subnet address is odd. These are special addresses with specific purposes and cannot be used for general node addressing:

- The first subnet address is known as the network address. It is used to identify the network, that is, it is the official designation for a particular subnet.
- The last subnet address is known as the broadcast address. It is used to address all nodes in the network at the same time, that is, for broadcast addressing.

Default subnet masks (or natural masks) are:

• Class A:	255.0.0.0	(decimal)
	11111111.00000000.00000000.00000000	(binary)
	F F 0 0 0 0 0 0	(hexadecimal)
• Class B:	255.255.0.0	
	11111111.11111111.00000000.00000000	
	F F F F 0 0 0 0	
• Class C:	255.255.255.0	
	11111111.11111111.11111111.00000000	
	F F F F F F 0 0	

For information on the default subnet masks used by a CPUM card, see 4.1 Default IP address parameters.

THIS PAGE INTENTIONALLY LEFT BLANK

2 CPUM CARD DIRECTORY STRUCTURE AND CONFIGURATION FILES

For a networked VM600 rack, the CPUM card contains a number of different files that define the configuration and operation of the CPUM card and VM600 rack.

This includes a file that provides basic version information, and configuration files for all of the communication interfaces and protocols supported by a CPUM card (and its associated IOCN card (optional)) such as Ethernet, Serial, Modbus, PROFIBUS and PROFINET.

The directory structure and configuration files used, such as `net.cfg` or `hosts`, depend on the version of firmware running on a CPUM card.

2.1 CPUM cards running firmware version 075 or later

For CPUM cards running firmware version 075 or later, the directory structure and configuration files have been improved and reorganised. They are now significantly different compared to cards running firmware version 074 or earlier, as shown in Table 2-1.

Table 2-1: Evolution of configuration files for CPUM cards

CPUM card firmware version				Description (type of information)
070 or earlier	071 to 074	075 to 080	081 or later	
`hosts`	`hosts`	`net.cfg`		Network adapters (IP addressing) and default gateway (version 074 or earlier)

`mbcfg.2`
`mbcfg.3`
`mbcfg.4`
`mbcfg.5`
`mbcfg.tcp`
	`modbusDefault.cfg`			Modbus communications.
				PROFINET communications (version 081 or later).

			`profinet.cfg`	Additional PROFINET communications
`tcpip.1`				Network adapters, subnetting and default gateway (version 074 or earlier)
`version.txt`				Version information
		`launcher.cfg`		Launcher process (for monitoring all other processes)
		`ntp.conf`		NTP server
		`sysinit`		Default gateway
		`uc.cfg`		Summary information
		`xmcsrv.cfg`		XMx16 cards – summary
		`xmcnn.xml`		XMx16 card – individual

See 6.1.1 CPUM card configuration files for more information on the directory structure and configuration files for CPUM cards running firmware version 075 or later.

2.2 CPUM cards running firmware version 074 or earlier

Refer to the *VM600 networking manual* (edition 6 or earlier) for information on the directory structure and configuration files for CPUM cards running firmware version 074 or earlier.

2.3 Modbus configuration files

It is sometimes necessary to change the Modbus address of a rack or various communications parameters such as baud rate or parity. This is done by editing the Modbus-related configuration files stored on the CPUM.

The configuration files to edit depend on the implementation of the Modbus software interface, which depends on the version of firmware running on the CPUM card.

2.3.1 CPUM cards running firmware version 071 or later

For CPUM cards running firmware version 071 or later, the Modbus server is more flexible and completely configurable by a single file. This configuration file is listed in Table 2-2 and is stored on the CPUM.

Table 2-2: Modbus configuration file for CPUM cards running firmware version 071 or later

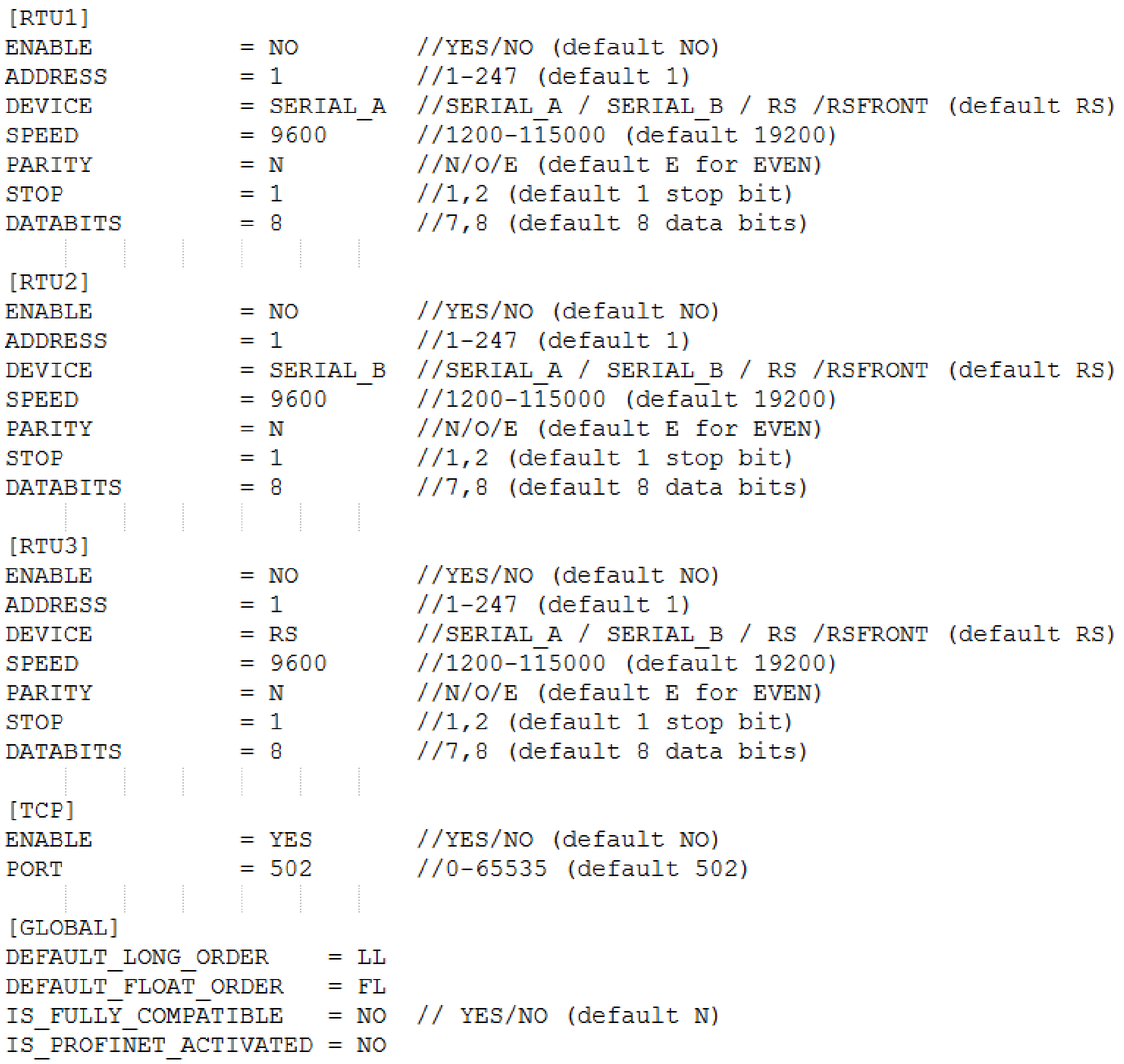
Modbus configuration file	Description
`modbusDefault.cfg`	The configuration file for Modbus communications and PROFINET communications (version 081 or later)

Note: The directory structure (location) for the Modbus configuration files on the CPUM card is:
`etc/cpum/mbsrv/modbusDefault.cfg`

NOTE: For CPUM cards running firmware version 071 or later, only the single configuration file listed in Table 2-2 must be modified to change communications parameters. The configuration file to edit is the same for all communications interfaces.

The communication parameters are defined in the top sections of the `modbusDefault.cfg` configuration file, labelled [RTUx], [TCP] and [GLOBAL]. See Figure 2-1 and 9.2 Definition of registers for additional information.

NOTE: For compatibility with existing equipment, the default Modbus configuration file supplied with all delivered systems is set up to support the same behaviour as existing CPUM cards (that is, CPUM cards running firmware version 067 or earlier).



```
[RTU1]
ENABLE          = NO        //YES/NO (default NO)
ADDRESS         = 1         //1-247 (default 1)
DEVICE          = SERIAL_A  //SERIAL_A / SERIAL_B / RS /RSFRONT (default RS)
SPEED           = 9600      //1200-115000 (default 19200)
PARITY          = N         //N/O/E (default E for EVEN)
STOP            = 1         //1,2 (default 1 stop bit)
DATABITS        = 8         //7,8 (default 8 data bits)

[RTU2]
ENABLE          = NO        //YES/NO (default NO)
ADDRESS         = 1         //1-247 (default 1)
DEVICE          = SERIAL_B  //SERIAL_A / SERIAL_B / RS /RSFRONT (default RS)
SPEED           = 9600      //1200-115000 (default 19200)
PARITY          = N         //N/O/E (default E for EVEN)
STOP            = 1         //1,2 (default 1 stop bit)
DATABITS        = 8         //7,8 (default 8 data bits)

[RTU3]
ENABLE          = NO        //YES/NO (default NO)
ADDRESS         = 1         //1-247 (default 1)
DEVICE          = RS        //SERIAL_A / SERIAL_B / RS /RSFRONT (default RS)
SPEED           = 9600      //1200-115000 (default 19200)
PARITY          = N         //N/O/E (default E for EVEN)
STOP            = 1         //1,2 (default 1 stop bit)
DATABITS        = 8         //7,8 (default 8 data bits)

[TCP]
ENABLE          = YES       //YES/NO (default NO)
PORT            = 502       //0-65535 (default 502)

[GLOBAL]
DEFAULT_LONG_ORDER     = LL
DEFAULT_FLOAT_ORDER    = FL
IS_FULLY_COMPATIBLE    = NO  // YES/NO (default N)
IS_PROFINET_ACTIVATED  = NO
```

Figure 2-1: Extract from a typical `modbusDefault.cfg` configuration file used for Modbus communications

2.3.2 CPUM cards running firmware version 067 or earlier

For CPUM cards running firmware version 067 or earlier, the Modbus server is configured using multiple files. These configuration files are listed in Table 2-3 and are stored on the CPUM.

Table 2-3: Modbus configuration files for CPUM cards running firmware version 067 or earlier

Modbus configuration file	Description
`mbcfg.2`	The configuration file for communication that occurs through the **RS-232** connector on the CPUM card. This uses a 9-pin D-type connector.
`mbcfg.3`	The configuration file for communication that occurs through the **RS** connector on the IOCN card. This uses a 6P6C (RJ11/RJ25) connector.
`mbcfg.4`	The configuration file for communication that occurs through the **A** connectors on the IOCN card. These use 6P6C (RJ11/RJ25) connectors.
`mbcfg.5`	The configuration file for communication that occurs through the **B** connectors on the IOCN card. These use 6P6C (RJ11/RJ25) connectors.
`mbcfg.tcp`	The configuration file for communication that occurs through the **1** connector on the IOCN for the primary network interface (ETH1) and through the **2** connector on the front panel of the IOCN card for the secondary network interface (ETH2), if installed. These use 8P8C (RJ45) connectors.

NOTE: For CPUM cards running firmware version 067 or earlier, only some of the configuration files listed in Table 2-3 must be modified to change communications parameters. The configuration file to edit depends on the communications interface being used.

Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information on the connectors and interfaces supported by a CPUM / IOCN card pair.

2.4 PROFINET configuration file

2.4.1 CPUM cards running firmware version 081 or later

For CPUM cards running firmware version 081 or later, the implementation of the PROFINET software interface builds on the existing Modbus server by using an additional PROFINET configuration file. This configuration file is listed in Table 2-4 and is stored on the CPUM.

Table 2-4: PROFINET configuration files for CPUM cards running firmware version 081 or later

PROFINET configuration file	Description
`profinet.cfg`	An additional configuration file for PROFINET communications (version 081 or later)

Note: The directory structure (location) for the PROFINET configuration file on the CPUM card is: `etc/cpum/mbsrv/profinet.cfg`

See 10.6 PROFINET configuration files for additional information.

2.4.2 CPUM cards running firmware version 801

For CPUM cards running firmware version 801, refer to the *VM600 networking manual, edition 8 (or earlier),* for information on the directory structure and configuration files related to the earlier implementation of the PROFINET software interface.

2.5 Working with configuration files

A CPUM card’s configuration files can be displayed or edited in a number of different ways. See also 1.2 Communication recommendations.

For CPUM cards running firmware version 075 or later:

- Using an Ethernet communications link either to edit files directly on the CPUM card or to download files from the CPUM card, edit the files locally using a text editor on a computer, then upload the files to the CPUM card. For example, using the CPUM Configurator software (see 6 CPUM Configurator).
- Using an Ethernet communications link to download files from the CPUM card, edit the files locally using a text editor on a computer, then upload the files to the CPUM card. For example, an FTP session using FileZilla Client and Notepad++ (a text editor).
- Using an Ethernet communications link to display and edit files directly on the CPUM card. For example, a Telnet session running Midnight Commander on the CPUM card.

For CPUM cards running firmware version 074:

- Using an Ethernet communications link to download files from the CPUM card, edit the files locally using a text editor on a computer, then upload the files to the CPUM card. For example, an FTP session using FileZilla Client and Notepad++ (a text editor).
- Using a serial communications link or an Ethernet communications link to display and edit files on the CPUM card. For example, a Telnet session running Midnight Commander on the CPUM card.

For CPUM cards running firmware version 073 or earlier:

- Using a serial communications link or an Ethernet communications link to display and edit files directly on the CPUM card. For example, a serial or a Telnet session running the vi text editor on the CPUM card.

- Using an Ethernet communications link to download files from the CPUM card, edit the files locally using a text editor on the computer, then upload the files to the CPUM card. For example, an FTP session using FileZilla Client and Notepad++ (a text editor).

For information on Ethernet communications links and the tools used, see 4 Setting up an Ethernet connection, 5 Ethernet communications with a VM600 rack, 6 CPUM Configurator and 7 Terminal emulation. For information on serial communications links and the tools used, see 3 Serial communications with a VM600 rack.

In general, serial communications is only used with a networked VM600 rack in order to discover the IP address of the CPUM card. Because once the IP address of a CPUM card (VM600 rack) is known, the management of networked VM600 racks can be more easily achieved using Ethernet-based communications such as:

- The CPUM Configurator software running on a computer (see 6 CPUM Configurator).
- A terminal emulation session on a computer running Midnight Commander on the CPUM card (see 7 Terminal emulation).

2.6 CPUM card configurations

CPUM cards are typically supplied pre-configured with the configuration specified at the time of ordering using the ordering number (PNR) and particular option code.

NOTE: A VM600 rack is usually fully configured in the factory before delivery so that it can be employed as is. That is, if no other configuration is specified, a VM600 rack is supplied with a ‘standard’ Modbus configuration as described in 2.6.1 Standard configurations.
For information on other configuration possibilities, including customer-specific configurations, contact Meggitt SA.

2.6.1 Standard configurations

If no other configuration is specified, a CPUM card is supplied with a ‘standard’ Modbus configuration that features:

- Default IP addresses: 10.10.56.56 for the primary network interface (ETH1) and 10.10.56.156 for the secondary network interface (ETH2).
- Default NTP configuration: local synchronisation (that is, the CPUM uses an internal clock as a time reference).
- Default Modbus configuration: standard register mapping for MPC4 and AMC8 cards in slots 3 to 14 of the VM600 rack.

  NOTE: A complete list of Modbus starting addresses as an Excel® spreadsheet is available from Meggitt SA on request.

- No configurations for any other cards in the VM600 rack.

NOTE: The ‘standard’ Modbus configuration includes support for Modbus but PROFINET is not included by default. For information on PROFINET configuration possibilities, see 10 Setting up a PROFINET connection (CPUM firmware version 081 or later).

2.6.2 Customer-specific configurations

When specified, a CPUM card is supplied with a ‘customer-specific’ configuration that corresponds to the configuration agreed with Meggitt SA and specified using an ordering number (PNR) and particular option code.

In such cases, the CPUM card is typically pre-loaded with the ‘customer-specific’ configuration before shipping.

2.6.3 Changing configurations

The CPUM Configurator software can be used to download a configuration from a CPUM card or upload a configuration to a CPUM card (see 6 CPUM Configurator).

Accordingly, this allows the configuration running on a CPUM card to be modified as required. For example, by downloading, modifying and then uploading the configuration or by uploading a new different configuration.

NOTE: It is strongly recommended to use the CPUM Configurator software to make a backup copy of the configuration running on a CPUM card.

NOTE: A backup copy of the configuration running on a CPUM card is especially important if you are going to make changes to the supplied configuration yourself, as you can revert to the backup copy if you experience problems.

NOTE: If required, an archive copy of the configuration pre-loaded on a CPUM card before shipping can be obtained from Meggitt SA by contacting the Customer support department with customer-specific information such as the Meggitt SA order number, customer order number and/or ordering number (PNR) and particular option code for the CPUM card. See 12.1 Contacting us.

THIS PAGE INTENTIONALLY LEFT BLANK

3 SERIAL COMMUNICATIONS WITH A VM600 RACK

There are many different ways of communicating with and configuring networked VM600 racks.

NOTE: Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information on networked racks and stand-alone racks.

Historically, VM600 racks were managed using a terminal emulator program (for example, PuTTY or HyperTerminal) running on a computer to establish communications with the VM600 rack (CPUM card), then running the vi text editor provided by the CPUM card's firmware to edit the configuration files as necessary.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information on using HyperTerminal and the vi text editor to manage networked VM600 racks.

However, once the IP address of a CPUM card (VM600 rack) is known, the management of networked VM600 racks can be more easily achieved using Ethernet-based communications such as:

- The CPUM Configurator software running on a computer (see 6 CPUM Configurator).
- A terminal emulation session on a computer running Midnight Commander on the CPUM card (see 7 Terminal emulation).

However, before Ethernet-based communications can be used, the IP address of the CPUM card in the networked VM600 rack must be known. This information can be obtained directly from the CPUM card using a serial communications link, as described below.

3.1 Default serial communication parameters

The default serial communication parameters factory for a CPUM card are given in Table 3-1

Table 3-1: Default serial communication parameters for a CPUM card

Baud rate	Character length (data bits)	Parity	Stop bits	Flow control
57600	8	None	1	None

Any terminal emulation software used to communicate with a CPUM card must use be configured with the communication parameters given in Table 3-1.

3.2 Establishing serial communications with a CPUM card

The exact procedure to be followed will depend on the operating system and software tools used. The following description uses the PuTTY terminal emulator on Windows® 7. If you are using a different operating system or terminal emulator that supports serial communications protocols, you can still follow this description for guidance.

In case of questions or problems, consult the Windows help or contact your system administrator. If no solution can be found, contact your local Meggitt representative.

1- Connect an RS-232 cable between the D.sub (**RS-232**) connector on the front panel of the CPUM card and the serial port of a computer. A straight-through RS-232 cable must be used, as shown in Figure 3-1.

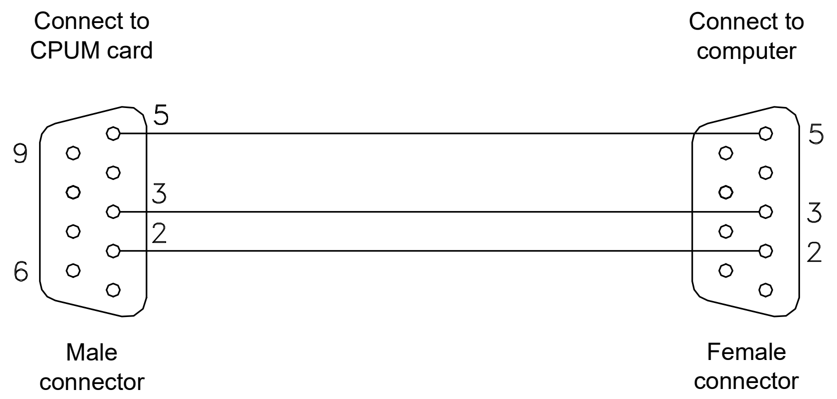
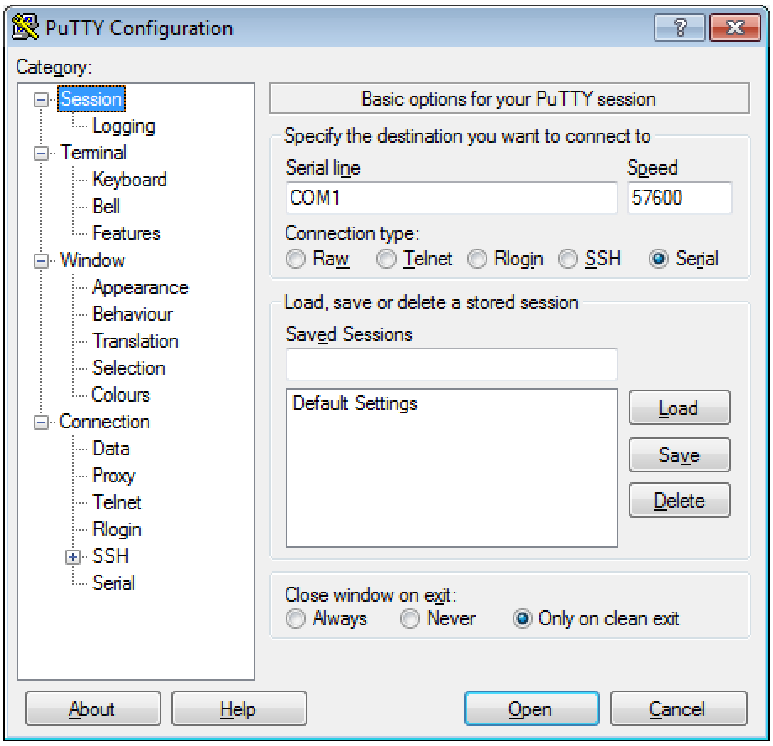


Figure 3-1: Interface cable used to connect a CPUM card to the serial port of a computer running terminal emulation software

Alternatively, a serial to USB (RS-232↔USB) converter cable can be used for a computer without an RS-232 connector.

2- Start PuTTY and select a **Serial** connection, as shown in Figure 3-2.

Alternatively, a different terminal editor that supports serial connections could be used.



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM1

Speed: 57600

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Load, save or delete a stored session

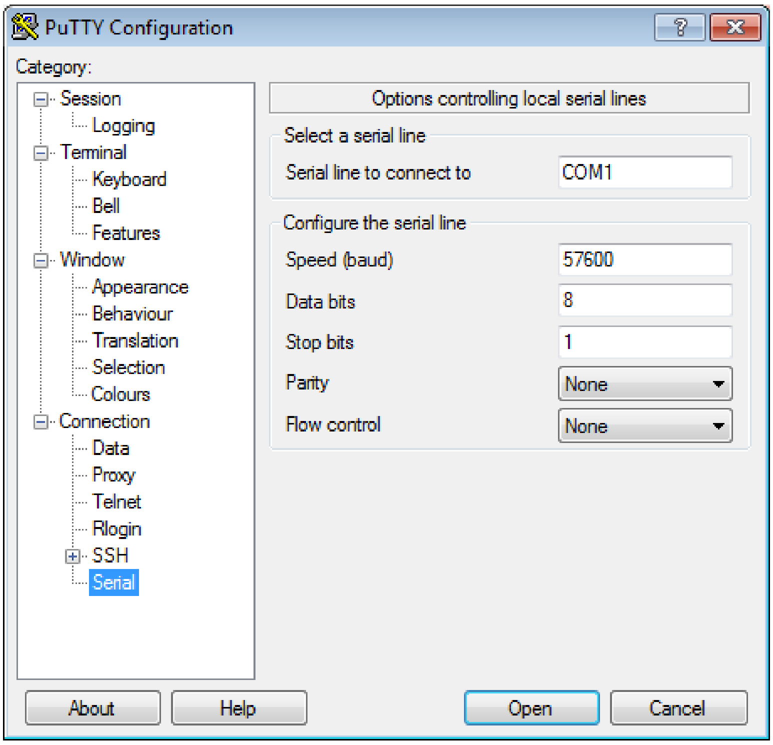
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 3-2: Using PuTTY terminal emulation software – selecting a serial connection

3- Configure the PuTTY serial session options, as shown in Figure 3-3.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM1

Configure the serial line

Speed (baud): 57600

Data bits: 8

Stop bits: 1

Parity: None

Flow control: None

Figure 3-3: Using PuTTY terminal emulation software – configuring a serial connection

4- Click **Open** to start the PuTTY session, then press the ENTER key.

5- When prompted for a login, type **user** (the default login), then press ENTER to continue. For example:

```
login: user
```

When prompted for a password, type **config** (the default password), then press ENTER to continue. For example:

```
password: config
```

6- After the login and password are accepted and the user is logged in, the command prompt is displayed. For example:

```
/usr/user #
```

3.3 Identifying the firmware version of a CPUM card

It is important to know the version of the firmware running on a CPUM card as the networking features, software compatibility, and configuration files and structure can be different, depending on the firmware version (see 2 CPUM card directory structure and configuration files).

However, the firmware version is given in the `version.txt` file for all CPUM cards.

1- Establish serial communications with a CPUM card as described in 3.2 Establishing serial communications with a CPUM card.

2- At the command prompt (`/usr/user #`), type **ls**, then press ENTER to continue. For example:

```
/usr/user # ls
```

The list command lists the files in the current working directory (`/usr/user`) are listed, including the `version.txt` file which contains the firmware version information for a CPUM card.

3- At the command prompt (`/usr/user #`), type **more version.txt**, then press ENTER to continue. For example:

```
/usr/user # more version.txt
```

The `more` command displays the contents of the specified text file one screen at a time. If required, pressing ENTER or SPACEBAR displays the next *n* lines of text.

Alternatively, the `cat` command can be used to display the contents of a file without pausing.

3.3.1 About the version.txt file

3.3.1.1 Example version.txt file

```
CPU-M : 200-595-075-HHh
CF    : 209-595-300-075

Software version at last startup
--------------------------------
SW_VER : 0.7;5
SW     : 209-595-300-075

Individual SW checksums
-----------------------
xmcsrv           365156814
mbsrv           3104950782
rpcamcsrv       1060545938
rpccmcsrv         49297480
rpccpusrv       1030630127
rpcmpcsrv       1406217733
vmesrv             1903091
testfp          2988981983
vmetest         1047897882
trigwdog        3766606088
setboot         2042170770
launcher        2954264382
```

3.3.1.2 Description

`CPU-M` is the ordering number (PNR) for the `CPUM` card.

In this PNR (`200-595-0SS-xHh`):

- `SS` represents the firmware version.
- `x` represents the hardware version:
  - `1` indicates the standard version of the card.
  - `3` indicates the Ethernet redundant version of the card.
  - `4` indicates the serial redundant version of the card.
- `H` increments are for major hardware modifications that can affect product interchangeability.
- h increments for minor hardware modifications that have no effect on interchangeability.

`CF` is the ordering number (PNR) for the CompactFlash memory card (containing the firmware) that is installed on the card.

In this PNR (`209-595-300-0SS`):

- `SS` represents the firmware version.

`SW_VER` is the version of the firmware running on the card. For example, version 075.

`SW` is the ordering number (PNR) for the firmware (on the CompactFlash memory) that is installed on the card.

NOTE: The version of firmware running on the CPUM card is given in multiple locations. For example, 3.3.1 About the version.txt file uses firmware version 075:

```
CPU-M  : xxx-xxx-075-xxx
CF     : xxx-xxx-xxx-075
SW_VER : 0.7;5
SW     : xxx-xxx-xxx-075.
```

3.4 Discovering the IP address of a CPUM card

3.4.1 CPUM cards running firmware version 075 or later

If the IP address of a CPUM card is not known, it can always be discovered using a serial connection to the RS-232 connector on the front panel of the CPUM card and terminal emulation software. For more information on CPUM card IP addresses, see 4.1 Default IP address parameters.

The directory structure and configuration files, depend on the version of firmware running on a CPUM card (see 2 CPUM card directory structure and configuration files). For a CPUM card running firmware version 075 or later, the IP address information is given in the `net.cfg` file.

1- Establish serial communications with a CPUM card as described in 3.2 Establishing serial communications with a CPUM card.

2- At the command prompt (`/usr/user #`), type **ls**, then press ENTER to continue. For example:

```
/usr/user # ls
```

The list command lists the files in the current working directory (`/usr/user`) are listed, including the `net.cfg` file which contains the IP address information for a CPUM card running firmware version 075 or later.

3- At the command prompt (`/usr/user # `), type **more net.cfg**, then press ENTER to continue. For example:

```
/usr/user # more net.cfg
```

The `more` command displays the contents of the specified text file one screen at a time. If required, pressing ENTER or SPACEBAR displays the next *n* lines of text.

4- In the `net.cfg` file, the IP addresses of the network interfaces for the CPUM card are displayed under `#main network connection` as `net0_ipaddr` (for the primary network interface (ETH1)) and under `#redundant network connection` as `net1_ipaddr` (for the secondary network interface (ETH2)).

3.4.2 About the net.cfg file

3.4.2.1 Example net.cfg file

```
#VM600 CPUM TCPIP Configuration

#main network connection
net0_ipaddr = 10.10.56.56
net0_netmask = 0xFFFFFF80
net0_force10Mb = 0
#redundant network connection
net1_ipaddr = 10.10.56.156
net1_netmask = 0xFFFFFF80
net1_force10Mb = 1
```

As shown in the example `net.cfg` file above:

- The `net.cfg` file requires netmasks in hexadecimal notation.
  (Netmasks are typically written in either hexadecimal or dot-decimal notation, for example, 0xFFFFFF80 or 255.255.255.128).
- The primary network interface (ETH1) can run at 100 Mbps, which is configured using `net0_force10Mb = 0`.
- The secondary network interface (ETH2) must run at 10 Mbps, which is configured using `net1_force10Mb = 1`.

NOTE: Only the primary network interface (ETH1) can run at 100 Mbps:
The secondary network interface (ETH2) is limited to 10 Mbps, in order to be backwards compatible with older VM600 CPUM systems.

NOTE: Only the primary network interface (ETH1) can be used for PROFINET communications.
The secondary network interface (ETH2) is limited to 10 Mbps, while the VM600 CPUM implements PROFINET IO which requires 100 Mbps.

3.4.2.2 Description of net.cfg file

# is used for line comments.

`#main network connection` is a comment used to indicate the section of the file used for the primary network interface (ETH1) that is available on either the front panel of the `CPUM` card or the front panel of the `IOCN` card.

NOTE: The primary network interface (ETH1) of a CPUM card can be routed either via the 8P8C connector (**NET**) on the front panel of the CPUM card or via the upper 8P8C connector (**1**) on the front panel of its associated IOCN card (optional). The choice is made using jumpers J42, J40, J52 and J53 on the CPUM card. For example, when all of these jumpers are removed, the primary network interface is routed via the CPUM card.
The secondary network interface (ETH2) of a CPUM card is routed via the lower 8P8C connector (**2**) on the front panel of its associated IOCN card (optional).
Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information.

`net0_ipaddr` is the IP address used for the primary network interface (ETH1).

`net0_netmask` is the network mask used for the primary network interface.

`net0_force10Mb` is used to limit the data transfer rate for the primary network interface, if required:

- `net0_force10Mb = 0` allows a data transfer rate of up to 100 Mbps.
- `net0_force10Mb = 1` limits the data transfer rate to 10 Mbps, which may be required for compatibility with older networks.

This setting could be required for compatibility with network infrastructure using older hardware (CPUM cards).

NOTE: For a CPUM / IOCN card pair, the Primary Ethernet Controller (ETH1) can operate at up to 100 Mbps on newer versions of the IOCN card (PNR: 200-566-000-113 and more recent) but is limited to 10 Mbps on older versions of the IOCN card (PNR: 200-566-000-112 and earlier).
The Secondary Ethernet Controller (ETH2) is always limited to 10 Mbps.

Under `#redundant network connection`, `#net1_ipaddr`, `#net1_netmask` and `net1_force10Mb` are the equivalent parameters used to configure the secondary network interface (ETH2) that is available on the front panel of the `IOCN` card, if installed.

3.4.3 CPUM cards running firmware version 074 or earlier

For more information on CPUM card IP addresses, see 4.1 Default IP address parameters.

For a CPUM card running firmware version 074 or earlier, IP addresses are defined in the `hosts` file. So, The same procedure as 3.4.1 CPUM cards running firmware version 075 or later can be used but to access the `hosts` file (rather than the `net.cfg` file).

For example, the IP address of the network interface for the CPUM card is displayed under `#Primary Ethernet Port` as `node1`.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

THIS PAGE INTENTIONALLY LEFT BLANK

4 SETTING UP AN ETHERNET CONNECTION

A direct Ethernet connection is the quickest and simplest method of communicating with a VM600 rack (see Figure 4-1).

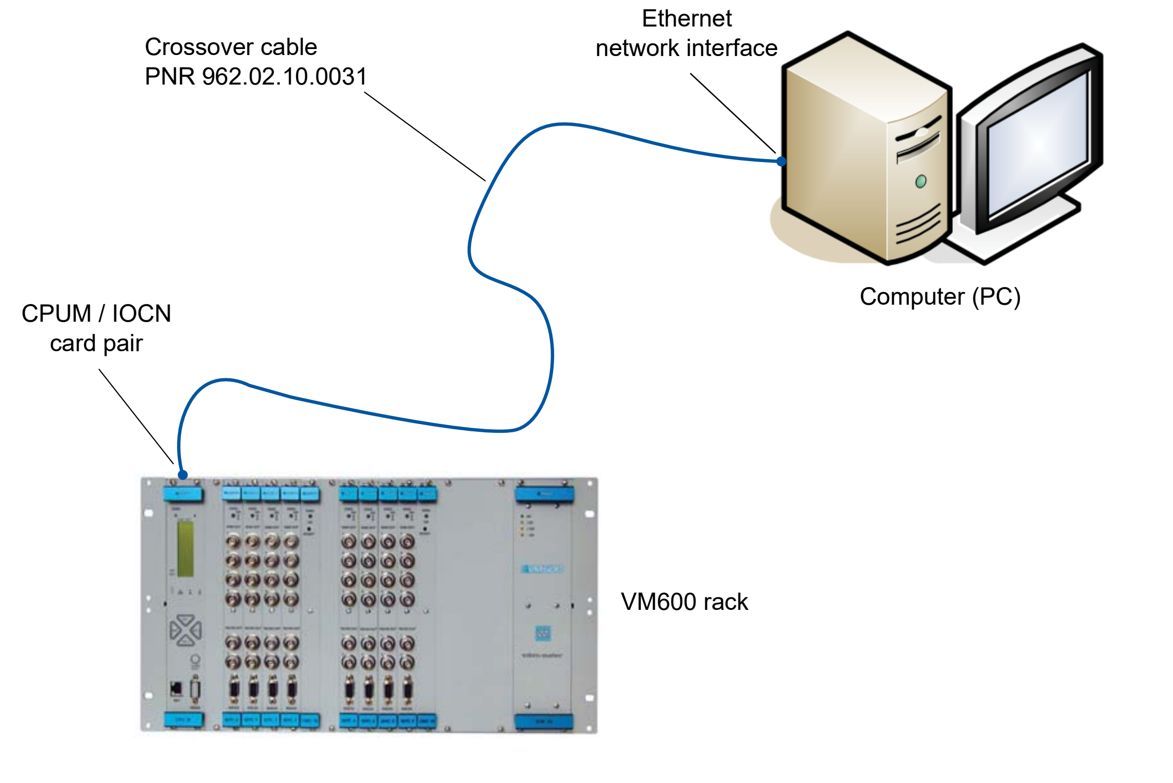


Figure 4-1: Direct connection of a computer to a VM600 rack (without using a LAN)

The exact procedure to be followed will depend on the operating system used. The following description uses the PuTTY terminal emulator on Windows® 7. If you are using a different operating system or terminal emulator that supports Telnet communications protocols, you can still follow this description for guidance, but the dialog boxes and menus shown in screen shots may look quite different.

In case of questions or problems, consult the Windows help or contact your system administrator. If no solution can be found, contact your local Meggitt representative.

4.1 Default IP address parameters

Like any Ethernet (TCP/IP) network compatible device, a CPUM card uses an IP address to uniquely identify it on the network.

NOTE: Static IP addresses must be manually assigned and configured for a CPUM card as the card does not support DHCP (see 1.4.4.2 The dynamic host configuration protocol).

4.1.1 CPUM cards running firmware version 075 or later

The default Ethernet communication parameters factory assigned to a CPUM card running firmware version 075 or later are given in Table 4-1.

Table 4-1: Default IP address parameters for a CPUM card running firmware version 075 or later

IP address	Network mask	Description
10.10.56.56	0xFFFFFF80 (255.255.255.128)	Primary network interface (ETH1). This is the network interface available on either the CPUM card (**NET** connector) or the IOCN card (**1** connector).
10.10.56.156		Secondary network interface (ETH2). This is the network interface available on the IOCN card (**2** connector), if installed.

NOTE: For CPUM cards running firmware version 075, the default IP addresses remain the same but the default network mask has changed, in comparison with firmware version 074 or earlier (see Table 4-2).

NOTE: The primary network interface (ETH1) of a CPUM card can be routed either via the 8P8C connector (**NET**) on the front panel of the CPUM card or via the upper 8P8C connector (**1**) on the front panel of its associated IOCN card (optional). The choice is made using jumpers J42, J40, J52 and J53 on the CPUM card. For example, when all of these jumpers are removed, the primary network interface is routed via the CPUM card.
The secondary network interface (ETH2) of a CPUM card is routed via the lower 8P8C connector (**2**) on the front panel of its associated IOCN card (optional).
Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information.

For a CPUM card running firmware version 075 or later, IP addresses are defined in the `net.cfg` file (see 2 CPUM card directory structure and configuration files). These factory assigned settings can be modified as required.

4.1.2 CPUM cards running firmware version 074 or earlier

The default Ethernet communication parameters factory assigned to a CPUM card running firmware version 074 or later are given in Table 4-2.

Table 4-2: Default IP address parameters for a CPUM card running firmware version 074 or earlier

IP address	Network mask	Description
10.10.56.56	0xFFFF0000 (255.255.0.0)	Primary network interface (ETH1). This is the network interface available on either the CPUM card (**NET** connector) or the IOCN card (**1** connector).
10.10.56.156		Secondary network interface (ETH2). This is the network interface available on the IOCN card (**2** connector), if installed.

For a CPUM card running firmware version 074 or earlier, IP addresses are defined in the `hosts` file (see 2 CPUM card directory structure and configuration files).

Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

4.2 Configuring a computer for Ethernet communications with a VM600 rack

The network adapter (network interface card) of the computer to be used for communication with a VM600 rack must be configured to be compatible with the IP address and subnet mask used by the CPUM card, that is, both devices must be in the same subnet (see 1.4.5 IP addressing).

NOTE: It is quite typical, for a network to use a combination of DHCP assigned, auto IP assigned (dynamic) and/or manually assigned (static) addresses for its attached devices. However, good network administration is necessary to ensure that an IP address that has already been allocated in one way is not reused elsewhere in the network.
Therefore, it is always recommended to contact your IT department or network administrator before manually assigning a static IP address to a VM600 rack (CPUM card) on a network.

Optionally, the `hosts` file of the computer to be used for communication with a VM600 rack can be edited in order that the VM600 rack (CPUM card) can be identified by a hostname in addition to the IP address (see 4.2.3 Editing a computer's hosts file).

4.2.1 Requirements

1- A computer running one of the following Windows operating systems: Windows 98, Windows NT 4.0, Windows XP or Windows 7.

2- A unique IP address for your computer.

3- A VM600 rack containing a CPUM card (and its associated IOCN card (optional)) with at least one Ethernet interface.

4- A unique IP address for the VM600 rack.

5- An Ethernet crossover cable allowing Ethernet communication between the computer a and a CPUM (or IOCN) card.
A suitable Ethernet crossover cable is available from Meggitt SA (PNR 962.02.10.0031).

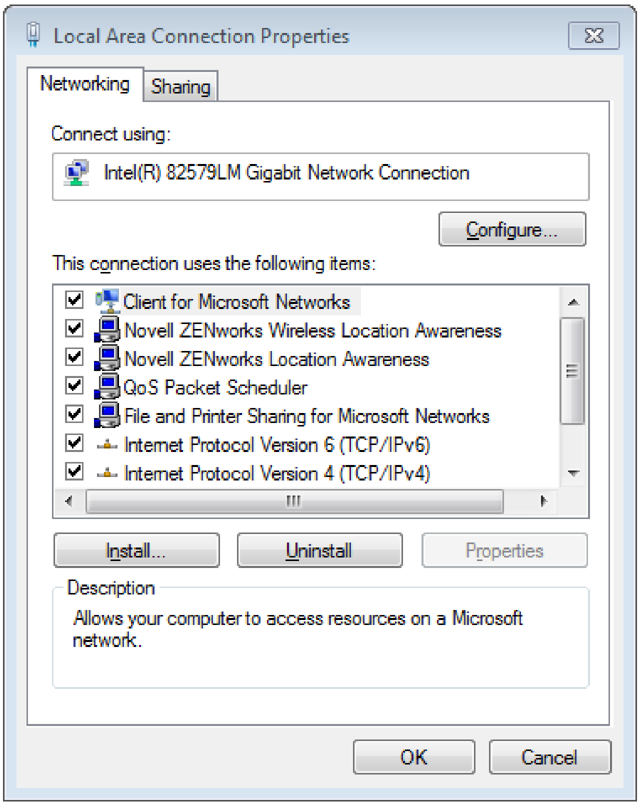
NOTE: The IP address of the VM600 rack (CPUM card) is set in the factory to 10.10.56.56 (see 4.1 Default IP address parameters). This is a default address and it is highly recommended to change it.
To change the address for a CPUM card running firmware version 075 or later, edit the `net.cfg` file stored on the CPUM card (see 3.4.2 About the net.cfg file).
To change the address for a CPUM card running firmware version 074 or earlier, edit the `hosts` file stored on the CPUM card, (see 3.4.3 CPUM cards running firmware version 074 or earlier).

4.2.2 Configuring a computer's network adapter

1- Display the properties for the network adapter on the computer. For example:

- On a Windows 7 computer, from the Windows Start menu, click **Control Panel > Network and Sharing Center**. In the Network and Sharing Center window that appears, click **Change adapter settings** to view the Network Connections, then right-click the Local Area Connection icon and click Properties.
- On a Windows XP computer, from the Windows Start menu, click **Control Panel** > **Network and Internet Connections** > **Network Connections**. Then right-click on the Local Area Connection icon and select **Properties**.

The Local Area Connection Properties dialog box should appear, as shown in Figure 4-2.



Local Area Connection Properties

Connect using:
Intel(R) 82579LM Gigabit Network Connection

This connection uses the following items:

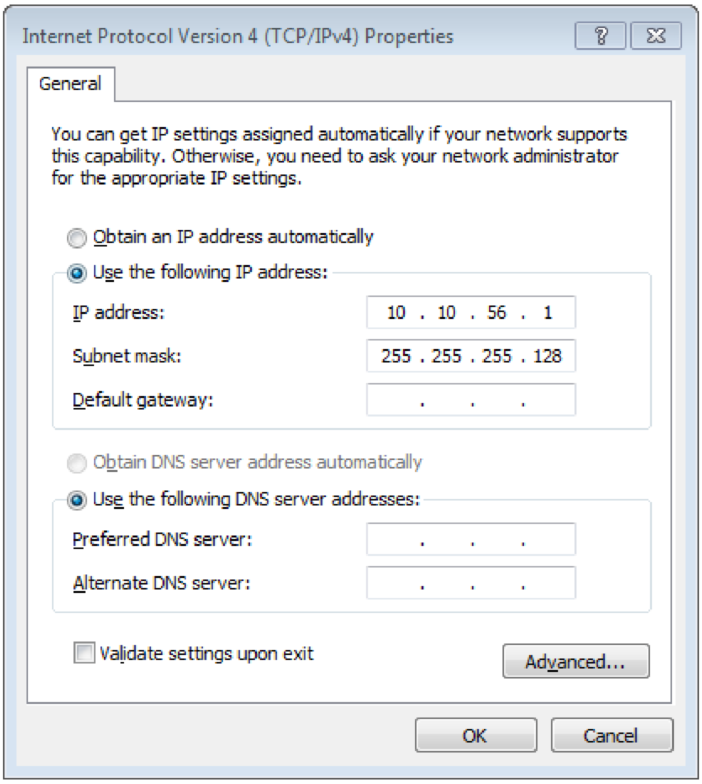
☑ Client for Microsoft Networks
☑ Novell ZENworks Wireless Location Awareness
☑ Novell ZENworks Location Awareness
☑ QoS Packet Scheduler
☑ File and Printer Sharing for Microsoft Networks
☑ Internet Protocol Version 6 (TCP/IPv6)
☑ Internet Protocol Version 4 (TCP/IPv4)

Description
Allows your computer to access resources on a Microsoft network.

Figure 4-2: Windows Local Area Connections dialog box

2- Select the **Internet Protocol Version 4 (TCP/IPv4)** item, then click the **Properties** button.

The Internet Protocol Version 4 (TCP/IPv4) Properties dialog box should appear, as shown in Figure 4-3.



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 10 . 10 . 56 . 1

Subnet mask: 255 . 255 . 255 . 128

Default gateway: . . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

□ Validate settings upon exit

Figure 4-3: Windows Internet Protocol Version 4 (TCP/IPv4) Properties dialog box

3- On the **General** tab, select **Use the following IP address**, then enter the following information for this network interface of the computer:

Under **IP address**, enter the IP address in dot-decimal notation.

NOTE: Contact your IT department or network administrator to ensure that the static IP address to be used is not already used elsewhere on the network.

Under **Subnet mask**, enter the subnet mask in dot-decimal notation.

Netmasks can be written in hexadecimal or dot-decimal notation (for example, 0xFFFFFF80 or 255.255.255.128). The Windows Internet Protocol Version 4 (TCP/IPv4) Properties dialog box requires netmasks in dot-decimal.

Under **Default gateway**, enter the default gateway in dot-decimal notation, if applicable.

Click **OK** to close the Windows Internet Protocol Version 4 (TCP/IPv4) Properties dialog box, then click **Close** to close the Windows Local Area Connections dialog box apply and apply the IP address settings to the network adapter.

4.2.3 Editing a computer's hosts file

A hostname is a label assigned to a device connected to a computer network that is used to identify the device. The computer's `hosts` file is used to map the IP address of a device to a hostname, so that when the hostname is used, the computer can refer to the `hosts` file and resolve the hostname to its IP address.

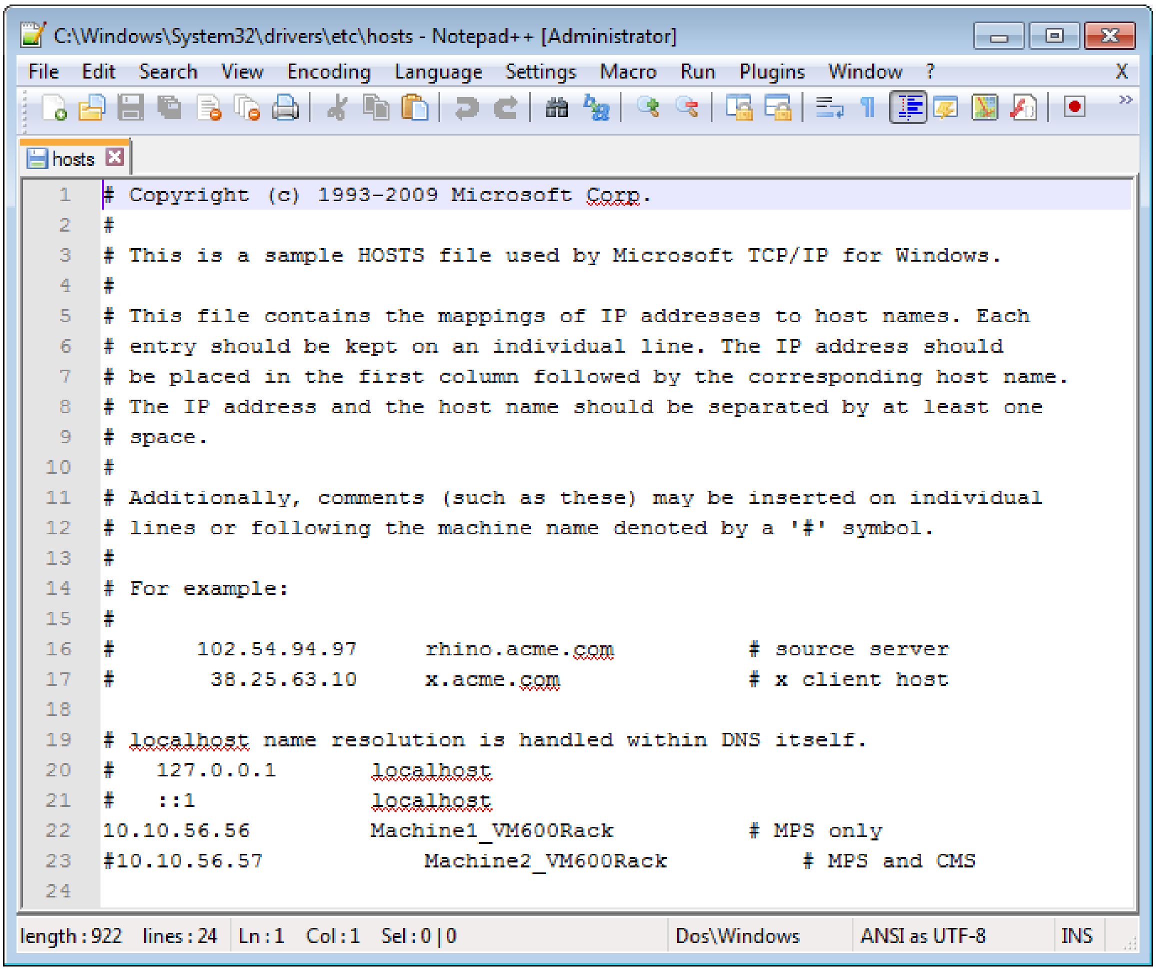
Using hostnames is optional but it helps in the management of systems. For example, it makes accessing a network device more easy, compared to remembering and using IP addresses. Mapping IP addresses to hostnames can also help to resolve DNS issues.

1- Open the (local) `hosts` file on the computer's hard disk using a text editor such as Notepad++ or a similar editor. The `hosts` file is found in the Windows operating system area of the computer. For example: If Windows XP is the operating system used, the file can be found here:

   On a Windows XP computer, the default location for the `hosts` file is
   `C:\WINNT\system32\drivers\etc`

   On a Windows 7 computer, the default location for the `hosts` file is
   `C:\Windows\System32\drivers\etc`

2- As shown in Figure 4-4, the `hosts` file lists the IP address of a device followed by its hostname, separated by one or more whitespace characters, on a single line.



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost
10.10.56.56         Machine1_VM600Rack          # MPS only
#10.10.56.57            Machine2_VM600Rack          # MPS and CMS

```

Figure 4-4: Example Windows `hosts` file

For example, the hosts file of Figure 4-4 includes hostnames for two VM600 racks:

- Machine1_VM600Rack with the IP address 10.10.56.56
- #Machine2_VM600Rack with the IP address 10.10.56.57 (currently commented out).

NOTE: # is used for line comments and can be used to temporarily "comment out" a line until it is required in the future.

3- Edit the hosts file as required, adding a new line, after the localhost entries, for each VM600 rack with its IP address (CPUM card) and a suitable hostname.

NOTE: The IP address of the VM600 rack (CPUM card) is not actually defined in this (local) computer "hosts" file, but in a different file on the CPUM card's memory (CompactFlash or DiskOnChip).
See 2 CPUM card directory structure and configuration files.

4- After editing, save the hosts file as a text file without any file name extension (that is, do not accidently use a *.txt or any other file name extension).

NOTE: The computer must be restarted for any changes to the hosts file to take effect (that is, be registered by Windows).

For an example of how hostnames can be used, see 4.4 Using VM600 software with a networked rack.

4.2.4 Testing communications

Ping is a computer network administration utility used to test the reachability of a host on an internet protocol (IP) network, that is, it checks that a device is connected to the same subnet as a computer. It also reports the round-trip time for messages sent from the originating host to a destination device.

The ping program can be run from a Windows command prompt. To test the communications link from a computer to a CPUM card:

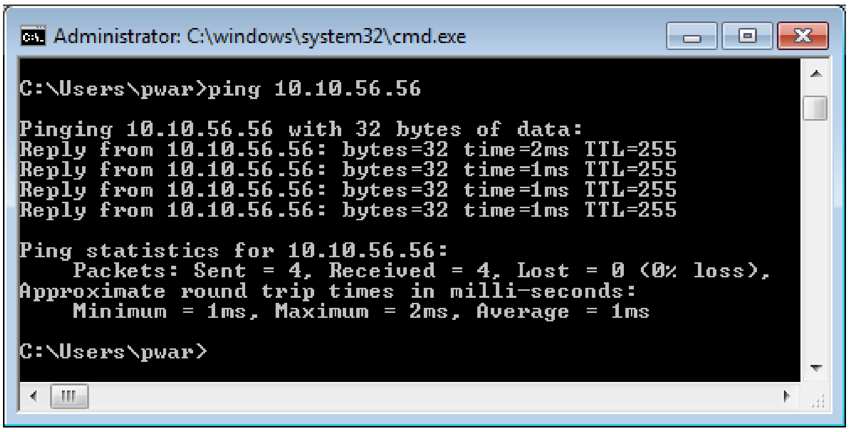
1- Start a command prompt window in one of the following ways:

Click **Start > Programs > Accessories > Command Prompt**.

Or click **Start > Run**, then type cmd in the Open text box and then click **OK**.

2- At the command prompt, type ping xxx.xxx.xxx.xxx, where xxx.xxx.xxx.xxx is replaced with the IP address of the device being checked. For example, a VM600 rack (CPUM card) at 10.10.56.56 and on the same network as the computer.

3- As shown in Figure 4-4, if everything is configured correctly, the device replies, which indicates that it is in the same subnet as your computer and is ready to be used.



```
C:\Users\pwar>ping 10.10.56.56

Pinging 10.10.56.56 with 32 bytes of data:
Reply from 10.10.56.56: bytes=32 time=2ms TTL=255
Reply from 10.10.56.56: bytes=32 time=1ms TTL=255
Reply from 10.10.56.56: bytes=32 time=1ms TTL=255
Reply from 10.10.56.56: bytes=32 time=1ms TTL=255

Ping statistics for 10.10.56.56:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\Users\pwar>
```

Figure 4-5: Pinging a VM600 rack (CPUM card)

NOTE: To be able to communicate with a device, the computer and the device must be connected to the same network and must have IP addresses belonging to the same subnet.

The ping program is also included in the CPUM firmware, so the `ping` command can be run in a terminal emulation session in order to test the communications link from a CPUM card to another network device, if required.

The CTRL+C shortcut key can be used to stop a ping command in a terminal emulation session.

4.3 Ethernet redundancy

CPUM cards are available in different versions, including standard, Ethernet redundant and serial redundant.

NOTE: Ethernet redundancy is only possible with Ethernet redundant versions of the CPUM card, that is, with cards equipped two Ethernet interfaces.
Refer to the CPUM *and IOCN card pair data sheet* for additional information.

For a redundant Ethernet configuration, two different IP addresses are attributed to a single CPUM / IOCN card pair. These addresses must be chosen carefully and they must follow the following two principles:

1- Both of the IP addresses of the CPUM must be defined in different subnets. For more information about subnets, see 1.4.5.3 Subnets and subnet masks.

2- A host computer which communicates with one of the CPUM addresses must have its IP address in the same subnet as the corresponding CPUM address.

As described in 1.4.5 IP addressing, an IP address is divided into a subnet (network) part and a host part. The subnet range is defined by the netmask.

Table 4-3 presents some typical netmask values and corresponding IP addresses in common use.

Table 4-3: Examples of netmasks and corresponding IP addresses

Netmask		IP address
Hexadecimal	Dot-decimal	
0xFF000000	255.0.0.0	nn.hh.hh.hh
0xFFFF0000	255.255.0.0	nn.nn.hh.hh
0xFFFFFF00	255.255.255.0	nn.nn.nn.hh
0xFFFFFF80	255.255.255.128	nn.nn.nn.0-127/128-255 (n=subnetwork, h=host)

Table 4-4 presents some valid IP addresses and netmask values for an Ethernet redundant version of the CPUM card and the two host computers.

Table 4-4: Examples of netmasks and corresponding valid IP addresses for an Ethernet redundant version of the CPUM card and a host computer

	Netmask	IP addresses	
		CPUM card (and network interface)	Host computer
Example 1	0xFFFFFF80 (255.255.255.128)	10.10.56.56 (primary) 10.10.56.156 (secondary)	10.10.56.1 10.10.56.129
Example 2	0xFFFFFF00 (255.255.255.0)	10.10.56.56 (primary) 10.10.57.56 (secondary)	10.10.56.1 10.10.57.1

4.3.1 Changing the IP addresses of the rack

As described in Table 4-3 and Table 4-4, consider a subnet nn.nn.nn.hh with the following netmask and IP addresses:

- IP address for the primary network interface (ETH1): 10.10.56.56
- IP address for the secondary network interface (ETH2): 10.10.56.156
- Netmask: 0xFFFFFF80

4.3.1.1 CPUM cards running firmware version 075 or later

For a CPUM card running firmware version 075 or later, the network interface (IP addressing) information is defined in the `net.cfg` file (see 2 CPUM card directory structure and configuration files).

To modify the IP addresses of the CPUM, the `net.cfg` file must be edited, for example, using the CPUM Configurator software (see 6 CPUM Configurator and 6.8 Changing the IP address of a VM600 rack (CPUM card)) or a terminal emulation session (see 7 Terminal emulation and 7.10 Changing the IP address of a VM600 rack (CPUM card)).

As shown in Figure 4-6 (top), the parameters displayed in the CPUM Configurator's **CPUM TCP-IP Configuration** dialog box correspond to the CPUM card's Ethernet interfaces as follows:

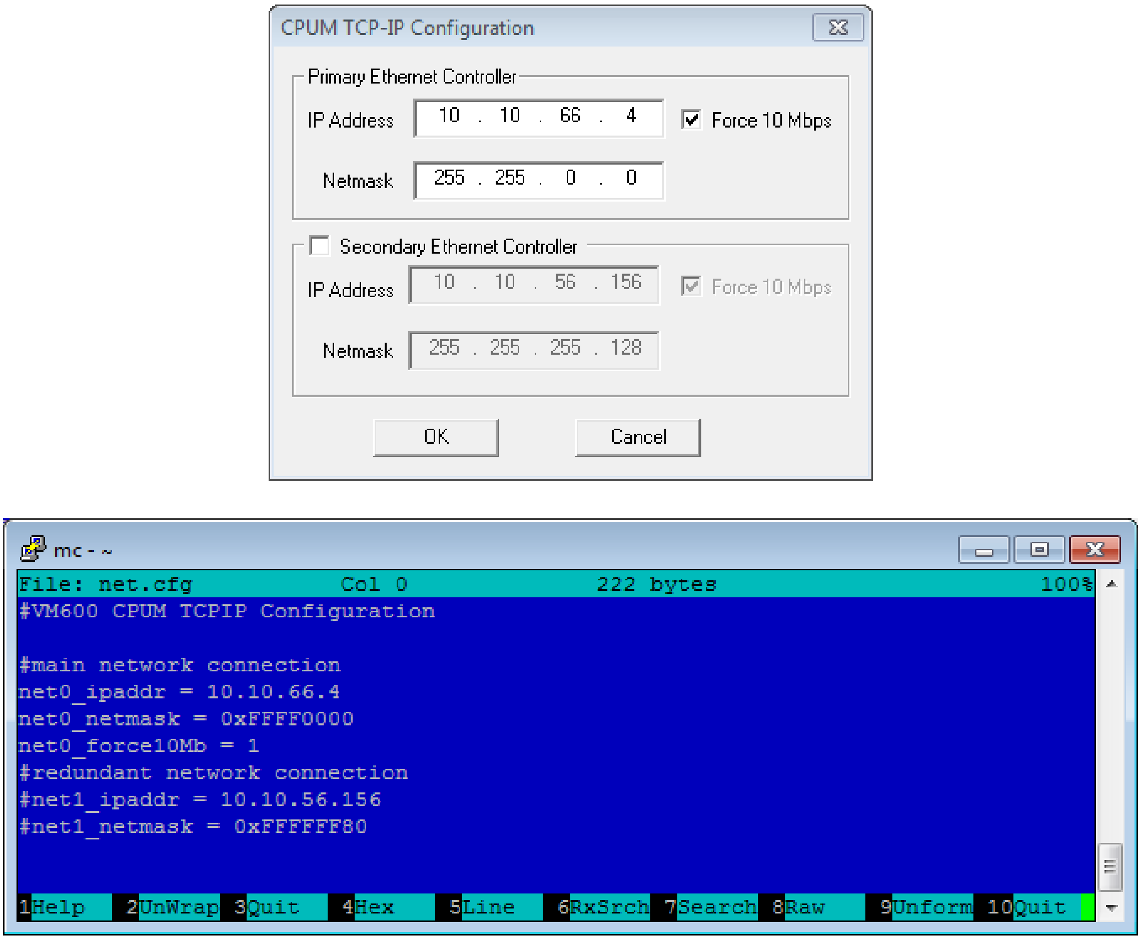
- `Primary Ethernet Controller` for the primary network interface (ETH1).
- `Secondary Ethernet Controller` for the secondary network interface (ETH2).

Editing the values in CPUM Configurator changes the values in the underlying `net.cfg` file.

As shown in Figure 4-6 (bottom), the parameters displayed in the `net.cfg` file correspond to the CPUM card's Ethernet interfaces as follows:

- `#main network connection` and `net0_`... for the primary network interface (ETH1).
- `#redundant network connection` and `net1_`... for the secondary network interface (ETH2).

For additional information on the `net.cfg` file, see 3.4.2- About the net.cfg file.



CPUM TCP-IP Configuration

Primary Ethernet Controller

IP Address: 10 . 10 . 66 . 4 ☑ Force 10 Mbps

Netmask: 255 . 255 . 0 . 0

□ Secondary Ethernet Controller

IP Address: 10 . 10 . 56 . 156 ☑ Force 10 Mbps

Netmask: 255 . 255 . 255 . 128

```
File: net.cfg              Col 0                222 bytes                  100%
#VM600 CPUM TCPIP Configuration

#main network connection
net0_ipaddr = 10.10.66.4
net0_netmask = 0xFFFF0000
net0_force10Mb = 1
#redundant network connection
#net1_ipaddr = 10.10.56.156
#net1_netmask = 0xFFFFFF80
```

1Help 2UnWrap 3Quit 4Hex 5Line 6RxSrch 7Search 8Raw 9Unform 10Quit

Figure 4-6: Network interface (IP addressing) information from the `net.cfg` file of a CPUM card displayed in CPUM Configurator (top) and in Midnight Commander (bottom)

Then complete the procedure by restarting the rack: turn off the VM600 rack, wait for 2 to 3 seconds, and turn on the VM600 rack. If you are unable to physically power down the rack, you can use the shutdown command instead.

After the CPUM card is correctly configured, the IP addresses of the two computers connected to the two Ethernet ports must be set as described in Section 4.2.2 Configuring a computer’s network adapter.

NOTE: Each computer’s IP address must be chosen in the same subnet as the corresponding CPUM card’s IP address.

For example, the first computer IP address corresponding to the primary CPUM address (10.10.56.56) can be set to 10.10.56.55 and the second one set to 10.10.57.55.

4.3.1.2 CPUM cards running firmware version 074 or earlier

For a CPUM card running firmware version 074 or earlier, the network interface (IP addressing) information is defined in the `hosts` file and the netmask is defined in the `tcpip.1` file (see 2 CPUM card directory structure and configuration files).

To modify the IP addresses of the CPUM, the `hosts` and `tcpip.1` files must be edited, for example, using a terminal emulation session (see 7 Terminal emulation).

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

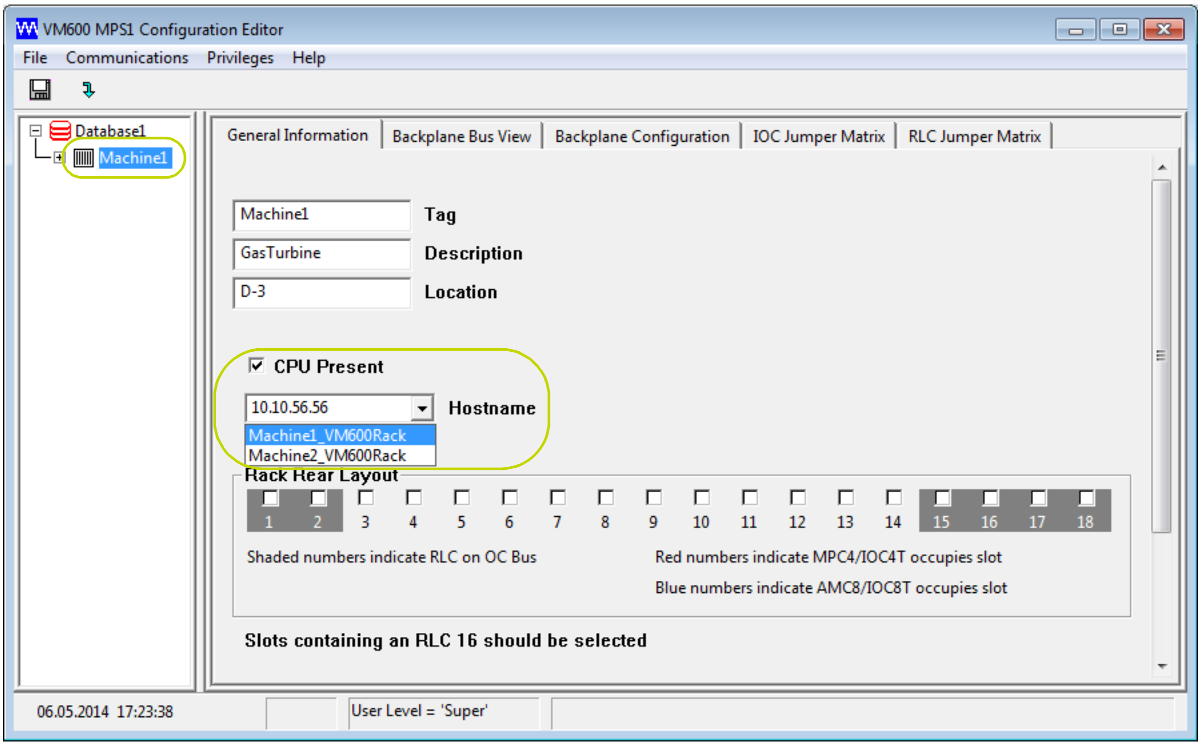
4.4 Using VM600 software with a networked rack

4.4.1 VM600 MPS software

1- Start the VM600 MPS software.

NOTE: Refer to a *VM600 MPSx configuration software for machinery protections systems software manual* for additional information.

2- In the tree structure (left), select a VM600 rack as shown in Figure 4-7.



Machine1 Tag

GasTurbine Description

D-3 Location

☑ CPU Present

10.10.56.56 Hostname

Machine1_VM600Rack
Machine2_VM600Rack

Shaded numbers indicate RLC on OC Bus

Red numbers indicate MPC4/IOC4T occupies slot

Blue numbers indicate AMC8/IOC8T occupies slot

Slots containing an RLC 16 should be selected

06.05.2014 17:23:38 User Level = 'Super'

Figure 4-7: Using hostnames in the VM600 MPS software

3- In the main window (centre), on the **General Information** tab:
Select the **CPU Present** check box.

For the **Hostname**, use the drop-down combo box to select the hostname corresponding to the VM600 rack, that was defined in the Windows hosts file (see 4.2.3 Editing a computer's hosts file).

Alternatively, the IP address of the VM600 rack (for example, 10.10.56.56) can be used by entering it directly in the **Hostname** field.

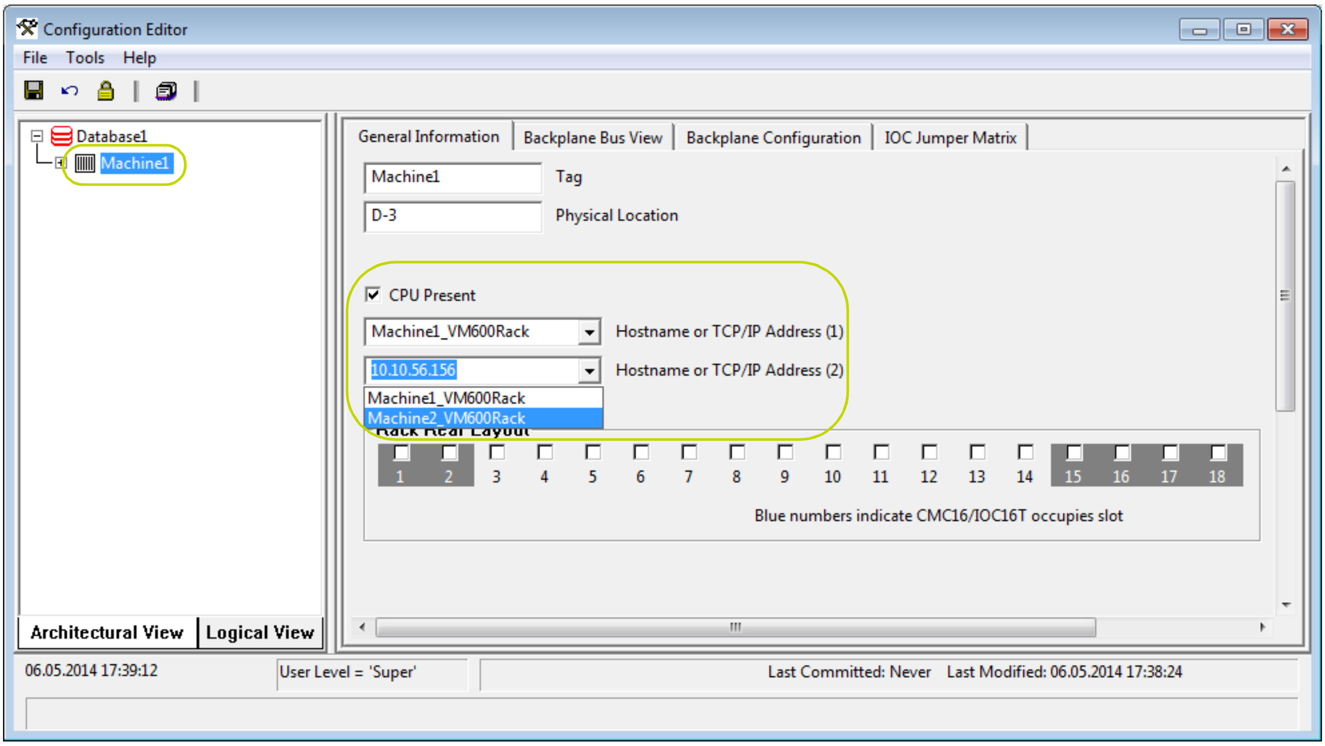
4- Communication between the VM600 MPS software and the VM600 rack over Ethernet should now be possible.

4.4.2 VM600 CMS software

1- Start the VM600 CMS software.

NOTE: Refer to a *VM600 CMS software for machinery protections systems software manual* for additional information.

2- In the **Architectural View** tree structure (left), select a VM600 rack as shown in Figure 4-8.



Tag: Machine1
Physical Location: D-3
☑ CPU Present
Hostname or TCP/IP Address (1): Machine1_VM600Rack
Hostname or TCP/IP Address (2): 10.10.56.156
Machine1_VM600Rack
Machine2_VM600Rack

06.05.2014 17:39:12 User Level = 'Super' Last Committed: Never Last Modified: 06.05.2014 17:38:24

Figure 4-8: Using hostnames in the VM600 CMS software

3- In the main window (centre), on the **General Information** tab:

Select the **CPU Present** check box.

For the **Hostname or TCP/IP address**, use the drop-down combo box to select the hostname corresponding to the VM600 rack, that was defined in the Windows hosts file (see 4.2.3 Editing a computer's hosts file).

Alternatively, the IP address of the VM600 rack (for example, 10.10.56.56) can be used by entering it directly in the **Hostname or TCP/IP address** field.

4- Communication between the VM600 CMS software and the VM600 rack over Ethernet should now be possible.

THIS PAGE INTENTIONALLY LEFT BLANK

5 ETHERNET COMMUNICATIONS WITH A VM600 RACK

There are many different ways of communicating with and configuring networked VM600 racks.

NOTE: Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information on networked racks and stand-alone racks.

Historically, VM600 racks were managed using a terminal emulator program (for example, PuTTY or HyperTerminal) running on a computer to establish communications with the VM600 rack, then running the vi text editor provided by the CPUM card's firmware to edit the configuration files as necessary.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information on using HyperTerminal and the vi text editor to manage networked VM600 racks.

However, the management of networked VM600 racks can be more easily achieved using Ethernet-based communications such as:

- The CPUM Configurator software running on a computer (see 6 CPUM Configurator).
- A terminal emulation session on a computer running Midnight Commander on the CPUM card (see 7 Terminal emulation).

However, before Ethernet-based communications can be used, the IP address of the CPUM card in the networked VM600 rack must be known. This information can be obtained directly from the CPUM card using a serial communications link (see 3 Serial communications with a VM600 rack).

NOTE: In order to use the CPUM Configurator software or a Telnet connection to a VM600 rack, the IP address of the CPUM card must be known.
If the IP address of the CPUM card is not known, then a serial (RS-232) connection to the card can be used in order to discover the card's IP address.
See 3.4 Discovering the IP address of a CPUM card.

In addition, the computer being used to communicate with the VM600 rack (CPUM card) must be configured correctly in order to allow Ethernet communications with the CPUM card. This means that the network adapter (network interface card) of the computer must be configured to be compatible with the IP address and subnet mask used by the CPUM card, that is, both devices must be in the same subnet. See 4 Setting up an Ethernet connection for additional information.

5.1 Establishing Ethernet communications with a CPUM card

The exact procedure to be followed will depend on the operating system and software tools used. The following description uses the PuTTY terminal emulator on Windows® 7. If you are using a different operating system or terminal emulator that supports the Telnet protocol, you can still follow this description for guidance.

In case of questions or problems, consult the Windows help or contact your system administrator. If no solution can be found, contact your local Meggitt representative.

NOTE: If your CPUM card does not support Ethernet communications, a PPP connection can be used to communicate with the rack for cards running firmware version 072 or earlier. Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information on PPP.

For a direct connection between a VM600 rack and a computer, use a crossover Ethernet cable between the 8P8C connector (**NET**) on the front panel of the CPUM card or the upper 8P8C connector (**1**) on the front panel of the IOCN card and the Ethernet port of the computer (not a standard cable).

NOTE: The primary network interface (ETH1) of a CPUM card can be routed either via the 8P8C connector (**NET**) on the front panel of the CPUM card or via the upper 8P8C connector (**1**) on the front panel of its associated IOCN card (optional). The choice is made using jumpers J42, J40, J52 and J53 on the CPUM card. For example, when all of these jumpers are removed, the primary network interface is routed via the CPUM card.
The secondary network interface (ETH2) of a CPUM card is routed via the lower 8P8C connector (**2**) on the front panel of its associated IOCN card (optional).
Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information.

For an indirect connection between a CPUM card and a computer via a network (using network switches), use standard Ethernet cables (not crossover cables).

See 4 Setting up an Ethernet connection.

5.2 Using an Ethernet connection to check the IP address of a CPUM card

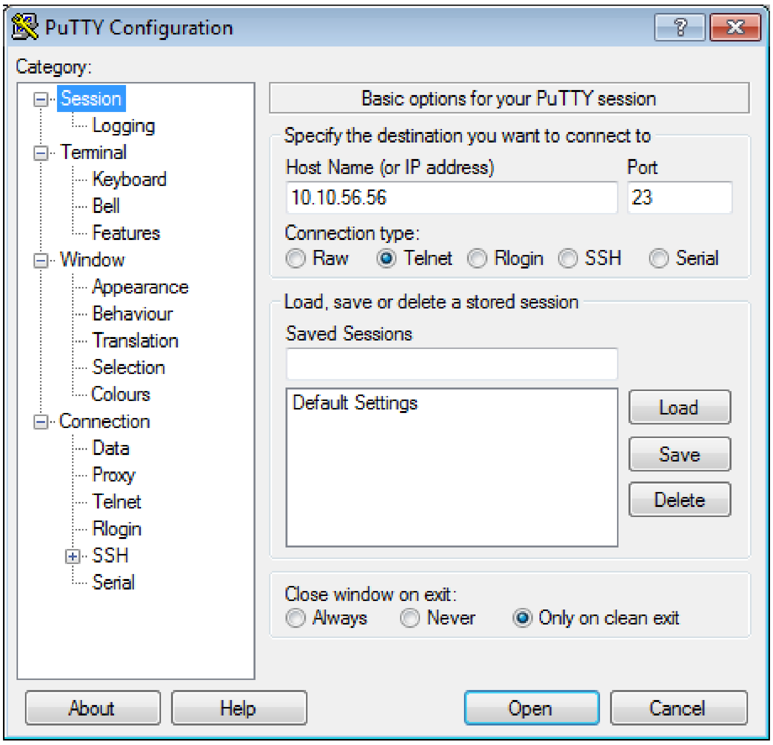
If the IP address of a CPUM card is not known, it can always be discovered using a serial connection to the RS-232 connector on the front panel of the CPUM card and terminal emulation software (see 3.4 Discovering the IP address of a CPUM card).

NOTE: The directory structure and configuration files, depend on the version of firmware running on a CPUM card (see 2 CPUM card directory structure and configuration files). For a CPUM card running firmware version 075 or later, the IP address information is given in the `net.cfg` file.

5.2.1 CPUM cards running firmware version 075 or later

1- Connect a VM600 rack (CPUM card) and a computer using the appropriate Ethernet cable.

2- Start PuTTY, select a **Telnet** connection and enter the IP address (**Host Name (or IP address)**) of the CPUM card, as shown in Figure 5-1. (Telnet always uses TCP port number 23 (**Port**).)



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 10.10.56.56

Port: 23

Connection type: ○ Raw ● Telnet ○ Rlogin ○ SSH ○ Serial

Load, save or delete a stored session

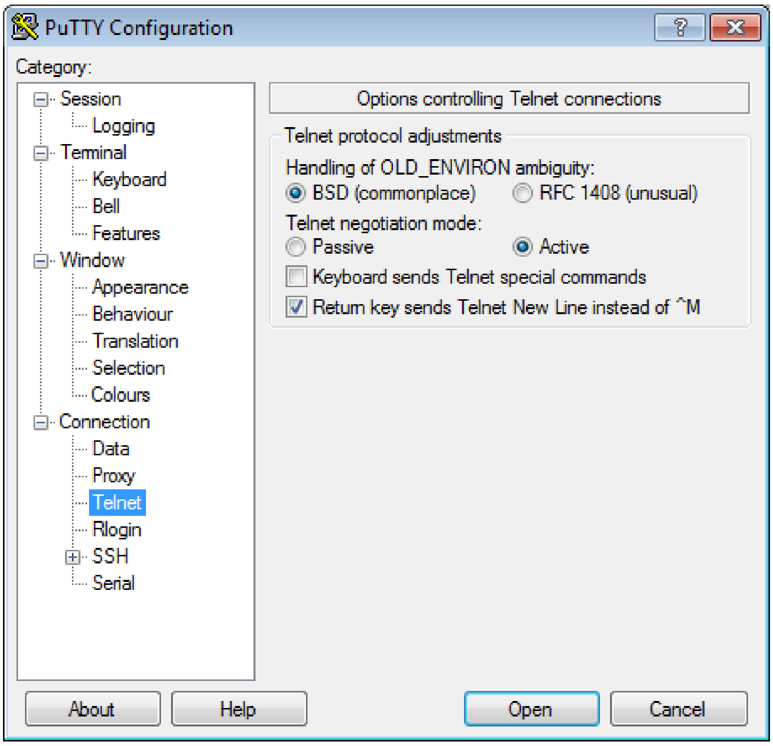
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 5-1: Using PuTTY terminal emulation software – selecting and configuring a Telnet connection

3- Accept the default PuTTY Telnet session options, as shown in Figure 5-2.



Options controlling Telnet connections

Telnet protocol adjustments

Handling of OLD_ENVIRON ambiguity: ● BSD (commonplace) ○ RFC 1408 (unusual)

Telnet negotiation mode: ○ Passive ● Active

□ Keyboard sends Telnet special commands

☑ Return key sends Telnet New Line instead of ^M

Figure 5-2: Using PuTTY terminal emulation software – default Telnet options

4- Click **Open** to start the PuTTY session, then press the ENTER key.

5- When prompted for a login, type **user** (the default login), then press ENTER to continue. For example:

```
login: user
```

When prompted for a password, type **config** (the default password), then press ENTER to continue. For example:

```
password: config
```

6- At the command prompt (`/usr/user #`), type **ls**, then press ENTER to continue. For example:

```
/usr/user # ls
```

The list command lists the files in the current working directory (`/usr/user`) are listed, including the `version.txt` file which contains the firmware version information for a CPUM card.

7- At the command prompt (`/usr/user #`), type **more net.cfg**, then press ENTER to continue. For example:

```
/usr/user # more net.cfg
```

The `more` command displays the contents of the specified text file one screen at a time. If required, pressing ENTER or SPACEBAR displays the next *n* lines of text.

Alternatively, the `cat` command can be used to display the contents of a file without pausing.

8- In the `net.cfg` file, the IP addresses of the network interfaces for the CPUM card are displayed under `#main network connection` as `net0_ipaddr` (for the primary network interface (ETH1)) and under `#redundant network connection` as `net1_ipaddr` (for the secondary network interface (ETH2)).

See 3.4.2 About the net.cfg file for additional information about the `net.cfg` file.

5.2.2 CPUM cards running firmware version 074 or earlier

For a CPUM card running firmware version 074 or earlier, IP addresses are defined in the `hosts` file. So, the same procedure as 5.2.1 CPUM cards running firmware version 075 or later can be used but accessing the `hosts` file (rather than the `net.cfg` file).

For example, the IP address of the network interface for the CPUM card is displayed under `#Primary Ethernet Port` as `node1`.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

5.3 Using an Ethernet connection to identify the firmware version of a CPUM card

It is important to know the version of the firmware running on a CPUM card as the networking features, software compatibility, and configuration files and structure can be different, depending on the firmware version.

For any CPUM card, the firmware version is given in the `version.txt` file.

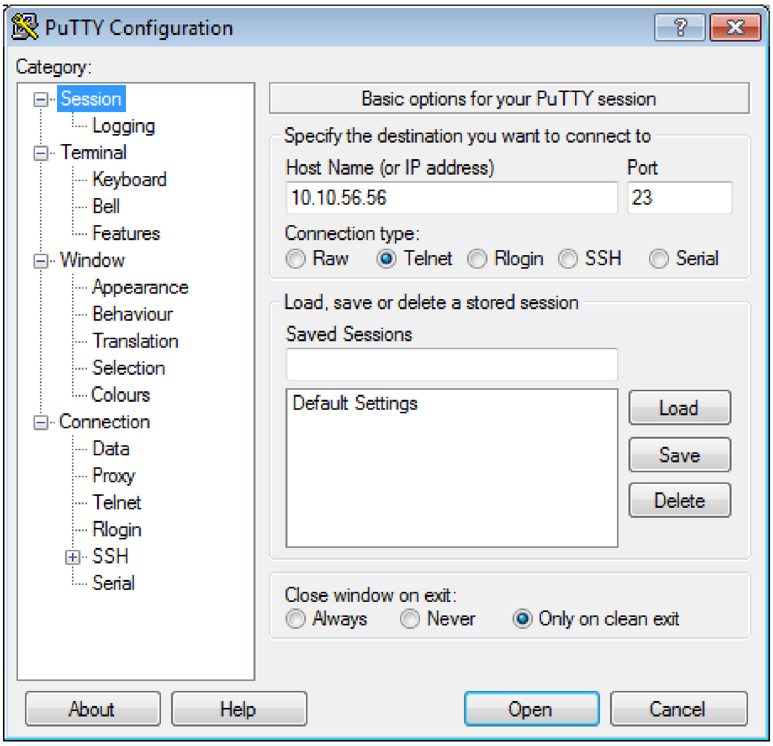
The version of the firmware running on a CPUM card can be identified using either a serial connection or an Ethernet connection to the CPUM card and terminal emulation software.

For information on establishing a serial connection to a CPUM card, see 3.2 Establishing serial communications with a CPUM card.

For information on establishing an Ethernet connection to a CPUM card, see below.

1- Connect a VM600 rack (CPUM card) and a computer using the appropriate Ethernet cable.

2- Start PuTTY, select a **Telnet** connection and enter the IP address (**Host Name (or IP address)**) of the CPUM card, as shown in Figure 5-3. (Telnet always uses TCP port number 23 (**Port**).)



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 10.10.56.56

Port: 23

Connection type: ○ Raw ● Telnet ○ Rlogin ○ SSH ○ Serial

Load, save or delete a stored session

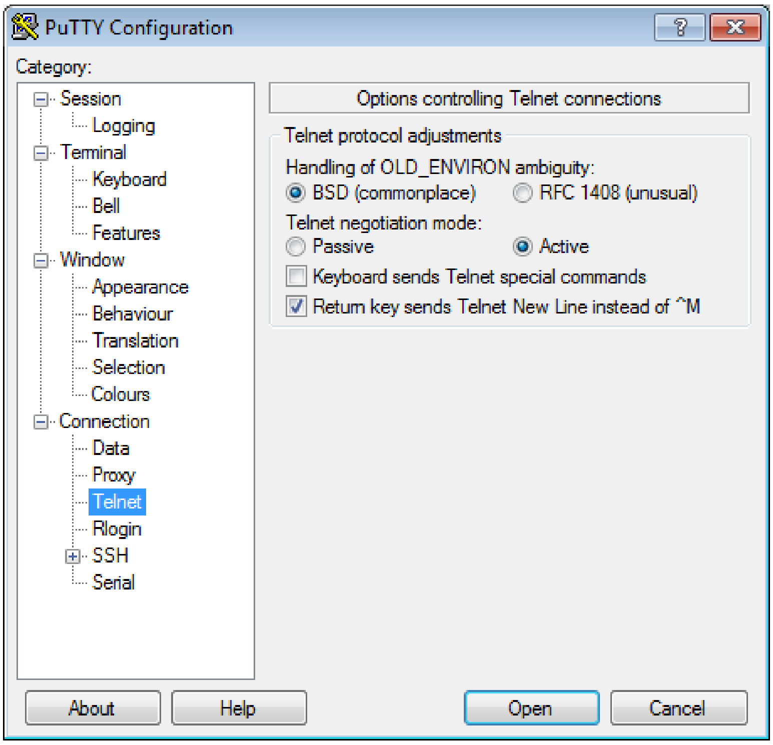
Saved Sessions:

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 5-3: Using PuTTY terminal emulation software – selecting and configuring a Telnet connection

3- Accept the default PuTTY Telnet session options, as shown in Figure 5-4.



Options controlling Telnet connections

Telnet protocol adjustments

Handling of OLD_ENVIRON ambiguity:
● BSD (commonplace) ○ RFC 1408 (unusual)

Telnet negotiation mode:
○ Passive ● Active

□ Keyboard sends Telnet special commands

☑ Return key sends Telnet New Line instead of ^M

Figure 5-4: Using PuTTY terminal emulation software – default Telnet options

4- Click **Open** to start the PuTTY session, then press the ENTER key.

5- When prompted for a login, type **user** (the default login), then press ENTER to continue. For example:

```
login: user
```

When prompted for a password, type **config** (the default password), then press ENTER to continue. For example:

```
password: config
```

6- At the command prompt (`/usr/user #`), type **ls**, then press ENTER to continue. For example:

```
/usr/user # ls
```

The list command lists the files in the current working directory (`/usr/user`) are listed, including the `version.txt` file which contains the firmware version information for a CPUM card.

7- At the command prompt (`/usr/user #`), type **more version.txt**, then press ENTER to continue. For example:

```
/usr/user # more version.txt
```

The more command displays the contents of the specified text file one screen at a time. If required, pressing ENTER or SPACEBAR displays the next *n* lines of text.

See 3.3.1 About the version.txt file for additional information about the `version.txt` file.

5.4 Editing configuration files

See 2.5 Working with configuration files.

5.5 Gateways

In general terms, a gateway is required to allow communication between two different networks or subnets (see 1.4.2 Connectivity within networks).

In the example shown in Figure 5-5, System 1 can communicate directly with System 2, as both have an IP address beginning with 10.10. However, System 1 cannot communicate directly with System 3 as the IP addresses are too different. A gateway is therefore needed.

System 1 can communicate with the gateway as both have an IP address beginning with 10.10. Likewise, System 3 can communicate with the gateway as both have an IP address beginning with 194.11.227.

NOTE: A gateway must belong to the same subnet as the network for which it is acting as a go-between, that is, the same subnet mask applies.

Note that if the subnet mask for subnet "A" was changed from 255.255.0.0 to 255.255.255.0, communication would no longer be possible between System 1 and System 2 or between System 1 and the gateway.

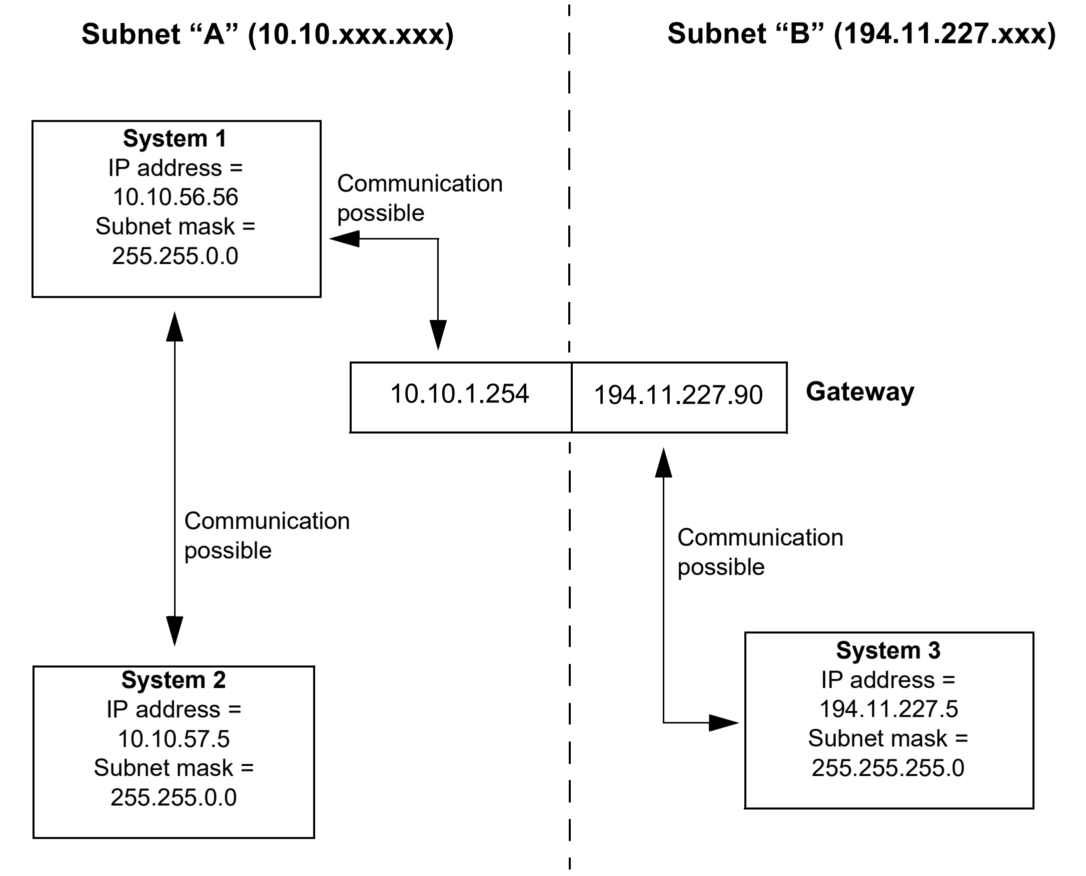


Figure 5-5: Communication between two subnets using a gateway

For a VM600 rack, Ethernet communication between the host computer and the VM600 rack can only take place if the IP addresses of the host computer and the VM600 rack (CPUM card) belong to the same subnet.

To get around this potential problem, a default gateway can be defined for a CPUM card.

NOTE: For proper operation, the IP addresses of the gateway and the CPUM must belong to the same subnet. If this is not the case, the gateway cannot be accessed by the CPUM.

In addition, the host computer and the gateway must also have IP addresses that are visible to each other.

5.6 Adding a default gateway to a CPUM card

5.6.1 CPUM cards running firmware version 075 or later

For a CPUM card running firmware version 075 or later, the default gateway for a CPUM card is defined and added using the `sysinit` file.

5.6.2 About the sysinit file

The `sysinit` file is a system initialisation file that contains commands to set up various device drivers and services for the CPUM card.

To add a default gateway to a CPUM card, a command must be added to the `Configure TCP/IP` section of the `sysinit` file, as shown in 5.6.2.1 Example sysinit file.

The path to the `sysinit` file is `/etc/system/sysinit`

5.6.2.1 Example sysinit file

The `Configure TCP/IP` section of an example `sysinit` file without a default gateway:

```
#--------------------------------
#
# Configure TCP/IP
startnet
```

The `Configure TCP/IP` section an example `sysinit` file with a default gateway:

```
#--------------------------------
#
# Configure TCP/IP
startnet
route add default 10.10.1.254
```

5.6.2.2 Description

To add a default gateway to a CPUM card, the `route` command is used. For example:

```
route add default 10.10.1.254
```

Where:

`route add default` is the command to add the default gateway.

`10.10.1.254` is the IP address of the default gateway, in dot-decimal notation.

As described in 2.5 Working with configuration files, the easiest way to edit the `sysinit` file is a terminal emulation session on a computer running Midnight Commander on the CPUM card (see 7 Terminal emulation).

NOTE: As `sysinit` is a system initialisation file, it is read its commands are run when the CPUM card starts, so the card must be restarted in order for any changes to this file to take effect.

5.6.3 CPUM cards running firmware version 074 or earlier

For a CPUM card running firmware version 074 or earlier, the default gateway for a CPUM card is defined and added using the `hosts` and `tcpip.1` files.

NOTE: Refer to the *VM600 networking manual* (edition 6 or earlier) for additional information.

THIS PAGE INTENTIONALLY LEFT BLANK

6 CPUM CONFIGURATOR

The management of networked VM600 racks with a CPUM card running firmware version 075 or later can be simplified using the CPUM Configurator software.

NOTE: CPUM Configurator is compatible with CPUM cards running firmware version 075 or later.

To identify the firmware version of a CPUM card, see 3.3 Identifying the firmware version of a CPUM card.

To work with CPUM cards running firmware version 074 or earlier, see 7 Terminal emulation or refer to the *VM600 networking manual* (edition 6 or earlier).

6.1 About CPUM Configurator

CPUM Configurator is a program that communicates with a CPUM card in a VM600 rack over an Ethernet (TCP/IP) link.

Basically, it provides a graphical user interface for a Telnet session between the CPUM Configurator (Telnet client) and a CPUM card (Telnet server), and is used primarily for configuring CPUM cards and managing VM600 racks over Ethernet links.

The CPUM Configurator software is a single Windows® program file (`*.exe`).

6.1.1 CPUM card configuration files

A CPUM card uses a number of different configuration files to configure separate features and functions of the CPUM card. It is these underlying configuration files that are accessed, modified and/or updated using CPUM Configurator.

The configuration files are application dependant, so not all files are used by all CPUM cards. For example, only a monitoring system that provides condition monitoring functions using XMx16 conditioning monitoring cards requires `xmcsrv.cfg` and `xmcnn.xml` configuration files.

In general, a complete user configuration for a CPUM card consists of the files given in Table 6-1.

Table 6-1: Configuration files for CPUM cards

CPUM card firmware version 075 or later	
Configuration file	**Description**
`launcher.cfg`	Launcher process (for monitoring all other processes): Configuration information for the launcher process (that monitors the other processes) running on the CPUM card.
`modbusDefault.cfg`	Modbus and PROFINET communications: Configuration information for Modbus and PROFINET communications, including register mapping information for the Modbus interface and PROFINET slot information for the PROFINET interface running on the CPUM card.
`profinet.cfg`	PROFINET communications: Additional configuration information for PROFINET communications, including the device name and gateway for the PROFINET interface running on the CPUM card.
`net.cfg`	Network adapters (IP addressing): Configuration information for the Ethernet interfaces, including IP address and netmask information (corresponding to the CPUM TCP-IP Configuration dialog box)
`ntp.conf`	NTP server: Configuration information for the local network time protocol (NTP) server running on the CPUM card.
`uc.cfg`	Summary information: Basic information about the complete user configuration downloaded from a CPUM card (**Save UC from CPUM**).
`xmcnn.xml`	XMx16 card – individual: Configuration information for an XMx16 card installed in slot *nn* of the same VM600 rack as the CPUM card that communicates with the VibroSight Server process running on the CPUM card.
`xmcsrv.cfg`	XMx16 cards – summary: Basic information about the configuration of the XMx16 cards installed in the same VM600 rack as the CPUM card (corresponding to the XMCSRV Configuration dialog box).

See also 2 CPUM card directory structure and configuration files.

6.2 Installing CPUM Configurator

The CPUM Configurator software is included with VM600 MPSx software version 2.7 or later and is copied to the computer as part of the VM600 MPSx software installation process.

CPUM Configurator is installed in the same folder as the VM600 MPSx software on the hard disk. The default folders are:

- `C:\Program Files\VM600_MPS\Bin` – This is the default folder for the VM600 MPSx software's program files on 32-bit versions of Windows operating systems.
- `C:\Program Files (x86)\VM600_MPS\Bin` – This is the default folder for the VM600 MPSx software's program files on 64-bit versions of Windows operating systems.

Alternative folders can be chosen if desired.

After installation by the VM600 MPSx software, CPUM Configurator appears on the Windows Start menu but no shortcut (icon) is added to the Windows desktop.

CPUM Configurator can also be obtained from Meggitt customer support (see 12.1 Contacting us), in which case, the CPUM Configurator program file (`CPUMConfig.exe`) must to copied to a folder on the local hard disk of the computer.

6.3 Starting CPUM Configurator

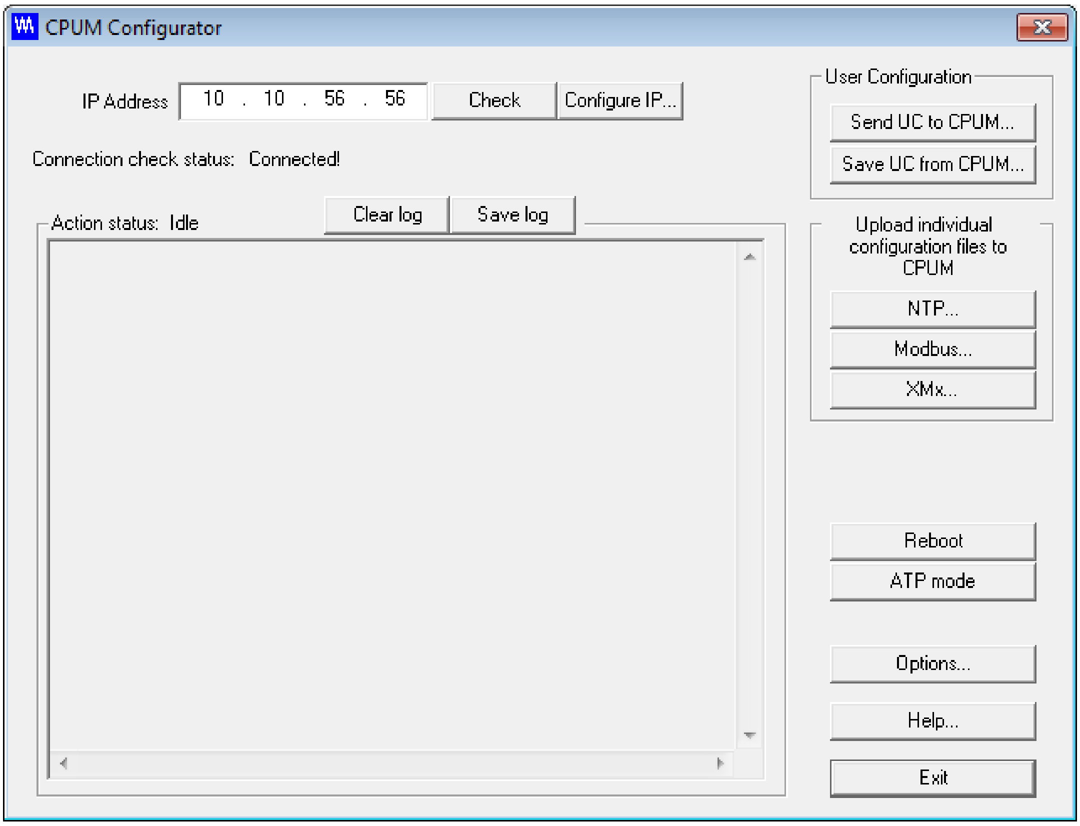
To start CPUM Configurator:

1- Click **Start > All Programs > VM600 MPS > CPUM Configurator**, or navigate to the CPUM Configurator program file (`CPUMConfig.exe`) on the local hard disk of the computer and double-click it.

2- When prompted for a login, type `user` (the default login) for the **User name**, type `config` (the default password) for the **Password**, then click **OK**.

If the **Login automatically** checkbox is selected before clicking **OK**, CPUM Configurator will remember and automatically use the user name and password the next time that CPUM Configurator is started.

3- The CPUM Configurator window is displayed, as shown in Figure 6-1.



IP Address 10 . 10 . 56 . 56

Connection check status: Connected!

Action status: Idle

Figure 6-1: CPUM Configurator

6.4 Stopping CPUM Configurator

To close CPUM Configurator, either:

- Click the **Exit** button.
- Or click the Windows **Close** button (top right).

This closes the connection to the CPUM card and exits CPUM Configurator.

6.5 Help on using CPUM Configurator

To obtain help on using CPUM Configurator:

- Click the **Help** button or press the F1 key.

The CPUM Configurator Help window that is displayed must be closed before continuing to work with CPUM Configurator.

6.6 Feedback when using CPUM Configurator

When using CPUM Configurator, feedback is given using individual dialog boxes and/or in the Action status output text box (see 6.6.1 Action status).

6.6.1 Action status

The status and results of operations performed by the CPUM Configurator software and the CPUM firmware are logged in the **Action status** output text box.

Some of the operations performed by a CPUM card are performed in the background and can take some time to finish, so their ongoing status is logged here.

In addition, other feedback to the user such as instructions to restart (reboot) the CPUM card are displayed here.

6.6.2 Clear Log

Click the **Clear Log** button to clear all of the information logged in the Action status output text box.

6.6.3 Save Log

Click the **Save Log** button to save the information currently logged in the Action status output text box to a text file (`*.txt`) on the computer running CPUM Configurator, for future reference. (That is, all of the information logged since the start of the CPUM Configurator session or the last Clear Log operation.)

The Save As dialog box that appears allows the location and file name of the file to be specified.

6.7 Establishing communications with a CPUM card

To start communicating with a CPUM card, in the CPUM Configurator window (see Figure 6-1):

1- In the **IP Address** text box, type the IP address of the CPUM card in dot-decimal notation. For example, `10.10.56.56`.

NOTE: If the IP address of a CPUM card is not known, it can be discovered using a direct serial link (RS-232) to the CPUM card. See 3.4 Discovering the IP address of a CPUM card.

2- Click the **Check** button to establish a connection between CPUM Configurator and the CPUM card with the address specified in the IP Address text box.

The status of the connection between CPUM Configurator and a CPUM card is displayed under **Connection Check Status**.

3- Either:

`Connection Check Status: Connected!` is displayed to indicate that a communications link has been established.

`Connection Check Status: No response from IP address` is displayed to indicate that a communications link has not been established.

A connection between CPUM Configurator and a CPUM card must be established (Connection Check Status: Connected!) before other CPUM Configurator operations can be performed.

6.8 Changing the IP address of a VM600 rack (CPUM card)

To change the IP address of a VM600 rack (CPUM card) using CPUM Configurator:

1- Click the **Configure IP** button to display the IP address and netmask (subnet mask) information configured for the CPUM card's Ethernet interfaces.

The **CPUM TCP-IP Configuration** dialog box that appears displays the currently configured **IP Address** and **Netmask** for each network interface (Primary Etherent Controller and Secondary Ethernet Controller) from the CPUM card. In addition, **Force 10 Mbps** indicates if the data transfer rate of the network interface is limited.

2- To change an IP address, in the appropriate **IP Address** text box, enter the required IP address for the Primary or Secondary Ethernet Controller of the CPUM card in dot-decimal notation.

To change an subnet mask, in the appropriate **Netmask** text box, enter the required subnet mask for the Primary or Secondary Ethernet Controller of the CPUM card in dot-decimal notation.

To change the data transfer rate of the network interface for the Primary Ethernet Controller of the CPUM card, select or clear the **Force 10 Mbps** checkbox as appropriate: selecting **Force 10 Mbps** limits the data transfer rate to 10 Mbps.

NOTE: The **Secondary Ethernet Controller** checkbox must be selected in order to change the IP address and subnet mask for the secondary network interface. There is no requirement to limit the data transfer rate of the second secondary network interface, so this the **Force 10 Mbps** checkbox is unavailable (greyed out).

3- Click **OK** to transfer the new IP address settings to the CPUM card.

The values displayed in the **CPUM TCP-IP Configuration** dialog box correspond to the information stored in the `net.cfg` file on the CPUM card. Entering a new IP address or subnet mask (netmask) in the CPUM TCP-IP Configuration dialog box and clicking OK saves any changes to the file.

However, in order for any changes to the `net.cfg` file to take effect, the CPUM card must be restarted.

4- As instructed by the feedback in the Action status output text box, click **Reboot** to restart the CPUM card.

Alternatively, remove and then insert the CPUM card, or turn off and then turn on the power supply to the VM600 rack, in order to restart the CPUM card.

5- The CPUM card restarts using the new IP address settings.

NOTE: After restarting the CPUM card, it is necessary to re-establish communications with the CPUM card using the new IP address in the **IP Address** text box.

6.9 Downloading all configuration files from a CPUM card

Using CPUM Configurator, all of the configuration files on a CPUM card can be downloaded at the same time (that is, the complete user configuration (UC)).

To download all of the configuration files on a CPUM card:

1- Under **User Configuration**, click the **Save UC from CPUM** button.

The **Create New CPUM UC** dialog box appears.

2- Under **Path**, click the browse (**...**) button and use the dialog box that appears to navigate the folders on the computer and select a folder in which to save the configuration files. Click **OK** to continue.

Under **UC tag**, enter tag name to be used in the summary `uc.cfg` file generated by the download. This tag name is also used a subfolder in which the configuration files are saved.

Under **Description**, accept the default or enter a description for the configuration files.

Under **FW version**, accept the default or enter a description for the firmware running on the CPUM card.

3- Then click **Create** to download all of the configuration files from the CPUM card and create a local copy of the user configuration on the computer.

The directory structure for the downloaded configuration files is as follows:

```
\UC tag\uc.cfg
\UC tag\etc\net.cfg
\UC tag\etc\cpum\launcher\launcher.cfg
\UC tag\etc\cpum\mbsrv\modbusDefault.cfg
\UC tag\etc\cpum\xmcsrv\xmcnn.xml
\UC tag\etc\cpum\xmcsrv\xmcsrv.cfg
\UC tag\etc\ntp\ntp.conf
```

The configuration files downloaded from a CPUM card can be edited locally on a computer then uploaded to the card, in order to change the configuration running on the card.

The download configuration files operation can also be used as a backup of a CPUM card's configuration.

6.10 Uploading all configuration files to a CPUM card

Using CPUM Configurator, all of the configuration files for a CPUM card can be uploaded at the same time (that is, a complete user configuration (UC)).

To upload all of the configuration files to a CPUM card:

1- Under **User Configuration**, click the **Send UC to CPUM** button.

The **Open** dialog box appears.

2- Use the dialog box to navigate the folders on the computer and select the summary `uc.cfg` file corresponding to the configuration to upload.

3- Then click **Send UC** to upload all of the configuration files from the local copy of the complete user configuration on the computer to the CPUM card.

4- As instructed by the feedback in the Action status output text box, click **Reboot** to restart the CPUM card, which forces the card to use the new configuration.

Alternatively, remove and then insert the CPUM card, or turn off and then turn on the power supply to the VM600 rack, in order to restart the CPUM card.

The required directory structure for the uploaded configuration files is the same as the structure created when all configuration files are downloaded. See 6.9 Downloading all configuration files from a CPUM card.

The configuration files downloaded from a CPUM card can be edited locally on a computer then uploaded to the card, in order to change the configuration running on the card.

The upload configuration files operation can also be used to restore a CPUM card's configuration.

NOTE: The `net.cfg` file contains the network interface (IP addressing) information for a CPUM card running firmware version 075 or later. When a CPUM card starts, it reads the `net.cfg` file in order to learn its network interface settings.
So if a `net.cfg` file with different settings is uploaded to a CPUM card and the CPUM card is restarted, then it is necessary to re-establish communications with the CPUM card using the new IP address.

6.11 Uploading individual configuration files to a CPUM card

Using CPUM Configurator, certain configuration files for a CPUM card can also be uploaded individually.

6.11.1 Uploading an NTP configuration file to a CPUM card

To upload an NTP configuration file to a CPUM card:

1- Under **Upload individual configuration files to CPUM**, click the **NTP** button.
The **Open** dialog box appears.

2- Use the dialog box to navigate the folders on the computer and select the `ntp.conf` file to upload.

3- Then click **Send** to upload the configuration file to the CPUM card.

NOTE: The CPUM card must be restarted in order for this configuration change to apply to the running network time server process.

4- Feedback in the **Action status** output text box indicates the result of the operation.

5- As instructed by the feedback in the Action status output text box, click **Reboot** to restart the CPUM card, which forces the card to use the new configuration.
Alternatively, remove and then insert the CPUM card, or turn off and then turn on the power supply to the VM600 rack, in order to restart the CPUM card.

6.11.2 Uploading a Modbus configuration file to a CPUM card

To upload a Modbus configuration file to a CPUM card:

1- Under **Upload individual configuration files to CPUM**, click the **Modbus** button.
The **Open** dialog box appears.

2- Use the dialog box to navigate the folders on the computer and select the `modbusDefault.cfg` file to upload.

3- Then click **Configure** to upload the configuration file to the CPUM card.

NOTE: When the file is uploaded, it is checked (parsed) by the CPUM card to ensure that there are no errors and the configuration is valid.

4- Feedback in the **Action status** output text box indicates the result of the operation.

The CPUM card will automatically apply this configuration change to the running Modbus process (no restart is required).

6.11.3 Uploading an XMx16 configuration file to a CPUM card

NOTE: In the CPUM Configurator user interface, the **XMx** button is unavailable (greyed out) when the user is logged in with the default username and password (`user` and `config`, respectively) as this button is for a special feature that is reserved for factory use. It is described below for information only.

Certain machinery monitoring applications require that a VM600 XMx16 card (XMC16, XMV16 or XMVS16) supports communication with more than one server at the same time. For example, to allow a control system and a condition monitoring system running in parallel to “share” the measurement data from an XMx16 card.

The two servers used with a shared XMx16 card can be either two VibroSight Server software modules running on a host computer or computers, or a VibroSight Server software module running on a host computer and a VibroSight server process running on a CPUM card.

The VibroSight server process running on a CPUM card requires a copy of the configuration for the shared XMx16 card in order to be able to communicate with it. The CPUM card can obtain the XMx16 card’s configuration in one of two ways:

- Using CPUM Configurator to upload the XMx16 card’s configuration directly from the XMx16 card via the VME bus on the VM600 rack’s backplane.
- Using CPUM Configurator to upload the XMx16 card’s configuration from an individual device configuration file (`*.xml`) for an XMx16 card that was exported from the VibroSight Configurator module of the VibroSight® software.

To avoid conflicts, only one of the two servers can have read and write access to the shared VM60 XMx16 card (the other server will have read only access). The server with read and write access is the “master” and in a typical application, would be the more important system, for example, a control system. This access is configured using VibroSight Configurator.

To upload an XMx16 card configuration file to a CPUM card:

1- Under **Upload individual configuration files to CPUM**, click the **XMx** button.

The **XMCSRV Configuration** dialog box that appears allows XMx16 card configurations to be specified for up to two XMx16 cards.

2- For each XMx16 card configuration being "shared" with the VibroSight server process running on the CPUM card, under **XMx Card 1** and **XMx Card 2**:

- Enter the IP address (**IP Address**) and the VM600 rack slot number (**Slot**) for the card.
- Select **Force pairing** if the VibroSight server process running on the CPUM card is required to be the “master” of the XMx16 card (that is, allow read and write access to the XMx16 card by the CPUM card).

- Under **XMC board configuration**, either:

  Click **get configuration from Xmx card** to upload the XMx16 card's configuration directly from the XMx16 card via the VME bus on the VM600 rack's backplane, or

  Click **set configuration from file**, click the browse (**...**) button and use the dialog box that appears to navigate the folders on the computer and select an individual device configuration file (`xmcnn.xml`) to upload the XMx16 card's configuration that was exported from the VibroSight Configurator module of the VibroSight® software.

3- Then click **Configure** to upload the configuration file to the CPUM card.

NOTE: Alternatively, the **Import** button can be used to complete the **XMCSRV Configuration** dialog box using the information in a `xmcsrv.cfg` file (for example, from a previously saved complete user configuration).

4- Feedback in the **Action status** output text box indicates the result of the operation.

The CPUM card will automatically apply this configuration change to the running XMx16 card process (no restart is required).

6.12 Changing the configuration of a CPUM card

Using CPUM Configurator, the recommended way to change the configuration of a CPUM card is to:

1- Use **Save UC from CPUM** to download a copy of the complete user configuration (UC) from the CPUM card to a computer.

See 6.9 Downloading all configuration files from a CPUM card.

2- Edit the local copy of the configuration on the computer using a suitable text editor.

3- Use **Send UC to CPUM** to upload the modified copy of the user configuration from the computer to the CPUM card.

See 6.10 Uploading all configuration files to a CPUM card.

6.13 Changing CPUM Configurator options

The CPUM Configurator software has default behaviour that can be changed using the options settings:

1- Click the **Options** button to display the user configurable settings for CPUM Configurator.

The Options dialog box that appears displays the User Name and Password used to log in to CPUM Configurator and establish a connection with a CPUM card.

2- Select **Login Automatically** and click **OK** in order for CPUM Configurator to remember and use the password to automatically log in the next time that the software is started (no login window).

6.14 Rebooting a CPUM card

Click the **Reboot** button to force the CPUM card to restart.

(This is the equivalent of turning the power supply to the CPUM card off and on.)

If a CPUM Configurator operation has been performed that requires a restart such as modifying an IP address or network time protocol (NTP) configuration:

- A message is logged in the Action status output text box.
- Text will appear above the Reboot button (**Reboot is required to apply the changes**) as a reminder to restart the CPUM card.

6.15 ATP Mode

NOTE: In the CPUM Configurator user interface, the **ATP mode** button is unavailable (greyed out) when the user is logged in with the default username and password (`user` and `config`, respectively) as this button is for a special feature that is reserved for factory use. It is described below for information only.

The ATP Mode button puts the CPUM card in an acceptance test procedure (ATP) mode that supports the testing of the card (reserved for factory use only).

However, if a CPUM card is inadvertently put into ATP Mode, simply press the ALARM RESET button on the front panel of the CPUM card to exit ATP Mode and return to normal operation with the current configuration.

THIS PAGE INTENTIONALLY LEFT BLANK

7 TERMINAL EMULATION

The management of networked VM600 racks with a CPUM card running firmware version 075 or later can be performed using a terminal emulator and Midnight Commander.

This chapter of the manual will use the PuTTY terminal emulator program to demonstrate the use of terminal emulation with a CPUM card but other terminal emulators can also be used, for example, the HyperTerminal program or a VT100 video terminal.

NOTE: Midnight Commander is available on CPUM cards running firmware version 074 or later.

To identify the firmware version of a CPUM card, see 3.3 Identifying the firmware version of a CPUM card.

To work with CPUM cards running firmware version 073 or earlier, refer to the *VM600 networking manual* (edition 6 or earlier).

7.1 About PuTTY

PuTTY is a free open-source terminal emulator program that can be used to communicate with a CPUM card in a VM600 rack. The PuTTY program supports the Telnet protocol (Telnet client) which allows communication with a CPUM card (Telnet client) over an Ethernet link. It also supports the serial communication protocols which allows communication with a CPUM card over a serial link.

NOTE: CPUM Configurator is compatible with CPUM cards running firmware version 075 or later.

The Putty software is available as a single program file (`*.exe`) or as an installer from here:

- http://www.putty.org
- http://www.chiark.greenend.org.uk/~sgtatham/putty

Windows® and other operating systems are supported.

7.2 About Midnight Commander

Midnight Commander is a file manager that provides a text-based user interface which allows directory browsing, file management and text-based file editing to be easily performed. The main interface consists of two panels which display the file system. File selection is done using arrow keys, the ENTER key is used to select files and the function keys perform operations such as renaming, editing and copying files. Midnight Commander also includes an editor (called mcedit) and has mouse support.

Midnight Commander is started by typing `mc` (then pressing ENTER) at the command line when connected to a CPUM card in a terminal emulation session.

7.3 Starting a terminal emulation session using PuTTY and establishing communications with a CPUM card

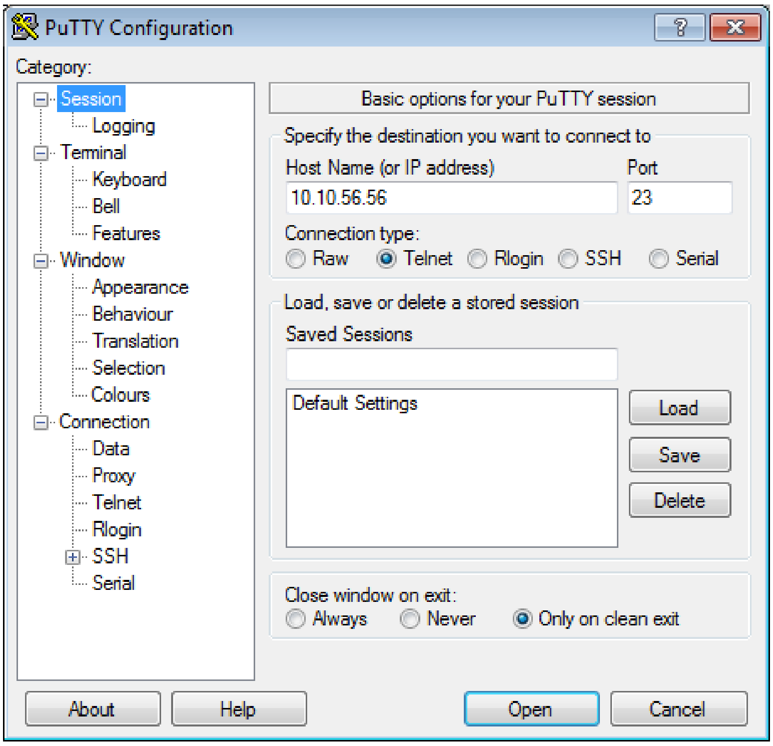
To start a terminal emulation session using PuTTY and establish communications with a CPUM card:

1- Double-click the PuTTY icon (shortcut) on the Windows desktop.

Alternatively, navigate to the PuTTY program file (`putty.exe`) on the local hard disk of the computer and double-click it.

2- Start PuTTY, select a **Telnet** connection and enter the IP address (**Host Name (or IP address)**) of the CPUM card, as shown in Figure 7-1. (Telnet always uses TCP port number 23 (**Port**).)

Alternatively, a different terminal editor that supports Telnet could be used.



Specify the destination you want to connect to

Host Name (or IP address): 10.10.56.56

Port: 23

Connection type: ○ Raw ● Telnet ○ Rlogin ○ SSH ○ Serial

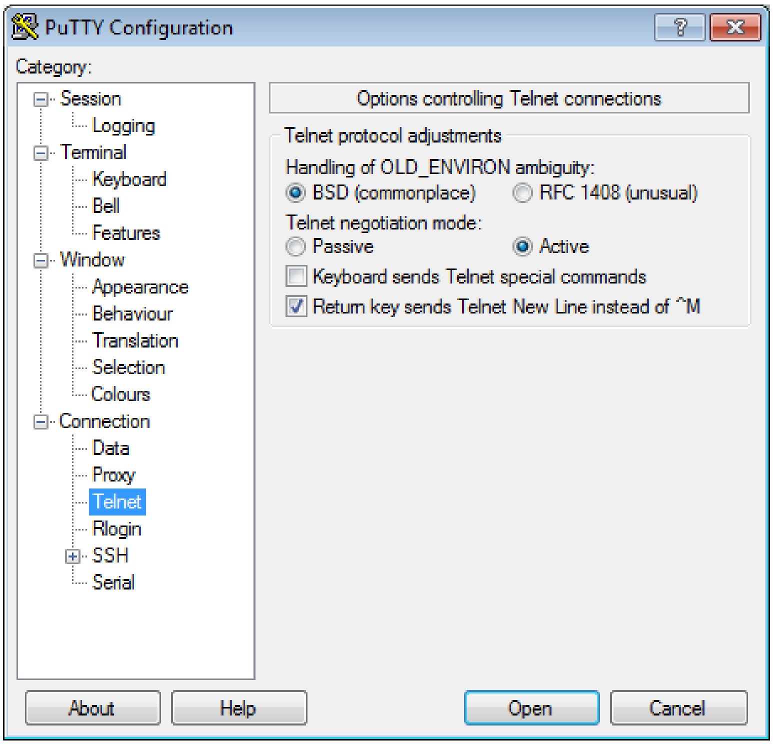
Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 7-1: Using PuTTY terminal emulation software – selecting and configuring a Telnet connection

3- Accept the default PuTTY Telnet session options, as shown in Figure 7-2.



Options controlling Telnet connections

Telnet protocol adjustments

Handling of OLD_ENVIRON ambiguity:
● BSD (commonplace) ○ RFC 1408 (unusual)

Telnet negotiation mode:
○ Passive ● Active

□ Keyboard sends Telnet special commands

☑ Return key sends Telnet New Line instead of ^M

Figure 7-2: Using PuTTY terminal emulation software – default Telnet options

4- Click **Open** to start the PuTTY session, then press the ENTER key.

5- When prompted for a login, type **user** (the default login), then press ENTER to continue. For example:

```
login: user
```

When prompted for a password, type **config** (the default password), then press ENTER to continue. For example:

```
password: config
```

The command-line prompt (`/usr/user #`) is displayed when the log in is successful.

The terminal command-line and Midnight Commander are used to perform all operations.

7.4 Stopping a terminal emulation session using PuTTY

To close a terminal emulation session using PuTTY, either:

- Type `logout` (then press ENTER) at the terminal command line.
- Or click Windows **Close** button (top right).

This ends the terminal emulation session and exits PuTTY.

7.5 Help on using PuTTY

To obtain help on using PuTTY:

- After starting PuTTY, click the **Help** button.

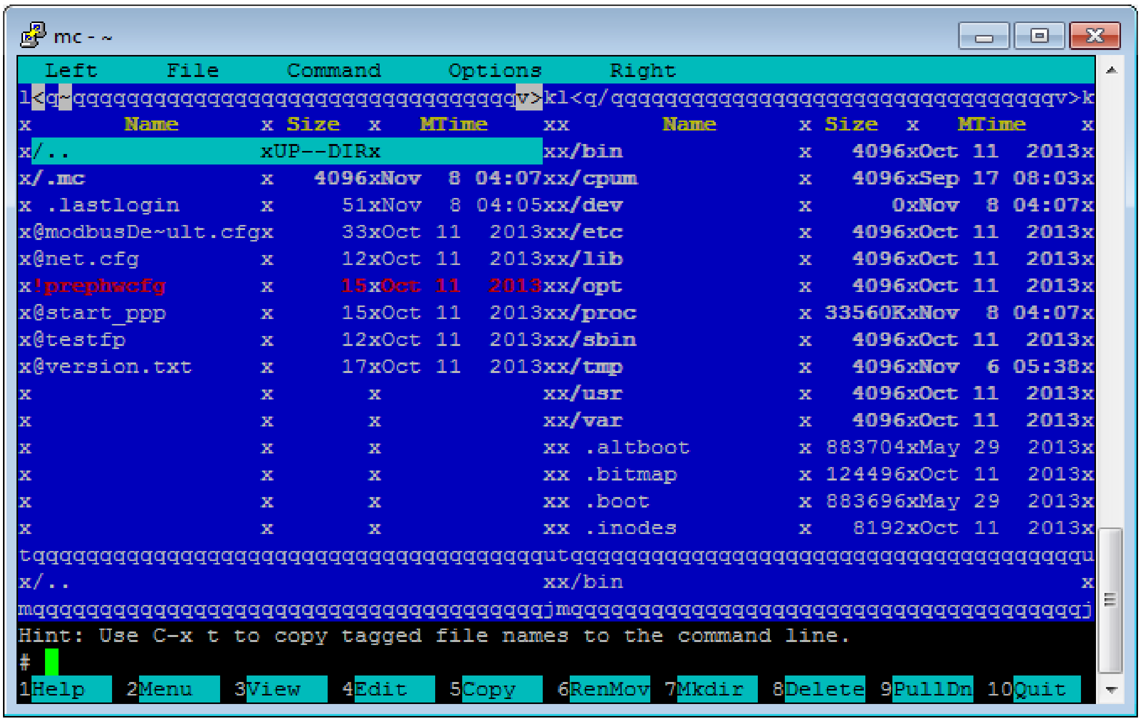
The PuTTY User Manual is displayed.

7.6 Starting Midnight Commander

To start Midnight Commander:

- Type mc (then press ENTER) at the terminal command line.

Midnight Commander starts and its user interface is displayed in the terminal window, as shown in Figure 7-3.



```
  Left     File      Command     Options     Right
l<q~qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqv>kl<q/qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqv>k
x       Name      x Size  x   MTime    xx       Name      x Size  x   MTime    x
x/..              xUP--DIRx            xx/bin             x   4096xOct 11  2013x
x/.mc             x   4096xNov  8 04:07xx/cpum            x   4096xSep 17 08:03x
x .lastlogin      x     51xNov  8 04:05xx/dev             x      0xNov  8 04:07x
x@modbusDe~ult.cfgx     33xOct 11  2013xx/etc             x   4096xOct 11  2013x
x@net.cfg         x     12xOct 11  2013xx/lib             x   4096xOct 11  2013x
x!prephwcfg       x     15xOct 11  2013xx/opt             x   4096xOct 11  2013x
x@start_ppp       x     15xOct 11  2013xx/proc            x 33560KxNov  8 04:07x
x@testfp          x     12xOct 11  2013xx/sbin            x   4096xOct 11  2013x
x@version.txt     x     17xOct 11  2013xx/tmp             x   4096xNov  6 05:38x
x                 x       x            xx/usr             x   4096xOct 11  2013x
x                 x       x            xx/var             x   4096xOct 11  2013x
x                 x       x            xx .altboot        x 883704xMay 29  2013x
x                 x       x            xx .bitmap         x 124496xOct 11  2013x
x                 x       x            xx .boot           x 883696xMay 29  2013x
x                 x       x            xx .inodes         x   8192xOct 11  2013x
tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqutqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqu
x/..                                  xx/bin                                 x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqjmqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
Hint: Use C-x t to copy tagged file names to the command line.
# 
1Help    2Menu    3View    4Edit    5Copy    6RenMov  7Mkdir   8Delete  9PullDn 10Quit
```

Figure 7-3: Midnight Commander user interface

The command-line and Midnight Commander are used to perform all operations.

7.7 Midnight Commander user interface

The Midnight Commander user interface consists of:

- A menu line (top) providing access to commands.
- Two directory panels (left and right) displaying the file system.
- A command line (#) allowing Midnight Commander commands to be entered.

  Just above the command line is a hint line that shows random tips.
- A function key line (bottom) providing keyboard shortcuts access to the most frequently used commands.

7.7.0.1 Navigating Midnight Commander

To switch between the two directory panels (left or right), either press the TAB key to change to the other directory panel, or use the pointer to click in the required directory panel.

To move around a directory panel, either use the UP ARROW and DOWN ARROW keys, or or use the pointer. The PAGE UP and PAGE DOWN keys can be used to move up and down a long directory once screen at a time. The HOME and END keys can be used to jump to the top or bottom of a long directory.

To change up into a parent directory, move to the top of a directory (`/..`) and press the ENTER key. (`/..` refers to a parent directory in the Unix file system used by a CPUM card.)

To change down into a subdirectory, move to the required directory (`/directoryname`) and press the ENTER key.

7.7.0.2 Midnight Commander keyboard shortcuts

The functions keys (F1 to F10) provide access to the most frequently used commands.

Alternatively, use the pointer to select a function by clicking on it in the function key line (bottom).

The actual function provided can be different depending on the context, for example, when browsing a directory, using the file viewer or using the editor. The context-sensitive function is always displayed on the function key line (bottom).

If no physical function keys are available, the function can still be accessed using an ESC key number sequence. For example, pressing the ESC key and then typing **1** is the equivalent of pressing the F1 key. (Pressing the ESC key and then typing **0** is the equivalent of pressing the F10 key.)

To switch between the terminal console (command-line prompt) and Midnight Commander while keeping mc running, press the CTRL key and then type **o** (that is, the lower-case letter).

Shortcuts for the menu commands (top) are displayed in the menus.

7.7.0.3 Midnight Commander menus

The menus provide access to all of the commands available in Midnight Commander.

Left menu – commands for working with the left directory panel. For example, to change the information displayed in the panel.

File menu – commands for working with the files. For example, to view or edit the selected file.

Commands menu – other commands.

Options menu – commands for configuring the appearance and behaviour of Midnight Commander.

Right menu – commands for working with the right directory panel. For example, to change the information displayed in the panel.

To display a menu’s commands, either press the F9 key (PullDn) to activate the menus and use the arrow keys to navigate the menus and select a command. Press ENTER to run the selected command. To deactivate the menus, press the ESC key, then press an arrow key.

Alternatively, use the pointer to display, select and/or run a menu command by clicking on it in the menu (top).

7.8 Exiting Midnight Commander

To close Midnight Commander:

- Either press the F10 key (Quit) or use the pointer to click it.

This exits Midnight Commander (that is, the mc process running on the CPUM card).

7.9 Help on using Midnight Commander

To obtain help on using Midnight Commander:

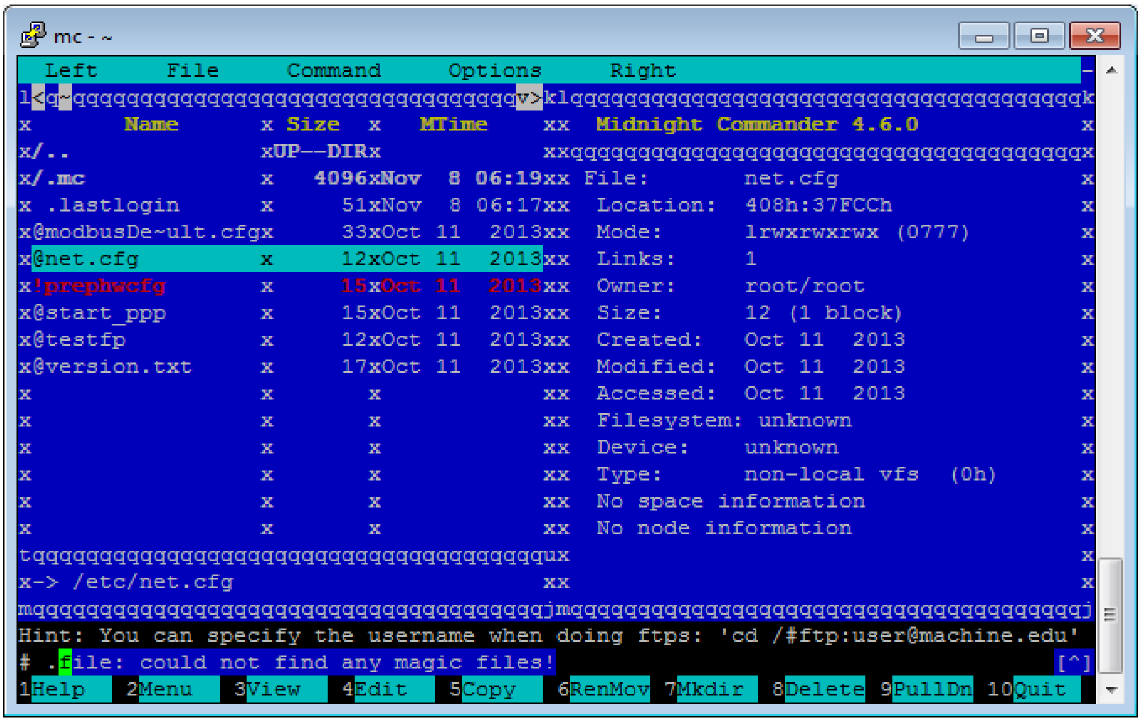
- Either press the F1 key (Help) or use the pointer to click 1 Help.

The Midnight Commander help is displayed.

7.10 Changing the IP address of a VM600 rack (CPUM card)

To change the IP address of a VM600 rack (CPUM card) using PuTTY and Midnight Commander:

1- In a Midnight Commander directory panel, navigate to the `net.cfg` file, as shown in Figure 7-4 (left).

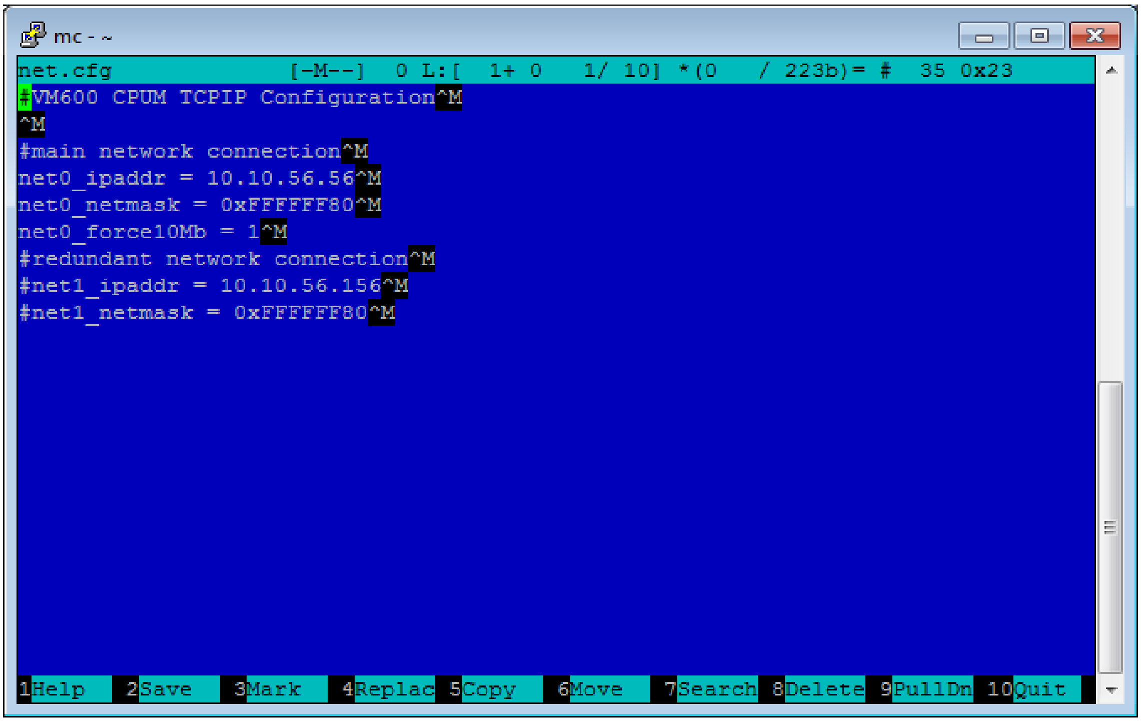


```
  Left      File      Command      Options      Right
l<q~qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqv>klqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x      Name       x Size  x   MTime    xx Midnight Commander 4.6.0             x
x/..              xUP--DIRx            xxqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqx
x/.mc             x   4096xNov  8 06:19xx File:       net.cfg                  x
x .lastlogin      x     51xNov  8 06:17xx  Location:  408h:37FCCh              x
x@modbusDe~ult.cfgx     33xOct 11  2013xx  Mode:      lrwxrwxrwx (0777)        x
x@net.cfg         x     12xOct 11  2013xx  Links:     1                        x
x!prephwcfg       x     15xOct 11  2013xx  Owner:     root/root                x
x@start_ppp       x     15xOct 11  2013xx  Size:      12 (1 block)             x
x@testfp          x     12xOct 11  2013xx  Created:   Oct 11  2013             x
x@version.txt     x     17xOct 11  2013xx  Modified:  Oct 11  2013             x
x                 x       x            xx  Accessed:  Oct 11  2013             x
x                 x       x            xx  Filesystem: unknown                 x
x                 x       x            xx  Device:    unknown                  x
x                 x       x            xx  Type:      non-local vfs  (0h)      x
x                 x       x            xx  No space information                x
x                 x       x            xx  No node information                 x
tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqux                                      x
x-> /etc/net.cfg                       xx                                      x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqjmqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
Hint: You can specify the username when doing ftps: 'cd /#ftp:user@machine.edu'
# .file: could not find any magic files!                                    [^]
1Help   2Menu   3View   4Edit   5Copy   6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Figure 7-4: Using Midnight Commander to select a file

The path to the `net.cfg` file is `/usr/user/net.cfg`

2- To edit the file using Midnight Commander, either press the F4 key (Edit) or use the pointer to click 4 Edit.
The selected file (`net.cfg`) is opened in n Midnight Commander's editor, as shown in Figure 7-5.



```
net.cfg                [-M--]  0 L:[  1+ 0   1/ 10] *(0   / 223b)= #  35 0x23
#VM600 CPUM TCPIP Configuration^M
^M
#main network connection^M
net0_ipaddr = 10.10.56.56^M
net0_netmask = 0xFFFFFF80^M
net0_force10Mb = 1^M
#redundant network connection^M
#net1_ipaddr = 10.10.56.156^M
#net1_netmask = 0xFFFFFF80^M

1Help  2Save  3Mark  4Replac 5Copy  6Move  7Search 8Delete 9PullDn 10Quit
```

Figure 7-5: Using Midnight Commander to edit a file

The `net.cfg` file displays the currently configured IP address (**netx_ipaddr**) and netmask (**netx_netmask**) for the CPUM card's Ethernet interfaces (primary network interface (ETH1) and secondary network interface (ETH2)). In addition, **net0_force10Mb** indicates if the data transfer rate of the network interface is limited. See 3.4.2 About the net.cfg file for additional information.

3- To change an IP address, edit the appropriate **netx_ipaddr** line and type the required IP address in dot-decimal notation.

To change an subnet mask, edit the appropriate **netx_netmask** line and type the required subnet mask in hexadecimal notation.

To change the data transfer rate for the primary network interface (ETH1), under `#main network connection`, edit the **net0_force10Mb** line as appropriate: **net0_force10Mb** = 1 limits the data transfer rate to 10 Mbps and **net0_force10Mb** = 0 does not limit the data transfer rate.

NOTE: The ^M characters at the end of each line correspond to a carriage return (0x0D) and should not be removed.
Only the main computer keyboard should be used when editing files with Midnight Commander: do not use a numeric keypad.

4- To save the edited file, either press the F2 key (Save) or use the pointer to click 2 Save. When prompted, select **Save** to save the file and any changes to the CPUM card. The F10 key (Quit) can be used to quit the file without saving any changes.

However, in order for any changes to the `net.cfg` file to take effect, the CPUM card must be restarted.

5- To close (quit) Midnight Commander and return to the terminal console (command-line prompt), either press the F10 key (Quit) or use the pointer to click 10 Quit.

6- To restart the CPUM card, at the command prompt (`/usr/user # `), type **shutdown -f**, then press ENTER to continue. For example:

```
/usr/user # shutdown -f
```

The shutdown command shuts down and reboots the CPUM card.

Alternatively, remove and then insert the CPUM card, or turn off and then turn on the power supply to the VM600 rack, in order to restart the CPUM card.

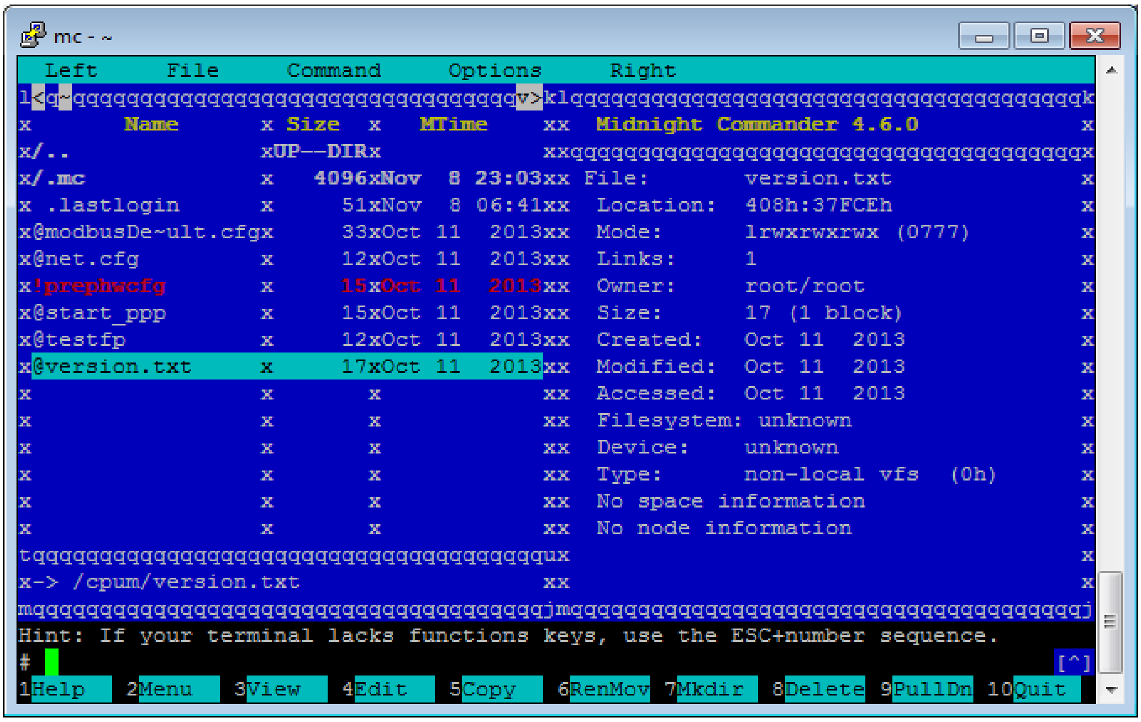
7- The CPUM card restarts using the new IP address settings.

NOTE: After restarting the CPUM card, it is necessary to re-establish communications with the CPUM card using the new IP address in the **IP Address** text box.

7.11 Viewing the contents of a file

To view the contents of a text file using PuTTY and Midnight Commander:

1- In a Midnight Commander directory panel, navigate to a file (for example, `version.txt`), as shown in Figure 7-6 (left).

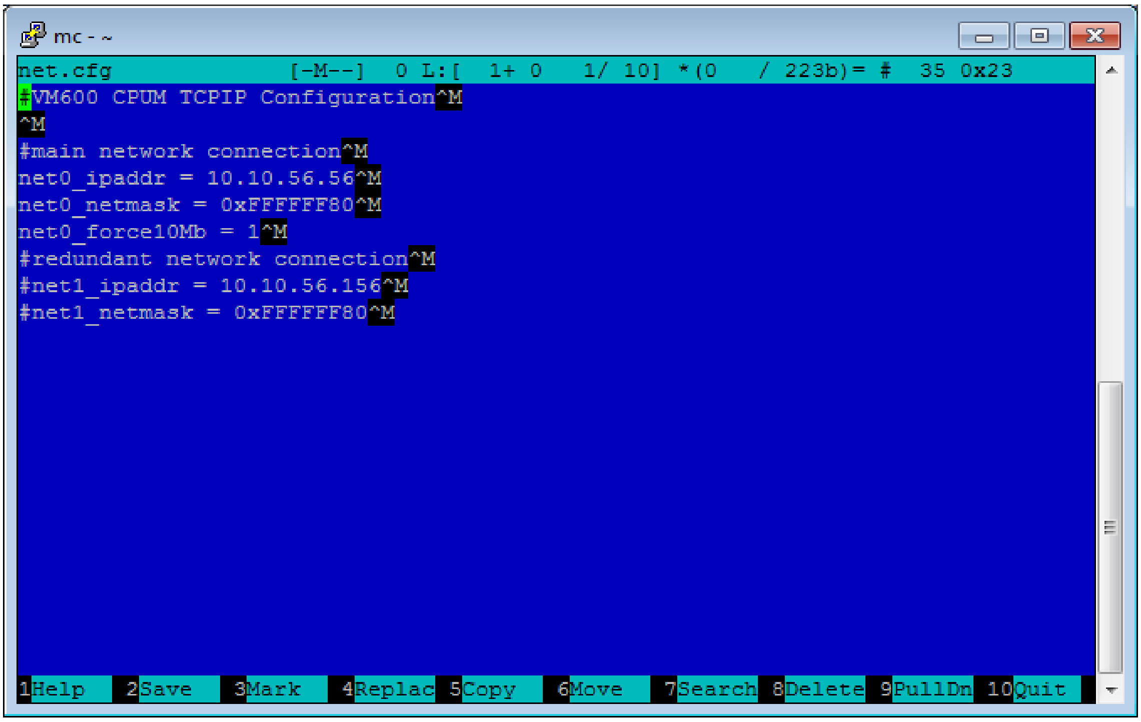


```
  Left      File      Command      Options      Right
l<q~qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqv>klqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x       Name      x Size  x   MTime    xx Midnight Commander 4.6.0           x
x/..              xUP--DIRx            xxqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqx
x/.mc             x   4096xNov  8 23:03xx File:        version.txt           x
x .lastlogin      x     51xNov  8 06:41xx  Location:   408h:37FCEh           x
x@modbusDe~ult.cfgx     33xOct 11  2013xx  Mode:       lrwxrwxrwx (0777)     x
x@net.cfg         x     12xOct 11  2013xx  Links:      1                     x
x!prephwcfg       x     15xOct 11  2013xx  Owner:      root/root             x
x@start_ppp       x     15xOct 11  2013xx  Size:       17  (1 block)         x
x@testfp          x     12xOct 11  2013xx  Created:    Oct 11  2013          x
x@version.txt     x     17xOct 11  2013xx  Modified:   Oct 11  2013          x
x                 x       x            xx  Accessed:   Oct 11  2013          x
x                 x       x            xx  Filesystem: unknown               x
x                 x       x            xx  Device:     unknown               x
x                 x       x            xx  Type:       non-local vfs  (0h)   x
x                 x       x            xx  No space information              x
x                 x       x            xx  No node information               x
tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqux                                      x
x-> /cpum/version.txt                  xx                                      x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqjmqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
Hint: If your terminal lacks functions keys, use the ESC+number sequence.
#                                                                           [^]
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Figure 7-6: Using Midnight Commander to select a file

The path to the `version.txt` file is `/usr/user/version.txt`

2- To view the file using Midnight Commander, either press the F3 key (View) or use the pointer to click 3 View.
The selected file (`version.txt`) is displayed in Midnight Commander, as shown in Figure 7-7.


mc - ~
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

```
net.cfg                  [-M--]   0 L:[  1+ 0   1/ 10] *(0    / 223b)= #   35 0x23
#VM600 CPUM TCPIP Configuration^M
^M
#main network connection^M
net0_ipaddr = 10.10.56.56^M
net0_netmask = 0xFFFFFF80^M
net0_force10Mb = 1^M
#redundant network connection^M
#net1_ipaddr = 10.10.56.156^M
#net1_netmask = 0xFFFFFF80^M
```

Figure 7-7: Using Midnight Commander to view a file

The version.txt file displays the version information for a CPUM card. See 3.3.1 About the version.txt file for additional information.

3- To close the displayed file, either press the F10 key (Quit) or use the pointer to click 10 Quit.

4- To close (quit) Midnight Commander and return to the terminal console (command-line prompt), either press the F10 key (Quit) or use the pointer to click 10 Quit.

7.12 Serial communications with a VM600 rack

Any operation or task that can be performed in a terminal emulation session can equally be performed in a serial communications session (see 3 Serial communications with a VM600 rack).

THIS PAGE INTENTIONALLY LEFT BLANK

8 SETTING UP A MODBUS CONNECTION (CPUM FIRMWARE VERSION 067 OR EARLIER)

8.1 Introduction

This chapter describes the implementation of the Modbus software interface for VM600 racks as used by CPUM cards running firmware version 067 or earlier.

Most, but not all, Modbus registers are available to CPUM cards running firmware version 067 or earlier. The tables in Appendix A: MPC4 Modbus register definitions and Appendix B: AMC8 Modbus register definitions include a column (CPUM < 71) that indicates which Modbus registers are available in a particular CPUM firmware version.

NOTE: A more flexible implementation of the Modbus server is available for CPUM cards running firmware version 071 or later (see 9 Setting up a Modbus connection (CPUM firmware version 071 or later)).

The Modbus software interface provides a Modbus server for a VM600 rack that communicates with the following VM600 cards:

- MPC4
- AMC8.

The transmitted data consists of real-time values (vibration level, pressure and so on), status and configuration information (such as alarms). This data can be used by any external system, such as a distributed control system (DCS), for the purposes of machinery monitoring.

See Appendix A: MPC4 Modbus register definitions for detailed MPC4 Modbus register mapping information and Appendix B: AMC8 Modbus register definitions for detailed AMC8 Modbus register mapping information.

8.2 Modbus

Modbus started in the 1970s as a serial communications protocol published by Modicon for use with programmable logic controllers (PLCs). On a Modbus network, controllers communicate using a master-slave technique, in which only one device (the master) can initiate queries (transactions). The other devices on the network (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query.

Since then, Modbus has evolved into become one of the most widely used standards in industrial automation and control. It has become an application layer messaging protocol for client-server communication between devices connected on different types of buses or networks. However, it is still mainly used to exchange data in the field of automation.

Today, Modbus is a client-server protocol based on transactions (see Figure 8-1), which consist of:

- A request issued by the client
- A response issued by the server.

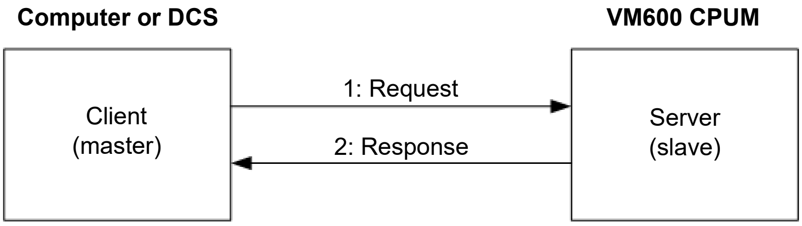


Figure 8-1: A Modbus transaction

In this client-server model of Modbus communication, the CPUM of the VM600 is the Modbus server that responds to external transaction requests from client equipment, such as a computer or DCS.

Modbus is a request/response protocol and offers services specified by function codes. Modbus function codes are elements of Modbus request/reply PDUs and are discussed further in 8.4 Description of Modbus RTU protocol.

Versions of the Modbus protocol exist for serial communication interfaces (the Modbus RTU protocol) and for Ethernet (the Modbus TCP protocol).

Visit the Modbus Organization website for the most up-to-date information:
http://www.modbus.org/specs.php

8.3 Modbus communication

Communication is performed using either:

- A serial line using the Modicon standard Modbus RTU protocol

  and/or

- An Ethernet line using the Modicon standard Modbus TCP protocol.

8.3.1 Modbus RTU

The Modbus RTU protocol can be used with RS-232 and RS-485 serial interfaces. The address field of a Modbus RTU message frame contains eight bits so valid slave device addresses are in the range from 1 to 247 (decimal). (Address 00 is reserved for broadcast transactions to all server devices in the network, but is not supported by the Modbus implementation from Meggitt vibro-meter®.)

In practice, however, the maximum number of instruments or devices that can be addressed is limited by the RS-485 specification. This states that the bus can support up to 32 “unit loads”, so the maximum number of devices that can be connected depends on how much each connected device loads the system down.

The serial link can be half-duplex or full-duplex with selectable transmission rates from 1200 to 19200 bps. The range of supported communication parameters are given in Table 8-1. The default VM600 communication parameter settings can be seen in 8.4.2 Communications parameters for the VM600.

Table 8-1: RTU communication parameters supported by the VM600

Baud rate	Parity	Slave address
600, 1200, 4800, 9600, 19200, 38400, 57600, 115200	Even, odd or none	1 to 247

NOTE: When the CPUM card contains an optional serial communications module (allowing RS-485/RS-422 communication over the “A” and “B” pairs of connectors on its associated IOCN card), only full duplex transmission is possible.

8.3.2 Modbus TCP

The Modbus TCP protocol has exactly the same layout as the Modbus RTU protocol, with the exception of the framing sequence check pattern and the address interpretation.

Again, communications are based on the master-slave principle and the VM600 rack is the slave in the system. The master equipment requests data from the addressed slave, which can only respond. The slave cannot initiate a transaction. The master can address individual slaves to request or send data. It can also request an action to be taken by one or all the slaves in the network.

8.4 Description of Modbus RTU protocol

8.4.1 Frame and timing

NOTE: The following definitions of the Modbus protocol are adapted to the VM600 MPS.

The Modbus RTU protocol is a serial data transmission format widely used in communications with programmable controllers. It is easily adaptable to other types of remote units thanks to its particular message structure (that is, it doesn't operate with variables but with memory addresses).

As shown in Figure 8-2, the first character of a frame is the slave address, followed by the Modbus function code and the information field. Finally, two bytes are reserved for an error check code based on cyclic redundancy checking (CRC).

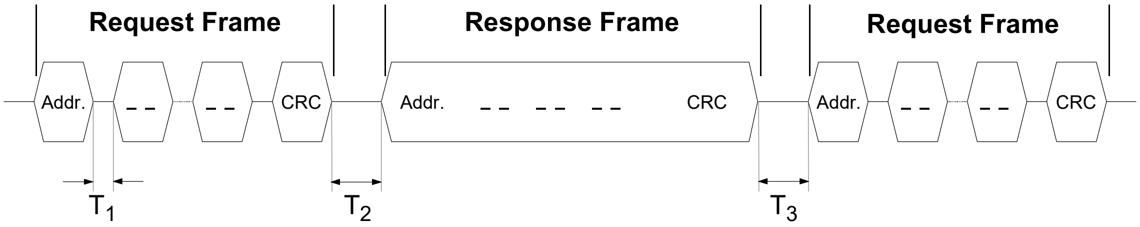
Each character is composed of 10 bits: 1x start bit, 8x data bits and 1x stop bit.

Slave address
Modbus function code
Data
CRC (2 bytes)

Figure 8-2: Constitution of a frame

The Modbus RTU protocol uses no delimiter characters at the beginning or at the end of a message. Each frame must be preceded and followed by a silent interval of at least 3.5 character times (CTs). See Figure 8-3.

The connected equipment (VM600 rack) detects the start of a message when any valid character (containing either its address or the address 00) has been received after a silent interval of at least 3.5 CT. The end of a message is interpreted by a silent interval of the same duration.



T_1 = time between two characters (min. = 0, max. = 3.5 CT)
T_2 = time between request and response (min. 3.5 CT)
T_3 = time between response and next request (min. 3.5 CT)
CT = 1 character time (= the time needed to transmit one character consisting of 10 bits)

Baud (bps)	Time for 3.5 CT
1200	30 ms
2400	15 ms
4800	8 ms
9600	4 ms
19200	2 ms

Figure 8-3: Timing considerations

8.4.2 Communications parameters for the VM600

By default, the communications parameters for the VM600 are:

• Baud rate	9600
• Parity	Even
• Number of bits	8
• Number of stop bits	1
• Slave address	1.

These settings can be modified if required by editing the appropriate configuration file stored on the CPUM (for example, “mbcfg.2”, “mbcfg.3”, “mbcfg.4” or “mbcfg.5”). This is done using either a VT100 terminal or a terminal emulator in the Windows environment). See 2.3 Modbus configuration files for additional information.

8.4.3 Modbus functions supported by the VM600

The Modbus functions supported by the VM600 are shown in Table 8-2.

Table 8-2: Modbus functions supported

Modbus function code	Function description
01 (0x01)	Read coils
02 (0x02)	Read discrete inputs
03 (0x03)	Read holding register
04 (0x04)	Read input registers

The application of these Modbus functions to a VM600 are shown in Table 8-3.

Table 8-3: The application of the Modbus functions supported by a VM600

Modbus function code	Use of function
01	Used to read VM600 status or configuration information (single-bit), such as alarm, danger and OK status
02	Same as Modbus function code 01
03	Used to read dynamic values and configuration information (in 16-bit integer format) from the VM600
04	Same as Modbus function code 03

When used to read values from a MPC4 card, Modbus functions 01 and 02 can be used interchangeably – they both return the same result.

Similarly, when used to read values from a MPC4 card, Modbus functions 03 and 04 can be used interchangeably. They both return the same result.

NOTE: If any unsupported Modbus function is requested, an exception code “01” will be returned to indicate the use of an illegal function.
If a value is requested that is out of the range of what is configured in the VM600, the exception code “02” will be returned to indicate the use of an illegal address.

8.4.4 Modbus function formats

8.4.4.1 Functions 01 and 02

Coils and discrete inputs can be read using function 01 or function 02 of the Modbus protocol.

NOTE: A maximum of 2040 discrete values (coils) can be requested at one time.

The request and response frames are shown in Figure 8-4.

Functions 01/02 (Read coils/Read discrete inputs)

Request frame

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Slave address	Modbus function	First coil address to be read	Number of coils to be read	CRC

Response frame

1 byte	1 byte	1 byte	n bytes	2 bytes
Address	Function	Byte count	Bit values (least significant bit is first coil value)	CRC

NOTE: When requesting a variable that is out of the range configured in the VM600, the exception code “02” will be returned to indicate the use of an illegal data address.

Example:

Read eight discrete values from Modbus address 01, starting from the first coil (= coil 0).

Request frame

0x01	**0x01**	**0x00**	**0x00**	**0x00**	**0x08**	**0xXX**	**0xXX**
Slave address	Modbus function	First register address to be read = 0x0000		8 bits to be read		CRC	

An example response if the coil has a value of 0b10010110 (binary) as follows:

0	is	0	Least significant bit (LSB)
1	is	1	
2	is	1	
3	is	0	
4	is	1	
5	is	0	
6	is	0	
7	is	1	Most significant bit (MSB)

This corresponds to a value of 0x96 (hexadecimal).

Response frame

0x01	0x01	0x01	0x96	2 bytes
Address	Function	Byte count	1 byte of data	CRC

Figure 8-4: Formats and example for Modbus function 01

8.4.4.2 Functions 03 and 04

Registers can be read using function 03 or function 04 of the Modbus protocol.

NOTE: A maximum of 127 registers can be requested at one time.

The request and response frames are shown in Figure 8-5.

Function 03/04 (Read multiple register)

Request frame

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Slave address	Modbus function	First register address to be read	Number of words (registers) to be read	CRC

Response frame

1 byte	1 byte	1 byte	n bytes	2 bytes
Address	Function	Byte count	Register values	CRC

NOTE: When requesting a variable that is out of the range configured in the VM600, the exception code “02” will be returned to indicate the use of an illegal data address.

Example:

Read one register from Modbus address 01, starting from register 55 (decimal).

Request frame

0x01	0x03	0x00	0x36	0x00	0x01	0xXX	0xXX
Slave address	Modbus function	First register address to be read = 0x0036 (See Note 1)		1 register to be read		CRC	

Note 1:
Register 01 (decimal) has register address 0x00 (hexadecimal)
Register 17 (decimal) has register address 0x10 (hexadecimal)
Register 49 (decimal) has register address 0x30 (hexadecimal) and so on.

An example response if register 55 has a value of 4660 (decimal), corresponding to hexadecimal 0x1234, is as follows:

Response frame

0x01	0x03	0x02	0x12	0x34	2 bytes
Address	Function	Byte count	Register value		CRC

Figure 8-5: Formats and example for Modbus function 03

8.4.5 Error handling

There are two types of error handling implemented by the Modbus server: communication-related errors and system-related errors.

8.4.5.1 Communication errors

The VM600 Modbus server provides two error exception codes (see Figure 8-6):

- If the client (master) requests a function that is not supported by the VM600, the exception code “01” will be returned. This code indicates an illegal function error exception response.
- If the client (master) requests an address that is not configured in the VM600, the exception code “02” will be returned. This code indicates an illegal data address error exception response.

NOTE: When an error occurs, the server (slave) returns a message response that includes an exception code. To distinguish between the message request and message response, the MSB (most significant byte) of the function code is set to “1” in the message response.

NOTE: Attempts to communicate with (access) an unplugged card will result in the following communication error: data values of 0 (zero) will be returned.

Error handling

Example:

Read one register from Modbus address 01, starting from register 256 (decimal).

Request frame

0x01	0x03	0x00	0xFF	0x00	0x01	0xXX	0xXX
Slave address	Modbus function	First register address to be read = 0x00FF (See Note 1)		1 register to be read		CRC	

Note 1:
Register 1 (decimal) has register address 0x00
Register 17 (decimal) has register address 0x10
Register 256 (decimal) has register address 0xFF and so on.

An example response if register 256 does not exist, is as follows:

Response frame

0x01	0x83	0x02	2 bytes
Address	Function	Exception code	CRC

Figure 8-6: Formats and example for error message

8.4.5.2 System errors

All other VM600 Modbus server errors, such as conversion errors, sensor not connected or data invalid are expressed using coils (status bits). Every card and channel has some status bits and most important bits shall be mapped and made available through Modbus.

For example, the fact that vibration values were not correctly acquired or processed does not cause Modbus error exceptions. The validity of the acquired and processed value should be checked using the status bits.

There are two types of configuration error status, physical errors and logical errors:

- A physical error occurs if the master requires data from a card that is missing or is not responding to the CPUM. For example, this error can be caused if a card is not responding due to a hardware defect or reconfiguration.
- A logical error occurs if the requested data point on a card or output is not properly configured.

For both types of configuration error, the response will be completely filled with zeros.

For physical configuration errors the Physical Configuration Error latch is set.

For logical configuration errors the Logical Configuration Error latch is set.

Every Modbus client has own error latches. The error latch will be cleared on the first non-configuration status request after error status request.

8.5 Description of Modbus TCP protocol

The Modbus TCP protocol has exactly the same layout as the Modbus RTU protocol, with the exception of the framing sequence check pattern (CRC) and the address interpretation.

NOTE: All requests are sent using TCP on registered port **502**.

Requests are normally sent in half-duplex mode on a given connection. Effectively, there is no benefit in sending additional requests on a single connection while a response is outstanding.

The Modbus 'slave address' field is replaced by a single-byte 'unit identifier'. This may be used to communicate using devices such as bridges and gateways which use a single IP address to support multiple independent end units. The VM600 does not use this feature. In the response, this field is always set to the received value, but it is not checked by the system.

The request and response are prefixed by six bytes shown in Table 8-4.

Table 8-4: Modbus TCP prefix bytes

Bytes	Function
0	Transaction identifier (usually 0) copied by server
1	Transaction identifier (usually 0) copied by server
2	Protocol identifier = 0
3	Protocol identifier = 0
4	Length field (upper byte) = 0 (since all messages are smaller than 256)
5	Length field (lower byte) = number of bytes following

Table 8-4: Modbus TCP prefix bytes (continued)

Bytes	Function
6	Unit identifier (previously 'slave address') – copied by server but ignored
7	Modbus function code
8 onwards	Data as required

Example:

A request to read one register at offset 55 (0x37 hexadecimal), containing a value of 4660 (0x1234 hexadecimal), gives the following:

		◄ TCP ►						◄ Modbus ►					
Request	:	00	00	00	00	00	06	0F	03	00	36	00	01
Response	:	00	00	00	00	00	05	0F	03	02	12	34	
Byte	*:*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

See 8.4 Description of Modbus RTU protocol for examples of data.

Users familiar with Modbus RTU should note that the 'CRC-16' or 'LRC' check fields are not needed in Modbus TCP. The TCP/IP and link layer (for example, Ethernet) checksum mechanisms are used instead to verify accurate delivery of the data packet.

8.6 MPC4 card

The information is this section applies to the implementation of the Modbus software interface for VM600 racks as used by CPUM cards running firmware version 067 or earlier.

Improved Modbus functionality is provided by CPUM cards running firmware version 071 or later. See 9 Setting up a Modbus connection (CPUM firmware version 071 or later) for more information.

8.6.1 Definition of registers

The register allocation is defined in files stored on the CPUM that are read when the VM600 rack is started up. These files are:

`mbcfg.2`	For Modbus RTU communication through the **RS-232** connector on the front panel of the CPUM card. This uses a 9-pin D-type connector.
`mbcfg.3`	For Modbus RTU communication through the **RS** connector on the front panel of the IOCN card. This uses a 6P6C (RJ11/RJ25) connector.
`mbcfg.4`	For Modbus RTU communication through the **A** connectors on the front panel of the IOCN card. These use 6P6C (RJ11/RJ25) connectors.
`mbcfg.5`	For Modbus RTU communication through the **B** connectors on the front panel of the IOCN card. These use 6P6C (RJ11/RJ25) connectors.
`mbcfg.tcp`	For Modbus TCP communication through the **1** connector on the front panel of the IOCN card for the primary network interface (ETH1) and through the **2** connector on the front panel of the IOCN card for the secondary network interface (ETH2), if installed. These use 8P8C (RJ45) connectors.

Refer to the *VM600 machinery protection system (MPS) hardware manual* for additional information on the connectors and interfaces supported by a CPUM / IOCN card pair.

These files can be modified if required by editing them with a VT100 terminal (or emulator from the Windows environment). See 2.3 Modbus configuration files for additional information.

NOTE: The `mbcfg.x` files and the `mbcfg.tcp2` file are only required to be edited if you need to change the communication parameters.

Each file is organised in seven columns (see Figure 8-7):

Slot
This is the number of the slot occupied by the DAU (data acquisition unit, that is, the card) in the VM600 rack.

It can have a value of between 3 and 14.

DAU
Indicates the type of card in the slot:

MPC = a MPC4 card.

Output
Defines the channel and/or output of the card in question. See Section 8.6.

Type
Specifies whether the (analog) value returned is the actual measured value or the full-scale setting defined for the output in question:

- VAL = Measured value
- FSD = Full-scale value.

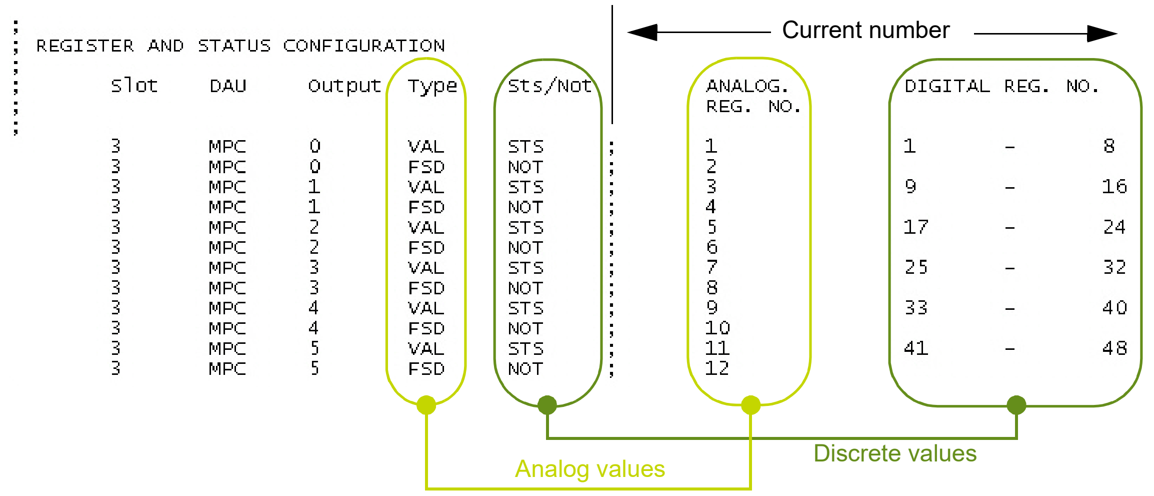
Sts/Not
Defines whether the discrete values concerning the output in question are requested or not:

- STS = Values requested (NB: STS = status)
- NOT = Values not available.

ANALOG. REG. NO.
Register used for analog values concerning the output in question.

DIGITAL REG. NO.
Register used for discrete (digital) values concerning the output in question.

The registers for analog values and discrete values are numbered sequentially and in parallel. The configuration file is scanned, with the numbering of both sets of registers starting at 1 (addressed as 0). Each time an analog register declaration or a discrete input is encountered, the register number (current number) is incremented.



```
;
; REGISTER AND STATUS CONFIGURATION
;
;       Slot      DAU       Output   Type    Sts/Not        ANALOG.      DIGITAL REG. NO.
;                                                           REG. NO.
;
        3         MPC       0        VAL     STS       ;    1            1      -      8
        3         MPC       0        FSD     NOT       ;    2
        3         MPC       1        VAL     STS       ;    3            9      -      16
        3         MPC       1        FSD     NOT       ;    4
        3         MPC       2        VAL     STS       ;    5            17     -      24
        3         MPC       2        FSD     NOT       ;    6
        3         MPC       3        VAL     STS       ;    7            25     -      32
        3         MPC       3        FSD     NOT       ;    8
        3         MPC       4        VAL     STS       ;    9            33     -      40
        3         MPC       4        FSD     NOT       ;    10
        3         MPC       5        VAL     STS       ;    11           41     -      48
        3         MPC       5        FSD     NOT       ;    12
```

Figure 8-7: Extract from a typical configuration file

8.6.2 Definition of outputs

The outputs listed in the third column of the configuration file (see Figure 8-7) can have eleven possible values for MPC4 cards, as defined in Table 8-5.

Table 8-5: Output channel coding for the MPC4

Output number	Definition
0	Measurement Channel 1, Output 1
1	Measurement Channel 1, Output 2
2	Measurement Channel 2, Output 1
3	Measurement Channel 2, Output 2
4	Measurement Channel 3, Output 1
5	Measurement Channel 3, Output 2
6	Measurement Channel 4, Output 1
7	Measurement Channel 4, Output 2
8	Dual-channel 1 and 2, Output 1
9	Dual-channel 3 and 4, Output 1
10	Speed Channel 1, Output 1
11	Speed Channel 2, Output 1

8.6.3 Analog values (registers)

Possible settings are:

- *VAL Actual measured value*
  VAL is coded as a 16-bit signed value (see Figure 8-8) scaled as follows:
  - 0 represents 0% of the full-scale defined in the FSD register
  - 16383 (0x3FFF) represents 100% of the full-scale defined in the FSD register
  - 49152 (0xC000) represents −100% of the full-scale defined in the FSD register

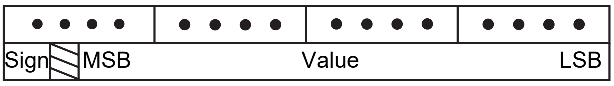


Figure 8-8: Signed 16-bit Integer

- *FSD Full-scale value*
  - FSD is coded as a 16-bit unsigned value (see Figure 8-9).

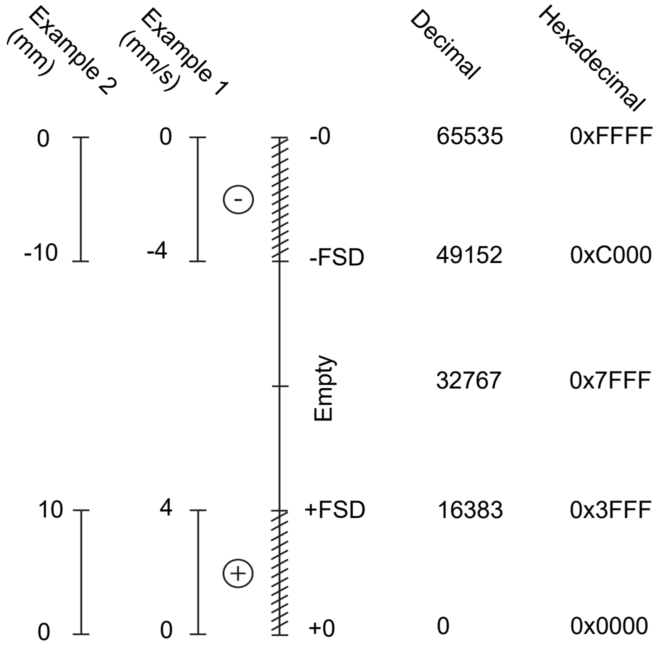


Figure 8-9: Analog Values

Example 1:

A value of 10000 is returned with a full-scale defined as 10.
(The physical unit is mm/s, but this information is not available from Modbus.)

The corresponding mechanical value is then calculated as follows:

(10000 / 16383) x 10 = 6.10 mm/s.

Example 2:

A value of 50000 is returned with a full-scale defined as 20.
The corresponding mechanical value is then calculated as follows:

(50000 - 65535) / 16383 x 20 = −18.96 mm.

NOTE: The two speed values (speed channel outputs 10 and 11 in Table 8-5) are coded differently. The scaled value is the measured RPM value divided by 4. **Therefore, to obtain RPM values, simply multiply the returned value by 4.**

8.6.4 Discrete values (coils, discrete inputs)

The possible settings are:

- STS Status (values) requested
- NOT Status (values) not available.

The map of all discrete values available for an output can be activated or not using the STS or NOT indication.

The eight discrete values available for the output will be mapped as shown in Table 8-6.

Table 8-6: Mapping of discrete values for the MPC4 channel status register

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	Invalid	Point is not defined	Point is defined
b1	A+	Channel not in Alarm+ condition	Channel in Alarm+ condition
b2	A−	Channel not in Alarm− condition	Channel in Alarm− condition
b3	D+	Channel not in Danger+ condition	Channel in Danger+ condition
b4	D−	Channel not in Danger− condition	Channel in Danger− condition
b5	SOK	OK system check failed	Sensor OK
b6		Reserved	
b7		Reserved	

Discrete values are not available individually (for example, you cannot request only the b3 bit). All 8 bits have to be requested.

NOTE: The implementation of the Modbus server for CPUM cards running firmware version 071 or later maps additional status bits available from the card to this channel status register (see Table 9-10).

8.6.5 Discrete values coded in an analog value register

The discrete values presented in Section 8.6.4 can also be accessed with Modbus functions 03 and 04 at register numbers after 1001 (see Table A-2 in Appendix A: MPC4 Modbus register definitions). The mapping is the same as the one defined in Table 8-6 but with the bits b8 to b15 set to zero.

8.6.6 Address map

The MPC4 card’s register definitions for Modbus can be found in the address map tables given in Appendix A: MPC4 Modbus register definitions. The Modbus register starting addresses can be determined using the information given in the tables.

8.7 AMC8 card

The information is this section applies to the implementation of the Modbus software interface for VM600 racks as used by CPUM cards running firmware version 067 or earlier.

Improved Modbus functionality is provided by CPUM cards running firmware version 071 or later. See 9 Setting up a Modbus connection (CPUM firmware version 071 or later) for more information.

The following sections describe the values or configuration parameters, either discrete or analog, that can be retrieved using the four Modbus functions. (See 8.4.3 Modbus functions supported by the VM600 for a list of these functions.)

8.7.1 Discrete values (coils)

Modbus function 01 is the read coil registers function. It is used to read the configuration of the alarms (discrete values), whether they are enabled or not. These configurations were made in the MPS Software in the **Processing and alarms** screen (for the channels) and under the **Alarms screen** (for the multi-channels).

The four alarm outputs that can be enabled are:

- Alert− Low
- Alert+ High
- Danger− Low
- Danger+ High.

8.7.2 Discrete values (discrete inputs)

Modbus function 02 is the read discrete inputs function. It is used to retrieve discrete values such as channel status (error and fail) and alarm register.

The five following registers define whether the system is in an Alert condition for each channel and multi-channel, or if there is a Global Channel OK Failure.

- Alarm+
- Alarm−
- Danger+
- Danger−
- Global Channel OK Fail.

The ten following registers specify whether there is an error or a standby state for each channel and multi-channel.

- ADC Error
- ADC Standby
- ADC PLL Lock Error
- ADC Transmission Error
- ADC Dynamic Configuration Error

- BIT Result Fail
- No Sample Error
- OK Error Fail
- Linearization Error
- Cold Junction Error.

The last two registers define whether each of the 16 Basic Functions and each of the 8 Advanced Functions were activated in the **Alarms Logical Combination** screen of the MPS Software.

- Basic Functions
- Advanced Functions.

8.7.3 Holding registers

Modbus function 03 is the read holding registers function. It is used to read analog values. In particular, it is used to read the current values on each channel of the AMC8 and the results of the logical combinations of alarms.

This function, as well as function 02 can be used to read the alarm status.

The meaning of the Input Registers is defined in Table 8-7.

Table 8-7: Read input register definitions

<table>
<tr><th>Register</th><th colspan="2">Definition</th></tr>
<tr><td>Processed Values</td><td colspan="2">The current value for each channel and multi-channel.</td></tr>
<tr><td rowspan="9">Logical Results</td><td colspan="2">The result of the logical combinations defined in the <b>Alarms Logical Combination</b> screen of the MPS for Basic Functions (BFs) from 1 to 16 and Advanced Functions (AFs) from 1 to 8 as follows:</td></tr>
<tr><td><b>Bit</b></td><td><b>Definition</b></td></tr>
<tr><td>0</td><td>BF1</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>15</td><td>BF15</td></tr>
<tr><td>16</td><td>AF1</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>23</td><td>AF8</td></tr>
<tr><td>24 to 31</td><td>0 (not used)</td></tr>
</table>

Table 8-7: Read input register definitions (continued)

<table>
<tr><th>Register</th><th colspan="2">Definition</th></tr>
<tr><td rowspan="18">Alarm Status</td><td colspan="2">Defines the system status and alarms with the following bit definitions:</td></tr>
<tr><td><b>Bit</b></td><td><b>Definition</b></td></tr>
<tr><td>0</td><td>Alarm+</td></tr>
<tr><td>1</td><td>Alarm−</td></tr>
<tr><td>2</td><td>Danger+</td></tr>
<tr><td>3</td><td>Danger−</td></tr>
<tr><td>4</td><td>Global Channel OK Fail</td></tr>
<tr><td>5</td><td>ADC Error</td></tr>
<tr><td>6</td><td>ADC Standby</td></tr>
<tr><td>7</td><td>ADC PLL Lock Error</td></tr>
<tr><td>8</td><td>ADC Transmission Error</td></tr>
<tr><td>9</td><td>ADC Dynamic Configuration Error</td></tr>
<tr><td>10</td><td>BIT Result Fail</td></tr>
<tr><td>11</td><td>No Sample</td></tr>
<tr><td>12</td><td>OK Error Fail</td></tr>
<tr><td>13</td><td>Linearization Error</td></tr>
<tr><td>14</td><td>Cold Junction Error</td></tr>
<tr><td>15</td><td>Reserved</td></tr>
</table>

8.7.4 Input registers

Modbus function 04 is the read input registers function. It is used to read configuration values such as minimum and maximum displayed values and alarm levels for each channel and multi-channel using the defined Output Unit.

The complete list of holding registers is shown below:

- Minimum Displayed Value
- Maximum Displayed Value
- Alert− Low Level
- Alert+ High Level
- Danger− Low Level
- Danger+ High Level
- Output Unit
    - Bit 0: User defined
    - Bit 1: Degrees Kelvin
    - Bit 2: Degrees Celsius
    - Bit 3: Degrees Fahrenheit
    - Bits 4 to 15: Not used.

8.7.5 Address map

The AMC8 card’s register definitions for Modbus can be found in the address map tables given in Appendix B: AMC8 Modbus register definitions. The Modbus register starting addresses can be determined using the information given in the appendix.

9 SETTING UP A MODBUS CONNECTION (CPUM FIRMWARE VERSION 071 OR LATER)

9.1 Introduction

This chapter describes the implementation of the Modbus software interface for VM600 racks as used by CPUM cards running firmware version 071 or later. This is the latest implementation of the Modbus software interface for the VM600 and includes extra registers to allow more information to be read using Modbus.

All Modbus registers are available for CPUM cards running firmware version 071 or later. It is also backwards compatible with the existing Modbus server for CPUM cards running firmware version 067 or earlier.

NOTE: Even if you are using the latest Modbus software interface (for CPUM cards running firmware version 071 or later), you should also read all of 8 Setting up a Modbus connection (CPUM firmware version 067 or earlier).
The information given in this chapter is in addition to the information already given for CPUM cards running firmware version 067 or earlier.

In practice, the improved Modbus server is much more flexible. Extensions have been added to the Modbus registers so that additional data can be provided by the VM600 system. This extra data includes:

- Basic and advanced functions for the MPC4
- Board level status information for the MPC4
- Board level status information for the AMC8.

However, the Modbus function codes 01 (Read coil status) and 02 (Read input status) used for reading discrete (individual one-bit) inputs have not changed.

See Appendix A: MPC4 Modbus register definitions for detailed MPC4 Modbus register mapping information and Appendix B: AMC8 Modbus register definitions for detailed AMC8 Modbus register mapping information.

9.1.1 Modbus functions supported by the VM600 (version 071 or later)

In addition to the existing Modbus functions (see 8.4.3 Modbus functions supported by the VM600), CPUM cards running firmware version 071 or later provide the additional Modbus functions shown in Table 9-1.

Table 9-1: Additional Modbus functions supported

Modbus function code	Function description
05 (0x05)	Write single coil
06 (0x06)	Write single register
15 (0x0F)	Write multiple coils
16 (0x10)	Write multiple registers

NOTE: If any unsupported Modbus function is requested, an exception code “01” will be returned to indicate the use of an illegal function.
If a value is requested that is out of the range of what is configured in the VM600, the exception code “02” will be returned to indicate the use of an illegal address.

9.2 Definition of registers

The configuration has changed considerably for firmware version 071 or later. The five separate configuration files that were used previously have been replaced by a single configuration file:

`modbusDefault.cfg` For all Modbus communications and configuration.

As before, the configuration file is a text file and can be read or modified using any text editor as required. This file can be modified, for example, by editing it with a VT100 terminal (or emulator from the Windows environment). See 2.3 Modbus configuration files for information on using the vi editor to modify files.

However, because of the size of this configuration file, it is only practical to make simple changes using the vi editor. For more significant changes, it is recommended to make the changes to the `modbusDefault.cfg` using a more advanced editor program on a computer. The edited configuration file can then be sent to the CPUM using a software utility called `ModbusConfigSender.exe` (developed by and freely available from Meggitt SA).

NOTE: In general, the `modbusDefault.cfg` file is only required to be edited if you need to change the communication parameters.

The content of the configuration file is not case-sensitive and is organised into the following sections: [RTUx], [TCP], [GLOBAL] and [MAPPING].

9.2.1 The RTUx section

The RTUx section can have one or more sections called [RTU1], [RTU2], [RTU3].

Each [RTUx] section shall have the settings shown in Table 9-2.

Table 9-2: RTUx section settings

RTUx section setting	Default value	Allowed values
ENABLE	N	Y/N
ADDRESS	1	1 to 247
DEVICE	RS	A/B/RS All of the ports are located on the IOC-N in the rear of the VM600 rack.
SPEED	19200	1200 to 11500
PARITY	Even	Even, None, Odd
STOP	1	1 or 2
DATABITS	8	7 or 8

9.2.2 The TCP section

The TCP section shall have the settings shown in Table 9-3.

Table 9-3: TCP section settings

TCP section setting	Default value	Allowed values
ENABLE	N	Y/N
PORT	502	0 to 65535

9.2.3 The GLOBAL section

Although 32-bit values were not part of the original Modbus specification, extensions have been implemented which allow 32-bit values to be retrieved using Modbus and two consecutive registers. The byte-order of data contained in the two consecutive registers, however, has not been standardised, so the user must specify the byte-order they prefer. The user has the ability to select 32-bit floating-point values and 32-bit long word values to be retrieved, in different formats.

NOTE: The GLOBAL section is optional.

The 32-bit floating point values can have the settings shown in Table 9-4. If this global section setting is missing, the floating point values will have the default order type FM1 (the most commonly encountered type).

Table 9-4: GLOBAL section settings

GLOBAL section setting	Default value	Allowed values
DEFAULT_FLOAT_ORDER	FM1	FB/FL/FM1/FM2
Where: FB: Float (32-bit) ordered as ‘big endian’. For example, ABCD. FL: Float (32-bit) ordered as ‘little endian’. For example, DCBA. FM1: Float (32-bit) ordered as ‘mixed endian 1’. For example, CDBA. FM2: Float (32-bit) ordered as ‘mixed endian 2’. For example, BADC.		

The 32-bit long word values can have the settings shown in Table 9-5. If this section is missing, the long word values will have the default order type LB.

Table 9-5: GLOBAL section settings

GLOBAL section setting	Default value	Allowed values
DEFAULT_LONG_ORDER	LB	LB/LL/LM1/LM2
Where: LB: Long word (32-bit) ordered as ‘big endian’. For example, ABCD. LL: Long word (32-bit) ordered as ‘little endian’. For example, DCBA. LM1: Long word (32-bit) ordered as ‘mixed endian 1’. For example, CDBA. LM2: Long word (32-bit) ordered as ‘mixed endian 2’. For example, BADC.		

9.2.4 The MAPPING section

The MAPPING section has the following syntax:

Address:Function[:DataType[:Min:Max]] = SlotNr:CardType:ValueName

NOTE: The section Min:Max is optional and is used only for scaled values.

The possible settings for each of the parameters are shown in Table 9-6.

Table 9-6: MAPPING section settings

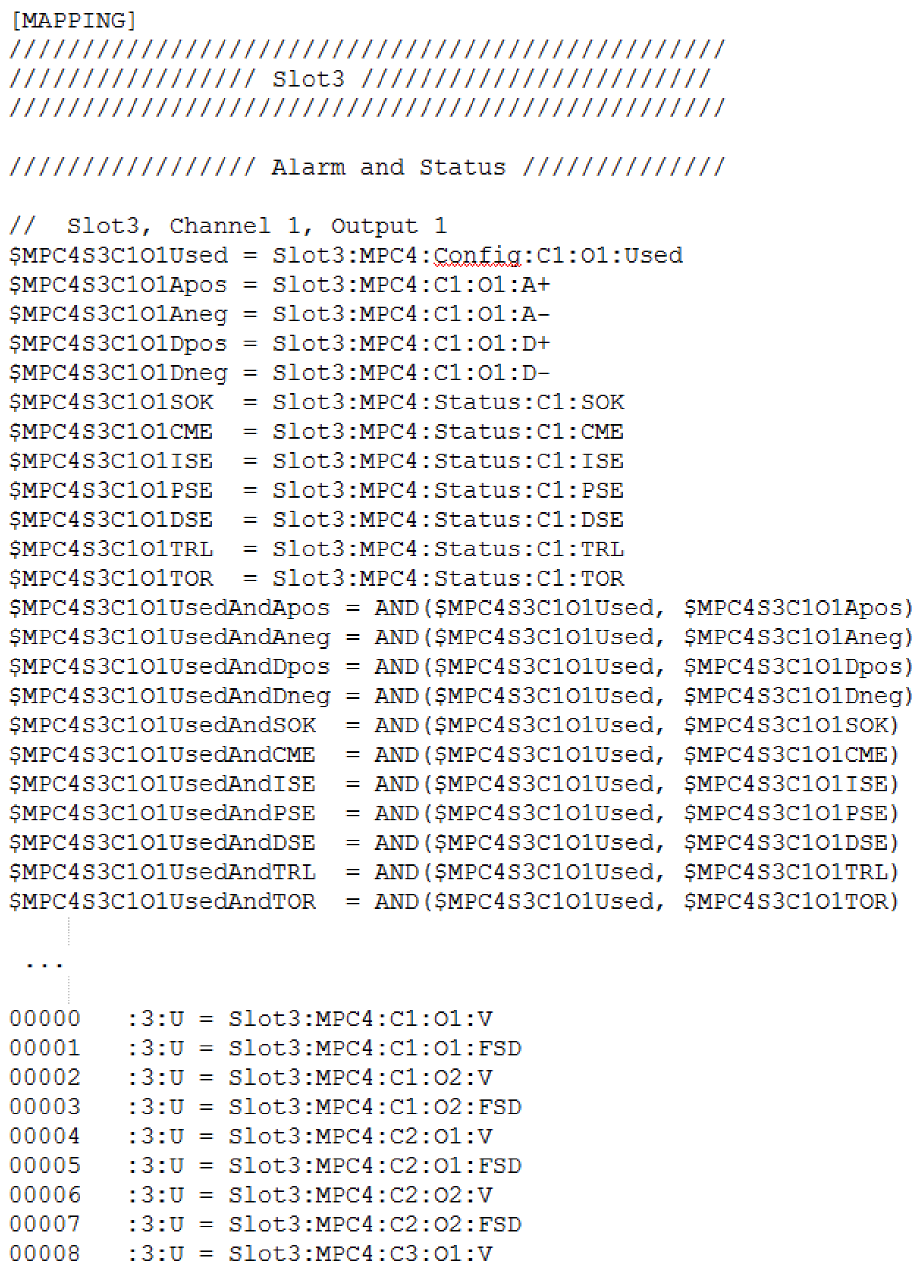
MAPPING section setting	Allowed values	
Address	0 to 65535	
Function	Modbus functions 01 to 06, 15 (0x0F) and 16 (0x10)	
DataType	One of the following:	
	B	Bit (1-bit)
	F	Float (32-bit). Ordering is of the type FM1 by default (CDBA). See Note 1
	FB	Float (32-bit). Ordered as ‘big endian’ (ABCD)
	FL	Float (32-bit). Ordered as ‘little endian’ (DCBA)
	FM1	Float (32-bit). Ordered as ‘mixed endian 1’ (CDBA)
	FM2	Float (32-bit). Ordered as ‘mixed endian 2’ (BADC)
	L	Long word (32-bit). Ordering is by default of the type LB (ABCD). See Note 2
	LB	Long word (32-bit). Ordered as ‘big endian’ (ABCD)
	LL	Long word (32-bit). Ordered as ‘little endian’ (DCBA)
	LM1	Long word (32-bit). Ordered as ‘mixed endian 1” (CDBA)
	LM2	Long word (32-bit). Ordered as ‘mixed endian 2’ (BADC)
	U	16-bit unsigned (0x0000 to 0xFFFF)
Min:Max	The section Min:Max is optional and shall be used only for scaled values. See Note 3	
SlotNr	This is dependent on the rack. For example, on the current VM600 System with the ABE 040/42 rack, the measurement card slots are numbered form 3 to 14.	

Table 9-6: MAPPING section settings (continued)

MAPPING section setting	Allowed values	
CardType	One of the following:	
	AMC8	For the Analogue Monitoring Card with 8 channels
	MPC4	For the Machinery Protection Card with 4 channels
	CMC16	For the Condition Monitoring Card with 16 channels
	XMC16	For the Extended Monitoring Card with 16 channels

Notes:
1. The DataType F can be set to FB, FL, FM1 or FM2. See 9.2.3 The GLOBAL section.
2. The DataType L can be set to LB, LL, LM1 or LM2. See 9.2.3 The GLOBAL section.
3. For every scaled value, the minimum and maximum values shall be readable by Modbus in the floating-point format. See 9.4.1 Analog values (registers).

Figure 9-1 is an extract from a typical `modbusDefault.cfg` file that shows the organisation of some data.



```
[MAPPING]
/////////////////////////////////////////////////
///////////////// Slot3 ////////////////////////
/////////////////////////////////////////////////

///////////////// Alarm and Status //////////////

//  Slot3, Channel 1, Output 1
$MPC4S3C1O1Used = Slot3:MPC4:Config:C1:O1:Used
$MPC4S3C1O1Apos = Slot3:MPC4:C1:O1:A+
$MPC4S3C1O1Aneg = Slot3:MPC4:C1:O1:A-
$MPC4S3C1O1Dpos = Slot3:MPC4:C1:O1:D+
$MPC4S3C1O1Dneg = Slot3:MPC4:C1:O1:D-
$MPC4S3C1O1SOK  = Slot3:MPC4:Status:C1:SOK
$MPC4S3C1O1CME  = Slot3:MPC4:Status:C1:CME
$MPC4S3C1O1ISE  = Slot3:MPC4:Status:C1:ISE
$MPC4S3C1O1PSE  = Slot3:MPC4:Status:C1:PSE
$MPC4S3C1O1DSE  = Slot3:MPC4:Status:C1:DSE
$MPC4S3C1O1TRL  = Slot3:MPC4:Status:C1:TRL
$MPC4S3C1O1TOR  = Slot3:MPC4:Status:C1:TOR
$MPC4S3C1O1UsedAndApos = AND($MPC4S3C1O1Used, $MPC4S3C1O1Apos)
$MPC4S3C1O1UsedAndAneg = AND($MPC4S3C1O1Used, $MPC4S3C1O1Aneg)
$MPC4S3C1O1UsedAndDpos = AND($MPC4S3C1O1Used, $MPC4S3C1O1Dpos)
$MPC4S3C1O1UsedAndDneg = AND($MPC4S3C1O1Used, $MPC4S3C1O1Dneg)
$MPC4S3C1O1UsedAndSOK  = AND($MPC4S3C1O1Used, $MPC4S3C1O1SOK)
$MPC4S3C1O1UsedAndCME  = AND($MPC4S3C1O1Used, $MPC4S3C1O1CME)
$MPC4S3C1O1UsedAndISE  = AND($MPC4S3C1O1Used, $MPC4S3C1O1ISE)
$MPC4S3C1O1UsedAndPSE  = AND($MPC4S3C1O1Used, $MPC4S3C1O1PSE)
$MPC4S3C1O1UsedAndDSE  = AND($MPC4S3C1O1Used, $MPC4S3C1O1DSE)
$MPC4S3C1O1UsedAndTRL  = AND($MPC4S3C1O1Used, $MPC4S3C1O1TRL)
$MPC4S3C1O1UsedAndTOR  = AND($MPC4S3C1O1Used, $MPC4S3C1O1TOR)

 ...

00000   :3:U = Slot3:MPC4:C1:O1:V
00001   :3:U = Slot3:MPC4:C1:O1:FSD
00002   :3:U = Slot3:MPC4:C1:O2:V
00003   :3:U = Slot3:MPC4:C1:O2:FSD
00004   :3:U = Slot3:MPC4:C2:O1:V
00005   :3:U = Slot3:MPC4:C2:O1:FSD
00006   :3:U = Slot3:MPC4:C2:O2:V
00007   :3:U = Slot3:MPC4:C2:O2:FSD
00008   :3:U = Slot3:MPC4:C3:O1:V
```

Figure 9-1: Extract from a typical `modbusDefault.cfg` configuration file showing the mapping syntax

9.3 Error handling

In addition to the Modbus server error handling already described (see 8.4.5 Error handling), the behaviour of CPUM cards running firmware version 071 or later has been improved to allow the reporting of an additional communication error.

9.3.1 Communication errors

The latest implementation of the VM600 Modbus server provides an additional error exception code:

- If the client (master) requests information from a VM600 card that is unplugged, the exception code “06” will be returned. This code traditionally indicates a slave device busy error exception response. (The master should retransmit the message later when the slave is available again.)

NOTE: When an error occurs, the server (slave) returns a message response that includes an exception code. To distinguish between the message request and message response, the MSB (most significant byte) of the function code is set to “1” in the message response.

NOTE: Attempts to communicate with (access) an unplugged card will result in the following communication error: the exception code “06” will be returned.

9.4 MPC4 card

In addition to the existing MPC4 functionality (see 8.4.3 Modbus functions supported by the VM600), CPUM cards running firmware version 071 or later provide additional functionality.

9.4.1 Analog values (registers)

It is possible to scale analog values to 16-bit unsigned values (the U DataType). See 8.6.3 Analog values (registers) for more information). Scaling is set using the Min:Max syntax described in 9.2.4 The MAPPING section.

Min is the minimum value corresponding to 0x0000 in the Modbus register and Max is the maximum corresponding to 0xFFFF in the Modbus register. Values outside the Min:Max range are clipped to 0x0000 and 0xFFFF, as necessary.

Each channel output's minimum and maximum values are fully independent of other minimum and maximum values used by the VM600 system.

9.4.1.1 Reading minimum and maximum values

It is possible to read the minimum and maximum values configured for a channel using Modbus:

Address:Function=MIN:Address:Function

or

Address:Function=MAX:Address:Function

However, they can only be read as floating point numbers (the F DataType). The following example helps to illustrate the process.

Example:

Configuration data from the mapping section of a typical configuration file:

```
...
30003:4:U:-50.50:250.0=Slot10:AMC8:C4:V
30004:4:F=MAX:30003:4
30006:4:F=MIN:30003:4
...
```

The first line of this data is analysed in Table 9-7, the second line in Table 9-8 and the third line in Table 9-9.

Table 9-7: Reading minimum and maximum values example – first line

Data	Syntax	Description
30003	Address	Modbus address
4	Function	Modbus function 04
U	DataType	16-bit unsigned data type
−50.50	Min	Minimum FSD
250	Max	Maximum FSD

Table 9-7: Reading minimum and maximum values example – first line (continued)

Data	Syntax	Description
=		Equals
10	SlotNr	Slot number
AMC8	CardType	Analogue Monitoring Card with 8 channels
C4		Channel
V		Data value

As can be seen from the analysis in Table 9-7, this line of the configuration file is used to configure the channel 4 vibration data value of the AMC8 card located in slot 10 of the rack.

This parameter is available at the Modbus address 30003 as a 16-bit unsigned value by using the Modbus function 04.

The parameter is a scaled value with a real-value range from −50.50 to 250.00. This real-value range corresponds to the 16-bit unsigned (see 8.6.3 Analog values (registers)) value as follows:

The minimum FSD value of −50.50 is encoded as 0x0000 in the Modbus register

The maximum FSD value of 250 is encoded as 0xFFFF in the Modbus side.

As these are the minimum and maximum values, real-world values less than −50.50 will be clipped to 0x0000 and values greater than 250 will be clipped to 0xFFFF.

NOTE: Unlike firmware version 067 or earlier, CPUM cards running firmware version 071 or later support non-symmetrical scaling values. That is, it is no longer necessary for the magnitudes of the minimum FSD and the maximum FSD to be equal.

Table 9-8: Reading minimum and maximum values example – second line

Data	Syntax	Description
30004	Address	Modbus address
4	Function	Modbus function
F	DataType	Data type
=		Equals
MAX		
30003		
4		

Table 9-8 illustrates reading the maximum channel scaling value using Modbus. The value is coded as an F data type (and therefore only available as floating) so it requires two consecutive 16-bit addresses. The values are available using Modbus function 04 at addresses 30004 and 30005.

Table 9-9: Reading minimum and maximum values example – third line

Data	Syntax	Description
30006	Address	Modbus address
4	Function	Modbus function
F	DataType	Data type
=		Equals
MIN		
30003		
4		

Table 9-9 illustrates reading the minimum channel scaling value using Modbus. The value is coded as an F data type (and therefore only available) as floating so it requires two consecutive 16-bit addresses. The values are available using Modbus function 04 at addresses 30006 and 30007.

9.4.1.2 Constant values

It is possible to set Modbus addresses to constant values. Constants can be set for the following data types:

- B (1-bit value)
- F (32-bit floating value)
- U (16-bit unsigned value).

Examples:

Data from the mapping section of a typical configuration file:

```
30004:1:B=1
```

Valid values for data type B are 0 or 1.

```
30005:4:F=10.0
```

Valid values for data type F can be any 32-bit floating-point value.

```
30004:4:U=1234
```

Valid values for data type U are between 0 (0x0000) and 65535 (0xFFFF).

9.4.1.3 Bit variables

It is now possible for the user to define bit variables.

9.4.1.3.1 Bit variable names

Bit variable names must consist of the letters ‘a-Z’, the digits ‘0-9’ and the underscore character ‘_’. No other special characters are allowed and bit variable names are not case sensitive. The rules for naming bit variables are:

Every bit variable name must have a ‘$’ prefix

The variable name cannot begin with a digit

The maximum length is 40 characters

Examples:

Valid bit variable names:

```
$abcd102
$Vibration_channel_2
$Easy
$OK_SOK_VALUE234
```

Invalid bit variable names:

```
$Channel1?!!!
$Channel 0
$3Channel
```

9.4.1.3.2 Bit variable operations

A number of operations can be performed on bit variables.

Assigning bit values. For example:

```
$Variable = SlotNr:CardType:ValueName
```

AND, OR, NOT and XOR bit operations. For example:

```
$a=AND($b,$c,$d)
$o=OR($o1,$o2,$o3,$o4,$o5,$o6,$o7,$o8,$o9,$o10)
$x=NOT($z)
```

The rules for bit operations on user defined variables are:

- One line can have only one operator
- For AND, OR and XOR operations, from 2 to 16 arguments are allowed
- For the NOT operation, only one argument is allowed
- A comma must be used as a separator between the variables
- A variable or a list of them has to be in parentheses
- The equations will be evaluated from top to down
- Recursive definitions (for example, `$x=NOT($x)`) are not supported.

Packing bits to a 16-bit unsigned integer is a special function. For example:

```
Address:Function:U=PACK($a,$b,$c,$d,$e,$f,$g,$h,$i,$j,$k,$l $m, 0, 0, 1)
```

As shown above, constants (that is, 0 or 1) can also be used.

The rules packing user defined variables are:

- The first parameter passed is the MSB (`$a` in the above example) and the last parameter passed is the LSB (`$p` in the above example)
- Constants 0 and 1 are allowed
- A comma must be used as a separator between the variables
- A list of variables has to be in parentheses
- Valid functions are 3 or 4.

Unpacking bits from a 16-bit unsigned integer is a special function. For example, to unpack bits from a 16-bit unsigned integer:

```
Address:Function:B=UNPACK(address, function, index)
$bitvariable=UNPACK(address, function, index)
```

Where:

- Address gives the register for the specified function code
- Function gives the Modbus function code
- The index gives the position in the 16-bit unsigned integer and should be between 1 and 16. The value 1 corresponds to the LSB.
- Valid Modbus functions for accessing the unpack operation are functions 1 and functions 2.

Assigning values to Modbus addresses: For example:

```
Address:Function:B =$Value
```

As the data type is B, only bit values are allowed.

Assigning constants to bit variable. For example:

```
$Variable = 0
```

The valid values are 0 and 1.

9.4.2 Discrete values (coils, discrete inputs)

In addition to the existing Modbus discrete values (see 8-14), CPUM cards running firmware version 071 or later provide additional values.

9.4.2.1 Channel status register

The discrete values available for the MPC4 channel status register are mapped as shown in Table 9-10. (This table includes the eight discrete values available with CPUM cards running firmware version 067 or earlier. See Table 8-6 in 8.6.2 Definition of outputs.)

Table 9-10: Mapping of discrete values for the MPC4 channel status register

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	Invalid	Point is not defined	Point is defined
b1	A+	Channel not in Alarm+ condition	Channel in Alarm+ condition
b2	A−	Channel not in Alarm− condition	Channel in Alarm− condition
b3	D+	Channel not in Danger+ condition	Channel in Danger+ condition
b4	D−	Channel not in Danger− condition	Channel in Danger− condition
b5	SOK	OK system check failed	Sensor OK
Additional available status information added to this register:			
b6	CME	No common mode rejection error	Common mode rejection error
b7	ISE	No input saturation error	Input saturation error
b8	PSE	No PGA saturation error	PGA saturation error
b9	DSE	No DSP saturation error	DSP saturation error
b10	TRL	No MPC track lost error	MPC track lost error
b11	TOR	No track out of range error	Track out of range error
b12 to b15	Not used		

NOTE: Bits b6 to b15 are not accessible using Modbus functions 01 or 02.

Discrete values are not available individually. For example, you cannot request only bit b3, that is, all eight bits have to be requested.

9.4.2.2 Speed channel status register

The discrete values available for the MPC4 speed status register are mapped as shown in Table 9-11.

Table 9-11: Mapping of discrete values for the MPC4 speed channel status register

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	Invalid	Point is not defined	Point is defined
b1	A+	Channel not in Alarm+ condition	Channel in Alarm+ condition
b2	A−	Channel not in Alarm− condition	Channel in Alarm− condition
b3	Not used		
b4	Not used		
b5	SOK	OK system check failed	Sensor OK

Table 9-11: Mapping of discrete values for the MPC4 speed channel status register

Bit	Status	Meaning when not set (0)	Meaning when set (1)
Additional available status added to this register:			
b6	SOL	No speed out of limit error	Speed out of limit error
b7	ERR	No error	Error
b8 to b15	Not used		

NOTE: Bits b6 to b15 are not accessible using Modbus functions 01 or 02.

Discrete values are not available individually. For example, you cannot request only bit b3, that is, all eight bits have to be requested.

9.4.3 Card status

CPUM cards running firmware version 071 or later provide status information for the MPC4 card. Three 16-bit status registers have been mapped to access the card status.

9.4.3.1 Status register 1 – basic and advanced functions

The basis functions (BF) and advanced functions (AF) available are as shown in Table 9-12.

Table 9-12: Mapping of card status register 1

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	BF1	Basic function 1 not active	Basic function 1 active
b1	BF2	Basic function 2 not active	Basic function 2 active
b2	BF3	Basic function 3 not active	Basic function 3 active
b3	BF4	Basic function 4 not active	Basic function 4 active
b4	BF5	Basic function 5 not active	Basic function 5 active
b5	BF6	Basic function 6 not active	Basic function 6 active
b6	BF7	Basic function 7 not active	Basic function 7 active
b7	BF8	Basic function 8 not active	Basic function 8 active
b8	AF1	Advanced function 1 not active	Advanced function 1 active
b9	AF2	Advanced function 2 not active	Advanced function 2 active
b10	AF3	Advanced function 3 not active	Advanced function 3 active
b11	AF4	Advanced function 4 not active	Advanced function 4 active
b12 to b15	Not used		

Modbus address: 1144 + ((slot - 3) * 3) + 0.

Modbus function: 03 or 04.

This information is accessed using the “Basic and advanced function” channel as defined in Table A-2 (available at addresses 1144, 1147, 1150, …, 1178).

9.4.3.2 Status register 2 – diagnostic and common status #1

As shown in Table 9-13.

Table 9-13: Mapping of card status register 2

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0 (= LSB)	CMF	No common monitoring failure	Common monitoring failure
b1	COF	No common sensor OK failure	Common sensor OK failure
b2	CPE	No common processing error	Common processing error
b3	CIE	No common input signal error	Common input signal error
b4	CA	No common alarm	Common alarm
b5	CD	No common danger	Common danger
b6	CSOL	No common speed out of limit	Common speed out of limit
b7	CTL	No common track lost	Common track lost
b8	CTOR	No common track out of track	Common track out of track
b9	CDSE	No common DSP saturation error	Common DSP saturation error
b10	CISE	No common input saturation error	Common input saturation error
b11	CCME	No common CMR error	Common CMR error
b12	SL	No status latched	Status latched
b13	DTM	Trip multiply input not active	Trip multiply input active
b14	DBP	Danger bypass input not active	Danger bypass input active
b15	AR	Alarm reset input not active	Alarm reset input active

Modbus address: 1144 + ((slot - 3) * 3) + 1.

Modbus function: 03 or 04.

This information is accessed using the “Board status #1” channel as defined in Table A-2 (available at addresses 1145, 1148, 1151, …, 1179).

9.4.3.3 Status register 3 – diagnostic and common status #2

As shown in Table 9-14.

Table 9-14: Mapping of card status register 3

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	MCR	MPC configuration not running	MPC configuration running
b1 to b15	Not used		

Modbus address: 1144 + ((slot - 3) * 3) + 2.

Modbus function: 03 or 04.

This information is accessed using the “Board status #2” channel as defined in Table A-2 (available at addresses 1146, 1149, 1152, ..., 1180).

9.4.4 Address map

The MPC4 card’s register definitions for Modbus can be found in the address map tables given in Appendix A: MPC4 Modbus register definitions. The Modbus register starting addresses can be determined using the information given in the appendix.

9.5 AMC8 card

In addition to the existing AMC8 functionality (see 8.7 AMC8 card), CPUM cards running firmware version 071 or later provide additional functionality.

9.5.1 Read holding registers

To be consistent with the MPC4 cards behaviour, a 16-bit register has been added for each channel and each multi-channel output that maps information defined in Table 9-15.

Table 9-15: Mapping of holding registers for AMC8 channel status

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	A−	Channel not in Alarm− condition	Channel in Alarm− condition
b1	A+	Channel not in Alarm+ condition	Channel in Alarm+ condition
b2	D−	Channel not in Danger− condition	Channel in Danger− condition
b3	D+	Channel not in Danger+ condition	Channel in Danger+ condition
b4 to b15	Not used		

Address offset (Aoff): from 38 to 49.

Modbus function: 03 or 04.

This information is accessed using the channels from “Channel 1” to “Channel 8” as defined in Table B-3 (available at addresses 4903, 4904, …, 7730).

Where 4903 is the first channel of the first slot and 7730 is the last channel of the last slot.

9.5.2 Card status

CPUM cards running firmware version 071 or later provide additional status information for the AMC8 card. A single 16-bit status register has been mapped to access the card status.

9.5.2.1 AMC8 status register 1- basic and advanced functions

As shown in Table 9-16.

Table 9-16: Mapping of card status register 1 for the MPC4

Bit	Status	Meaning when not set (0)	Meaning when set (1)
b0	ACNR	AMC configuration running	AMC configuration not running
b1	AR	Alarm reset input not active	Alarm reset input active
b2	DBP	Danger bypass input not active	Danger bypass input active
b3	SL	No status latched	Status latched
b4 to b15	Not used		

Address offset (Aoff): 50

Modbus function: 03 or 04.

This information is accessed using the “Basic and advanced function” channel as defined in Table A-2 (available at addresses 4915, 5171, 5427, …, 7731).

10 SETTING UP A PROFINET CONNECTION (CPUM FIRMWARE VERSION 081 OR LATER)

10.1 Introduction

This chapter describes the use of the PROFINET software interface for VM600 racks as used by CPUM cards running firmware version 081 or later. All of the registers and data that are available through Modbus can be made available through PROFINET for CPUM cards running this version of the firmware.

This manual describes the PROFINET software interface that supports industrial Ethernet communications interface for a VM600 rack containing the following cards:

- MPC4
- AMC8.

See Appendix C: MPC4 PROFINET slot definitions for detailed MPC4 PROFINET mapping information and Appendix D: AMC8 PROFINET slot definitions for detailed AMC8 PROFINET register information.

The table of Appendix E: Symbol names includes a list of predefined text description tags for use with a VM600 system. These can be used as symbol names to more easily identify and understand the contents of a register (see Figure 10-17).

This chapter includes an example that illustrates how to include a VM600 rack in a PROFINET network using Siemens SIMATIC STEP 7 and a specific SIMATIC S7 controller. The principles shown should be the same for any PROFINET system.

10.2 PROFINET

PROFINET is a standard for industrial Ethernet defined by PROFINET International (PI), an umbrella organisation responsible for both the PROFIBUS and PROFINET protocols.

PROFINET is based on industrial Ethernet which has been used in manufacturing plants for many years and brings Ethernet down to the field level so that you can directly interconnect to field devices (such as sensors and actuators). In most cases, existing network infrastructure components (network switches, cables, connectors) are compatible and can be used with PROFINET.

Visit the PI website for the most up-to-date information on PROFINET:
http://www.profibus.com

10.2.1 PROFINET protocols

PROFINET uses a modular concept and distinguishes between different types of communication for satisfying various response time requirements:

- TCP/IP for PROFINET CBA (Component Based Automation)
  No specific performance requirements: cycle times are in the range of 100 ms.
  Typically used for configuration and acyclic (that is, unscheduled and on-demand) read/write operations.
- Real-time (RT) protocol for PROFINET CBA and PROFINET IO
  Increased performance requirements: cycle times are up to 10 ms.
  Typically used for standard cyclic (that is, scheduled and repetitive) data transfer and alarms.
- Isochronous Real-time (IRT) PROFINET IO
  Maximum performance requirements: cycle times are less than 1 ms.
  Typically used for real-time control applications.

In addition, PROFINET distinguishes between different types of communication according to the nodes to be connected:

- PROFINET CBA
  This is suitable for communication among controllers via TCP/IP as well as additional RT requirements and focuses on distributed automation systems (higher level).
- PROFINET IO
  This supports communication between controllers and IO devices with RT and IRT requirements and focuses on data exchange with programmable controllers (lower level).

NOTE: It is possible to use all types of communication in parallel on the same network, that is, PROFINET IO and PROFINET CBA can communicate at the same time on the same bus.

The VM600 CPUM implements PROFINET IO to support PROFINET communications.

10.2.2 VM600 PROFINET implementation

The PROFINET software interface implemented for CPUM cards running firmware version 081 or later builds on the existing Modbus interface described in 8.2 Modbus.

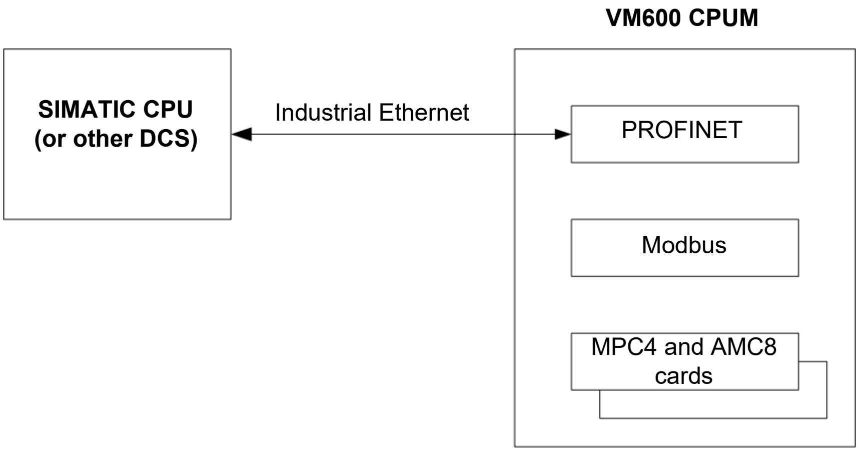


Figure 10-1: VM600 CPUM PROFINET implementation

As shown in Figure 10-1, the VM600 CPUM card includes an additional PROFINET layer above the existing Modbus server. Externally, this PROFINET layer uses the PROFINET IO protocol to handle all PROFINET communications over an industrial Ethernet interface. Internally, this PROFINET layer uses the existing Modbus interface to communicate with any cards in the VM600 rack.

Only the primary network interface (ETH1) of a CPUM card can be used for PROFINET communications. See 3.4.2.1 Example net.cfg file for further information.

NOTE: In order for PROFINET communications to work, only the primary network interface (ETH1) of the CPUM card can be used.

In addition, to enable and configure PROFINET communications, see 10.6 PROFINET configuration files for information.

10.3 PROFINET communication

Like other industrial Ethernet standards, such as Modbus TCP and Ethernet/IP, PROFINET uses TCP/IP and is part of the IEC 61158 (Industrial communication networks – Fieldbus specifications) and IEC 61784 (Digital data communications for measurement and control) standards.

Depending on the performance requirements of a PROFINET system, one of the following protocols can be used:

- TCP/IP for PROFINET CBA
- Real-time (RT) protocol for PROFINET CBA and PROFINET IO
- Isochronous Real-time (IRT) PROFINET IO.

Most PROFINET communication uses unmodified TCP/IP packets (ordinary Ethernet). This allows easy integration into existing networks, for example, in a factory or office.

The RT protocol bypasses TCP/IP and uses an optimised protocol stack to prioritise Ethernet packets. This is typically used to speed up communications with programmable controllers such as a PLC.

The IRT protocol uses an extension of the Ethernet stack to synchronise all communication devices. This allows high speed communication but does require customised hardware. The IRT protocol is not supported by the CPUM.

NOTE: Use of the IRT communication requires the relevant network area to be equipped with special IRT network switches.

10.3.1 Error handling

In addition to the Modbus server error handling already described (see 8.4.5 Error handling and 9.3 Error handling), the behaviour of CPUM cards running firmware version 081 or later has been improved to allow the reporting of additional communication errors.

10.3.1.1 Internal communication errors

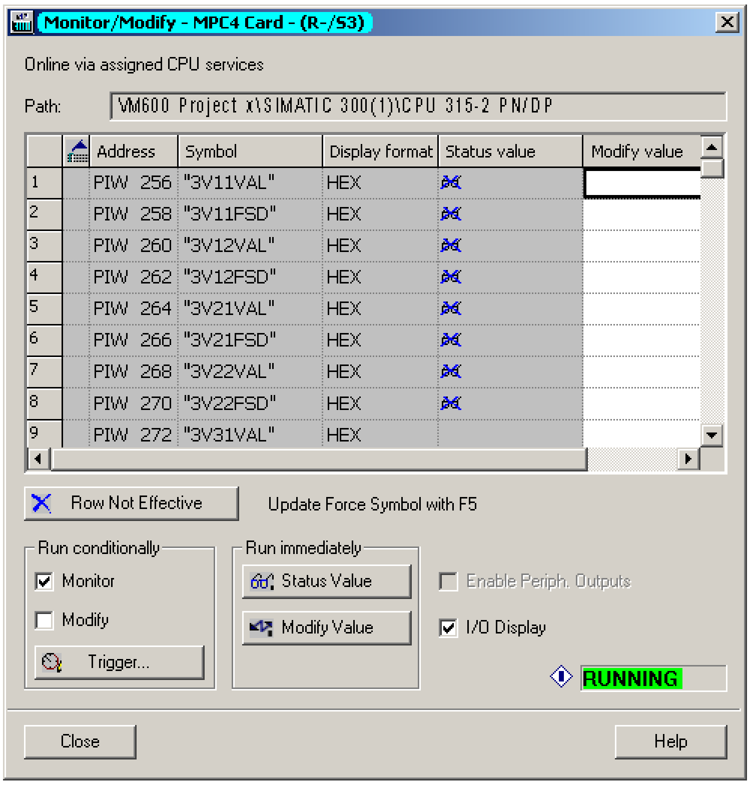
As the PROFINET layer uses the existing Modbus server to access the cards in the VM600 rack (see Figure 10-1), the internal communication errors reported will be as described in 8.4.5 Error handling and 9.3 Error handling.

10.3.1.2 External communication errors

The types of error handling implemented will depend on the PROFINET controller and software (DCS) that you are using.

NOTE: Attempts to communicate with (access) an unplugged card will result in a communication error being reported by the DCS.

For example, SIMATIC STEP 7 uses the data unavailable icon when it cannot obtain any valid data from an object (such as a VM600) to display. The use of this icon can be seen in the Status value column of Figure 10-2. (For comparison, Figure 10-17 shows the same window when data is available.)



Monitor/Modify - MPC4 Card - (R-/S3)

Online via assigned CPU services

Path: VM600 Project x\SIMATIC 300(1)\CPU 315-2 PN/DP

	Address	Symbol	Display format	Status value	Modify value
1	PIW 256	"3V11VAL"	HEX	✕	
2	PIW 258	"3V11FSD"	HEX	✕	
3	PIW 260	"3V12VAL"	HEX	✕	
4	PIW 262	"3V12FSD"	HEX	✕	
5	PIW 264	"3V21VAL"	HEX	✕	
6	PIW 266	"3V21FSD"	HEX	✕	
7	PIW 268	"3V22VAL"	HEX	✕	
8	PIW 270	"3V22FSD"	HEX	✕	
9	PIW 272	"3V31VAL"	HEX		

Update Force Symbol with F5

☑ Monitor

□ Modify

□ Enable Periph. Outputs

☑ I/O Display

RUNNING

Figure 10-2: SIMATIC Manager – the Monitor/Modify window when data is unavailable

10.3.1.3 System errors

See 8.4.5.2 System errors.

10.4 PROFINET IO

PROFINET IO uses cyclic data transfer to exchange data with programmable controllers over Ethernet. In fact, it can be thought of as PROFIBUS on Ethernet.

The data exchanged is organised as slots containing modules (that contain I/O points). The programmable controller and the device it is communicating with must understand the structure and meaning of the data being exchanged. This is accomplished using GSDML files (see 10.5 GSDML files).

PROFINET IO classifies devices depending on their role in a system. There are three possibilities:

- IO-Controller
  Devices that run a automation program to control a system or subsystem, for example, a Siemens SIMATIC S7-300 universal controller.
- IO-Device
  Devices distributed in a system or subsystem to perform sensing or actuator functions, for example, a VM600 rack and cards.
- IO-Supervisor
  A computer or human-machine interface, typically used for commissioning or monitoring operations.

A PROFINET IO system requires at least one IO-Controller and one IO-Device. Configurations consisting of one IO-Controller and multiple IO-Devices are the most commonly implemented (see 10.7 Configuring a VM600 system for operation as a PROFINET IO-Device).

10.5GSDML files

The integration of third-party devices into a PROFINET system depends on the use of GSDML files. This is identical to the concept used in PROFIBUS, which uses GSD files, except that in PROFINET, the system files are called GSDML files because they are XML-based.

The GSDML file is used to identify the basic operational characteristics of a PROFINET device, to aid interoperability and interchangeability. This makes it possible to have manufacturer-independent configuration tools. Typically, a GSDML file includes vendor information, timing information, the options or features supported and a list of the available I/O signals. A GSDML file is required for every PROFIBUS IO-Device.

Meggitt SA provides GSDML files for all PROFINET enabled devices that it supplies in order to describe the implementation of the IO-Device. For example:

- GSDML-V2.1-VM600-CPUM-20100624.xml.

NOTE: Contact your local Meggitt representative to obtain the most recent GSDML files available for Meggitt vibro-meter® IO-Devices.

See 10.7.3.4 Adding a vibro-meter IO-Device to SIMATIC STEP 7 for information on how to install (import) a GSDML file into the Siemens software.

For I/O data, the GSDML file describes the underlying structure of the cyclic input and output data transferred between the programmable controller and the IO-Device.
Any mismatch between the size and/or structure of the input and output data and the actual internal device structure should generate an alarm in the controller.

10.6 PROFINET configuration files

As the PROFINET software interface builds on the existing Modbus interface, the existing Modbus configuration file (`modbusDefault.cfg`) is still used, as shown in Figure 10-3.

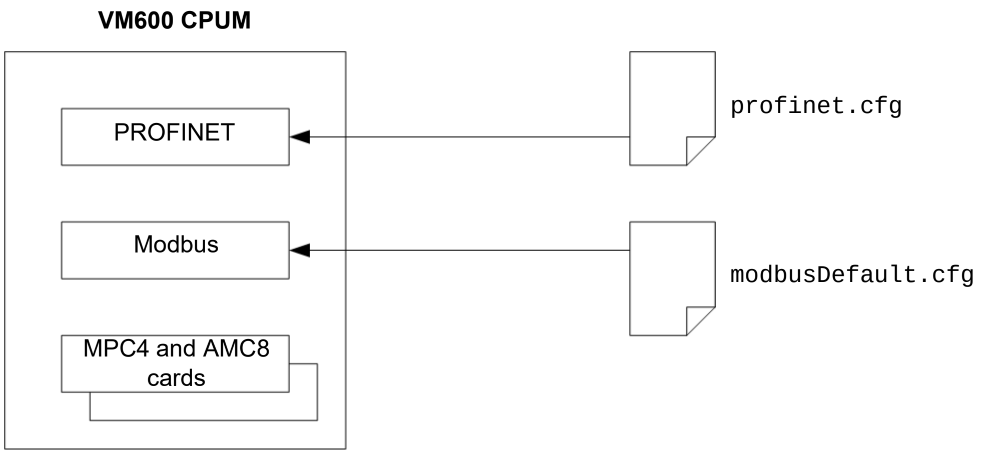


Figure 10-3: VM600 CPUM PROFINET configuration

However, to support the additional PROFINET layer, an additional PROFINET configuration file (`profinet.cfg`) is also required, as shown in Figure 10-3.

For more information on the Modbus configuration file, as used for PROFINET, see 10.6.1.1 modbusDefault.cfg file. For more information on the PROFINET configuration file, see 10.6.1.2 profinet.cfg file.

For general information on how to work with the Modbus and PROFINET configuration files, see 2.5 Working with configuration files and 2.6 CPUM card configurations.

10.6.1 CPUM cards running firmware version 081

10.6.1.1 modbusDefault.cfg file

Figure 10-4 is an extract from a typical `modbusDefault.cfg` configuration file. When used for PROFINET communications, this configuration file is basically the same as for Modbus but extended using additional PROFINET tags. The communication parameters are defined in the [RTUx], [TCP] and [GLOBAL] sections of this configuration file.

```
[GLOBAL]
DEFAULT_LONG_ORDER    = LL
DEFAULT_FLOAT_ORDER   = FL
IS_FULLY_COMPATIBLE   = NO  // YES/NO (default N)
IS_PROFINET_ACTIVATED = YES
```

Figure 10-4: Extract from a typical `modbusDefault.cfg` configuration file used for PROFINET communications showing the communication parameters

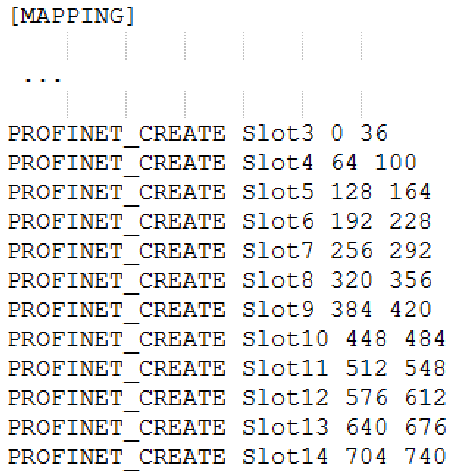
As show in Figure 10-4, for PROFINET communications, PROFINET is activated in the [GLOBAL] section using the IS_PROFINET_ACTIVATED flag:

- If this flag is set to YES (the true condition), the PROFINET support is activated
- If this flag is set to NO (the false condition), the PROFINET support is not activated
- If this flag is missing (not defined), the PROFINET support is not activated (that is, the default value is NO).

As show in Figure 10-5, for PROFINET communications, the PROFINET slots are defined at the end of the [MAPPING] section using the PROFINET_CREATE tag. At least one register should be defined by this tag. This means that the minimum PROFINET slot size is 2 bytes, while the maximum slot size is 128 bytes.

In general, the slot size should be a power of two. However, there are two exceptions:

- The MPC4 card has a slot size of 74 bytes
- The AMC8 card has a slot size of 102 bytes.



```
[MAPPING]

 ...

PROFINET_CREATE Slot3 0 36
PROFINET_CREATE Slot4 64 100
PROFINET_CREATE Slot5 128 164
PROFINET_CREATE Slot6 192 228
PROFINET_CREATE Slot7 256 292
PROFINET_CREATE Slot8 320 356
PROFINET_CREATE Slot9 384 420
PROFINET_CREATE Slot10 448 484
PROFINET_CREATE Slot11 512 548
PROFINET_CREATE Slot12 576 612
PROFINET_CREATE Slot13 640 676
PROFINET_CREATE Slot14 704 740
```

Figure 10-5: Extract from a typical `modbusDefault.cfg` configuration file used for PROFINET communications showing the PROFINET slots

As show in Figure 10-5, the PROFINET_CREATE tag uses the VM600 rack slot number (3 to 14), followed by the start and stop addresses of the Modbus registers where the card data is mapped, depending on the Modbus configuration.

For PROFINET communications, MPC4 registers must be defined as shown in Table 10-1 and AMC8 registers must be defined as shown in Table 10-2, as these tables (PROFINET slot definitions) correspond to the underlying PROFINET slot data structures defined in a GSDML file (see 10.5 GSDML files).

Table 10-1: MPC4 register definition format

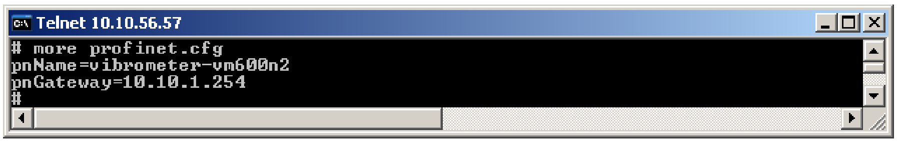
Type of channel	Required register format
Channel	Output1 Value, FSD, Status (Invalid, A+, A−, D+, D−, SOK) Output2 Value, FSD, Status (Invalid, A+, A−, D+, D−, SOK)
Dual channel	Value, FSD, Status (Invalid, A+, A−, D+, D−, SOK)
Speed	Value, Status (Invalid, A+, A−, SOK)
Basic and advanced functions	BF1 to BF8, AF1 to AF4
Common status	CMF, COF, CPR, CIE, CA, CD, CSOL, CTL, CTOR, CDSE, CISE, CCME, SL, TM, DBP, AR
Card status	MCR (MPC configuration running)

Table 10-2: AMC8 register definition format

Type of channel	Required register format
Channel	Output1 Value, FSD, Status (A−, A+, D−, D+, GOKF, ADERR, ADSTBY, ADLOCK, ADTXERR, ADCFGERR, BITF, NOSPL, OKF, LINERR, JERR)
Multiple channel	Value, FSD, Status (A−, A+, D−, D+, GOKF, ADERR, ADSTBY, ADLOCK, ADTXERR, ADCFGERR, BITF, NOSPL, OKF, LINERR, JERR)
Basic function	BF1 to BF16
Advanced function	AF1 to AF8
Channel and multiple channel alarm used	A− used, A+ used, D− used, D+ used
Card status	CSANR, CSSL, CSDB, CSAR

10.6.1.2 profinet.cfg file

Figure 10-6 shows a typical `profinet.cfg` configuration file. This configuration file defines the device name and gateway used for PROFINET communications.



```
# more profinet.cfg
pnName=vibrometer-vm600n2
pnGateway=10.10.1.254
#
```

Figure 10-6: A typical `profinet.cfg` file

10.7 Configuring a VM600 system for operation as a PROFINET IO-Device

This section includes an example of PROFINET based communications with a VM600 rack using Siemens SIMATIC STEP 7 software as an example of a PROFINET enabled DCS (see Figure 10-7). This example illustrates the concepts required for interfacing a VM600 rack to any DCS that supports PROFINET.

NOTE: In order for PROFINET communications to work, the information defined in the PROFINET configuration files (`modbusDefault.cfg` and `profinet.cfg`) on the CPUM card must match the information defined in the DCS (such as Siemens SIMATIC STEP 7).

10.7.1 Equipment used

10.7.1.1 Hardware

- VM600 rack containing the following cards:
  - CPUM card – running firmware version 081 or later
  - IOCN card
  - MPC4 card or cards
  - AMC8 card or cards.
- Siemens SIMATIC S7-300 universal controller (also known as a CPU or PLC):
  - CPU315-2 PN/DP (315-2EH13-0AB0) with external PSU.
- Computer running Microsoft® Windows XP.
- Netgear® 10/100 Mbps Fast Ethernet switch:
  - FS105.

10.7.1.2 Software

- The CPUM card must be running firmware version 081 or later.
- Siemens SIMATIC STEP 7 (version 5.4 service pack 5 was used).
- Visit the Siemens website for the most up-to-date information on the SIMATIC STEP 7 and their universal controllers:

http://www.automation.siemens.com/mcms/simatic-controller-/en/step7/Pages/Default.aspx

http://www.automation.siemens.com/mcms/programmable-logic-controller/en/simatic-s7-controller/Pages/Default.aspx

10.7.2 Prerequisites

Ensure that all of the hardware is connected as shown in the local network of Figure 10-7.

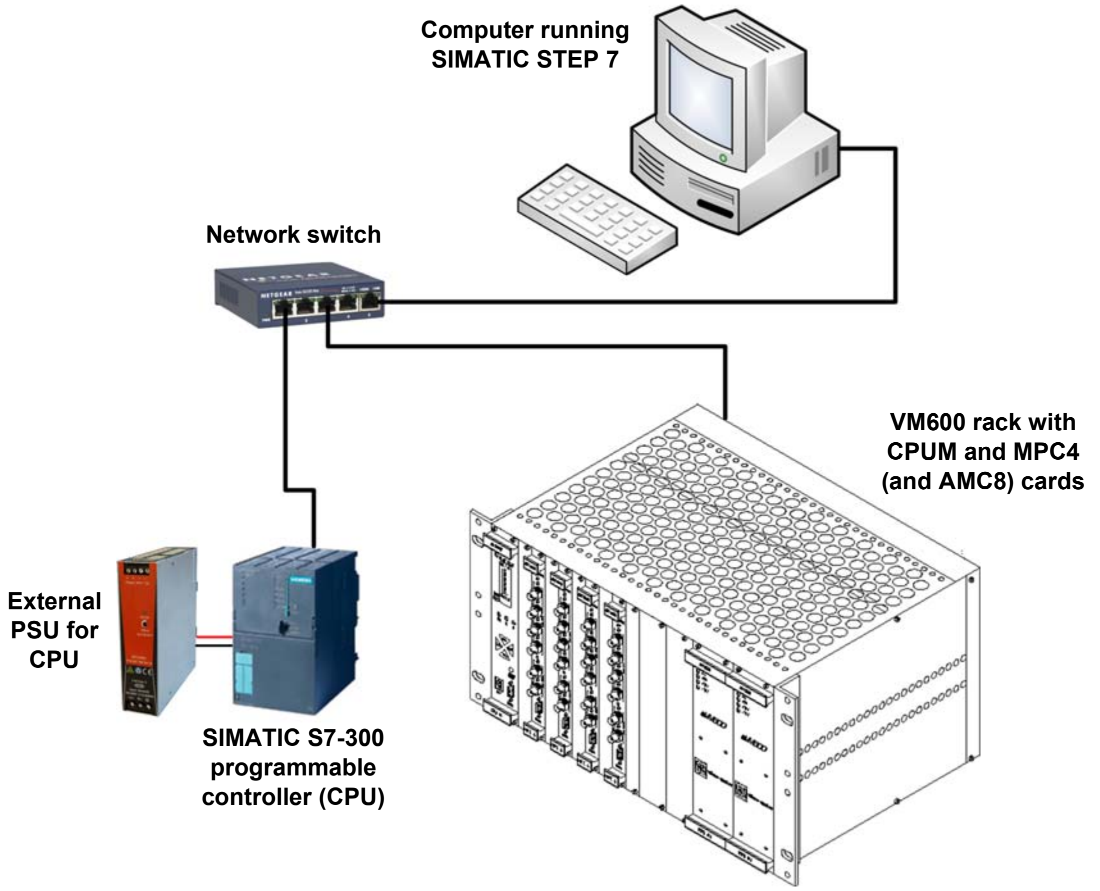


Figure 10-7: Example PROFINET network

10.7.3 Procedure for configuring a PROFINET network with SIMATIC STEP 7

A number of stages are involved in configuring and activating the SIMATIC STEP 7 software before communication with the VM600 rack over PROFINET can be observed. The exact steps required may differ depending on the supplier and version of DCS software being used.

10.7.3.1 Starting the SIMATIC software and creating a new project

1- Click **Start > SIMATIC > SIMATIC Manager** or use a desktop shortcut if available.

NOTE: If the STEP 7 “new project” wizard appears, click Cancel in order to manually configure the project.

The SIMATIC Manager window appears.

2- Click **File > New**.

The New Project dialog box appears.

3- Type a suitable Name for the project (for example, *VM600_PROFINET*), and ensure that the Type is set to Project. Click **OK**.
The new project will be created in SIMATIC Manager

10.7.3.2 Adding objects to the project

The SIMATIC software uses the concept of objects – every item added to a SIMATIC project is considered an object, including the Siemens controller and the VM600 hardware.

1- Right-click on the project (for example, *VM600_PROFINET)* and select **Insert New Object > SIMATIC 300 Station** to add a CPU controller to the project. (Alternatively, the Insert menu can be used to add objects to a project.)
A SIMATIC 300 CPU object is added to the project with a default name of SIMATIC 300(1).

2- Right-click on the project and select **Insert New Object > Industrial Ethernet** to add industrial Ethernet/PROFINET to the project.
An Ethernet object is added to the project with a default name of Ethernet(1).

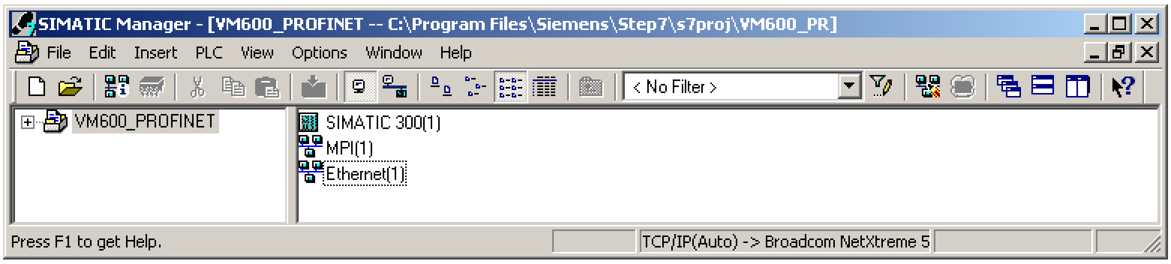

SIMATIC Manager - [VM600_PROFINET -- C:\Program Files\Siemens\Step7\s7proj\VM600_PR]
File Edit Insert PLC View Options Window Help
< No Filter >
VM600_PROFINET
SIMATIC 300(1)
MPI(1)
Ethernet(1)
Press F1 to get Help.
TCP/IP(Auto) -> Broadcom NetXtreme 5

Figure 10-8: SIMATIC Manager – adding objects to a project

3- Right-click on the MPI(1) object in the main window and select **Delete** to remove the MPI network. (This was added by default when the project was created but is not required.)

The project now contains two of the three objects required by this PROFINET system, that is, a PROFINET station (the CPU) and an industrial Ethernet/PROFINET network. The third component, the VM600 hardware, will be added later, after the GSDML files provided by Meggitt SA are registered with the SIMATIC software (see 10.7.3.4 Adding a vibro-meter IO-Device to SIMATIC STEP 7).

10.7.3.3 Configuring the SIMATIC CPU

The objects added to the SIMATIC project, namely the Siemens SIMATIC S7-300 universal controller (CPU), must now be configured.

1- Right-click on the SIMATIC 300(1) object and then click **Open Object**.

The HW Config window appears and displays the current configuration of the SIMATIC 300 PROFINET station.

The HW Config window consists of the Station window (top left), Hardware Catalog window (right) and Online window (bottom left). Note that the Station window is empty, which signifies that the hardware is not configured.

NOTE: To display the HW Config window at any time, right-click on the SIMATIC 300(1) object and click **Open Object**.

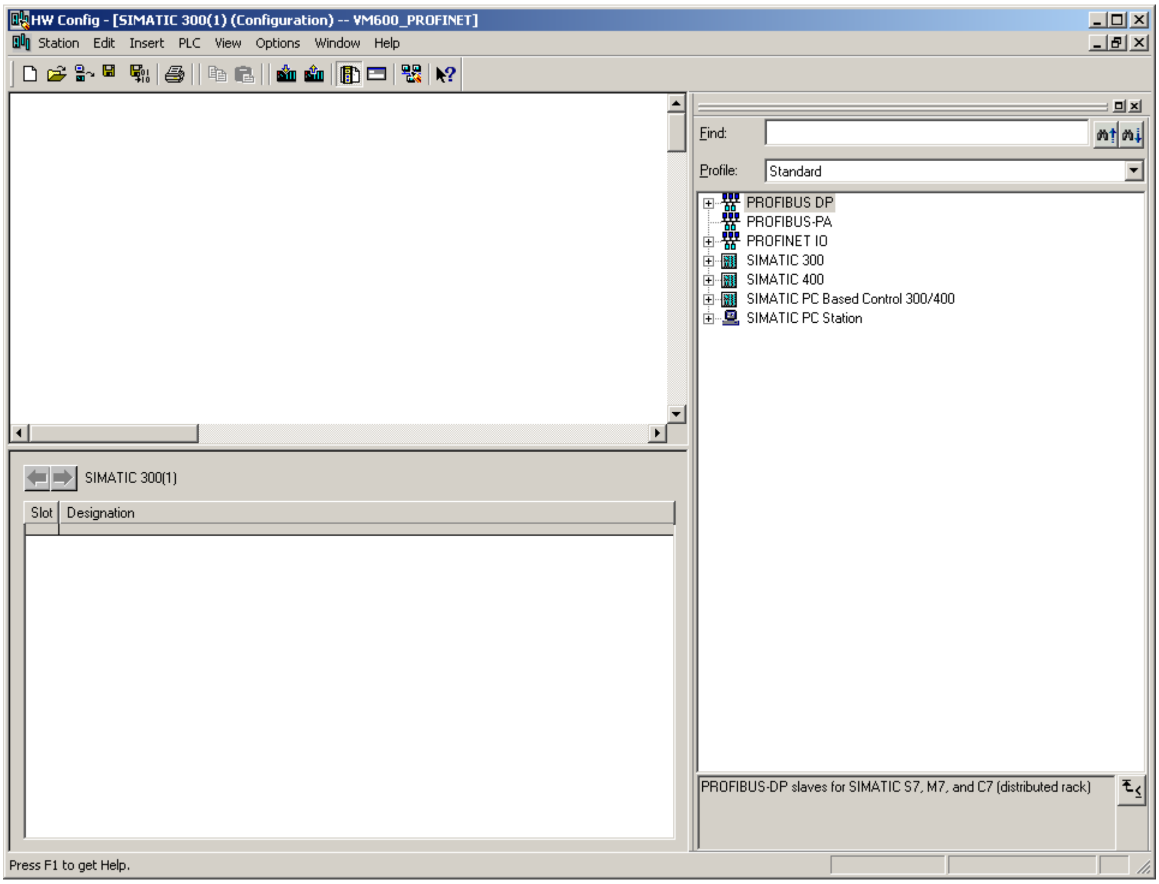
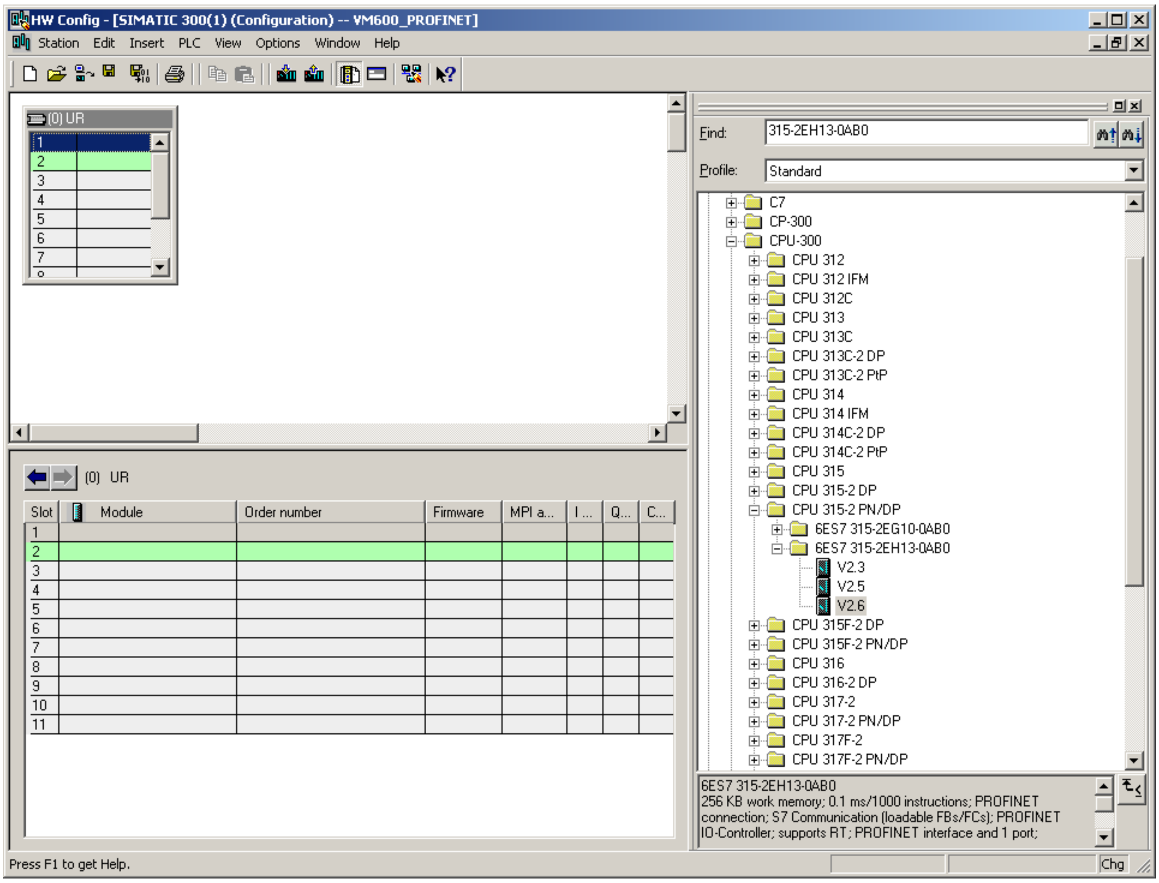

HW Config - [SIMATIC 300(1) (Configuration) -- VM600_PROFINET]
Station Edit Insert PLC View Options Window Help
Find:
Profile: Standard
PROFIBUS DP
PROFIBUS-PA
PROFINET IO
SIMATIC 300
SIMATIC 400
SIMATIC PC Based Control 300/400
SIMATIC PC Station
SIMATIC 300(1)
Slot
Designation
PROFIBUS-DP slaves for SIMATIC S7, M7, and C7 (distributed rack)
Press F1 to get Help.

Figure 10-9: HW Config – with unconfigured objects (the Station window is empty)

2- In the Station Window, right-click and click **Insert Object**. Then click **SIMATIC 300**, **RACK-300** and **Rail** in the dialog boxes that appear.
A Rail object will be added to the project (in the Station window) with a default name of (0)UR.

3- In the Hardware Catalog window, type the version of your CPU (controller) in the Find box, for example, 315-2EH13-0AB0 and press the ENTER key.

The SIMATIC software will search for and then list all the available firmware versions for your CPU.


HW Config - [SIMATIC 300(1) (Configuration) -- VM600_PROFINET]
Station Edit Insert PLC View Options Window Help
(0) UR
Press F1 to get Help.
Chg

(0) UR

Slot	Module	Order number	Firmware	MPI a...	I ...	Q...	C...
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

Find: 315-2EH13-0AB0

Profile: Standard

```
C7
CP-300
CPU-300
  CPU 312
  CPU 312 IFM
  CPU 312C
  CPU 313
  CPU 313C
  CPU 313C-2 DP
  CPU 313C-2 PtP
  CPU 314
  CPU 314 IFM
  CPU 314C-2 DP
  CPU 314C-2 PtP
  CPU 315
  CPU 315-2 DP
  CPU 315-2 PN/DP
    6ES7 315-2EG10-0AB0
    6ES7 315-2EH13-0AB0
      V2.3
      V2.5
      V2.6
  CPU 315F-2 DP
  CPU 315F-2 PN/DP
  CPU 316
  CPU 316-2 DP
  CPU 317-2
  CPU 317-2 PN/DP
  CPU 317F-2
  CPU 317F-2 PN/DP
```

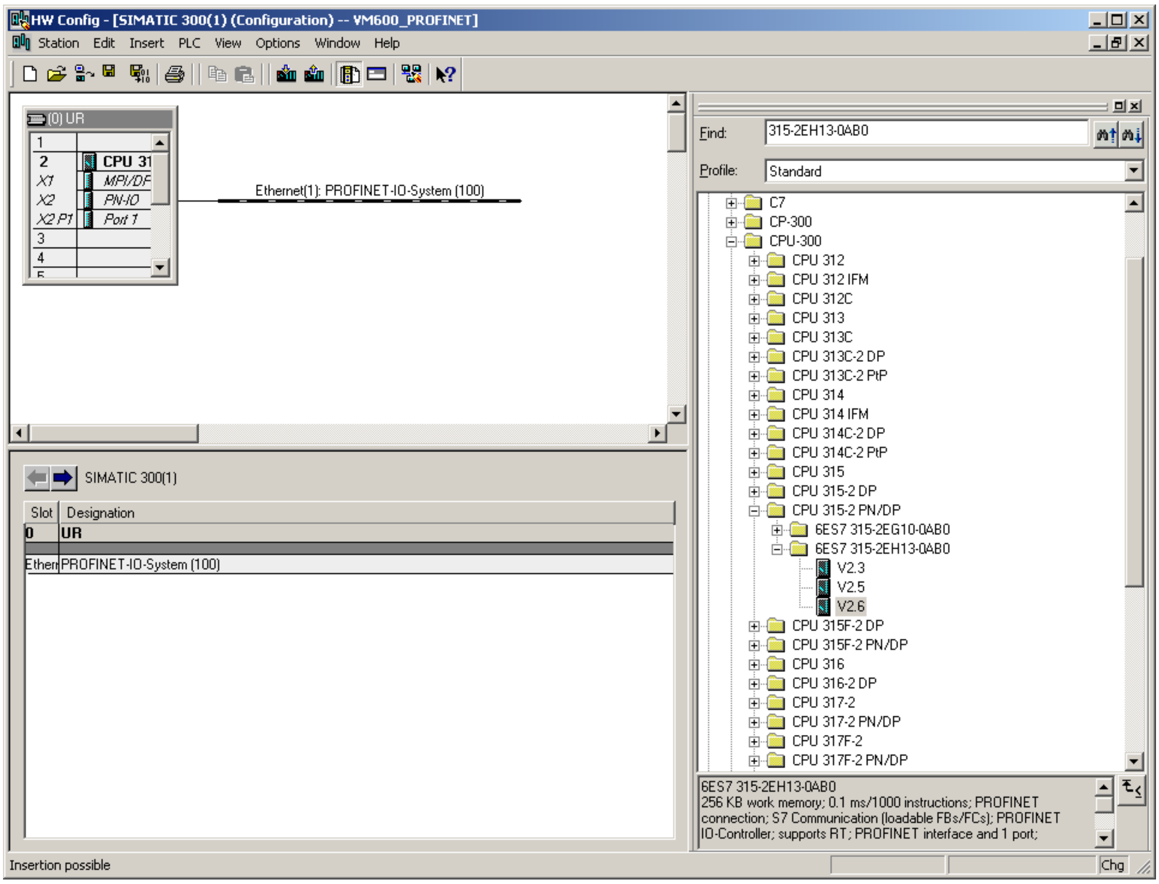
6ES7 315-2EH13-0AB0
256 KB work memory; 0.1 ms/1000 instructions; PROFINET connection; S7 Communication (loadable FBs/FCs); PROFINET IO-Controller; supports RT; PROFINET interface and 1 port;

Figure 10-10: HW Config – the Hardware Catalog window displaying the available CPU firmware

4- To associate the appropriate version of CPU firmware with your project, do one of the following:
Double-click on the appropriate version of the firmware
Or drag the appropriate version of the firmware to the second Slot of the rail.

5- The Properties – Ethernet interface PN-IO dialog box appears.
Enter the networking information appropriate to the Ethernet network in use. As an example, for the private local network defined in Figure 10-7, the following information could be appropriate:
10.10.52.62 for the IP address
255.255.255.0 for the Subnet mask
And Do not use router was selected.
Then select Ethernet(1) in the Subnet area and click **OK**.

A PROFINET IO system is added to the Rail object (in the Station window).



1	
2	CPU 31
X1	MPI/DF
X2	PN-IO
X2 P1	Port 1
3	
4	
5	

SIMATIC 300(1)

Slot	Designation
0	UR
Ether	PROFINET-IO-System (100)

Find: 315-2EH13-0AB0

Profile: Standard

6ES7 315-2EH13-0AB0
256 KB work memory; 0.1 ms/1000 instructions; PROFINET connection; S7 Communication (loadable FBs/FCs); PROFINET IO-Controller; supports RT; PROFINET interface and 1 port;

Figure 10-11: HW Config – the configured objects in the Station window

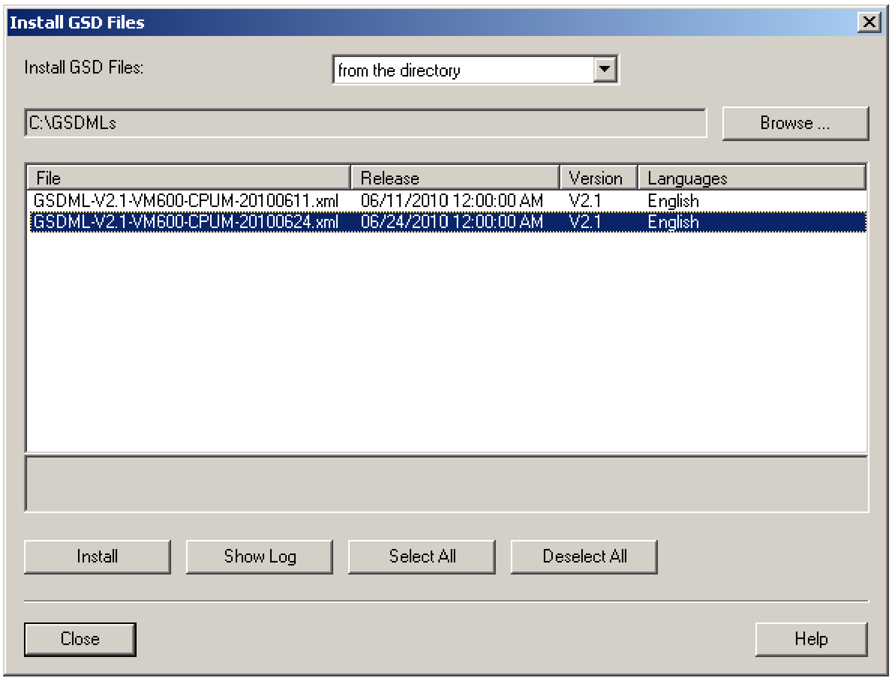
10.7.3.4 Adding a vibro-meter IO-Device to SIMATIC STEP 7

Before a vibro-meter® project can be added to the SIMATIC STEP 7 project, it is necessary to add the IO-Device as an object to the Hardware Catalog. This is done using the GSDML files provided by Meggitt SA (see 10.5 GSDML files).

1- Click **Options > Install GSD File**.

In the Install GSD Files dialog box that appears, ensure that the Install GSD Files option is set to from the directory. Click the Browse button and navigate to the folder where the GSDML files are stored. Click **OK**.

A list of all GSD files available in the folder are presented in the dialog box.



Install GSD Files

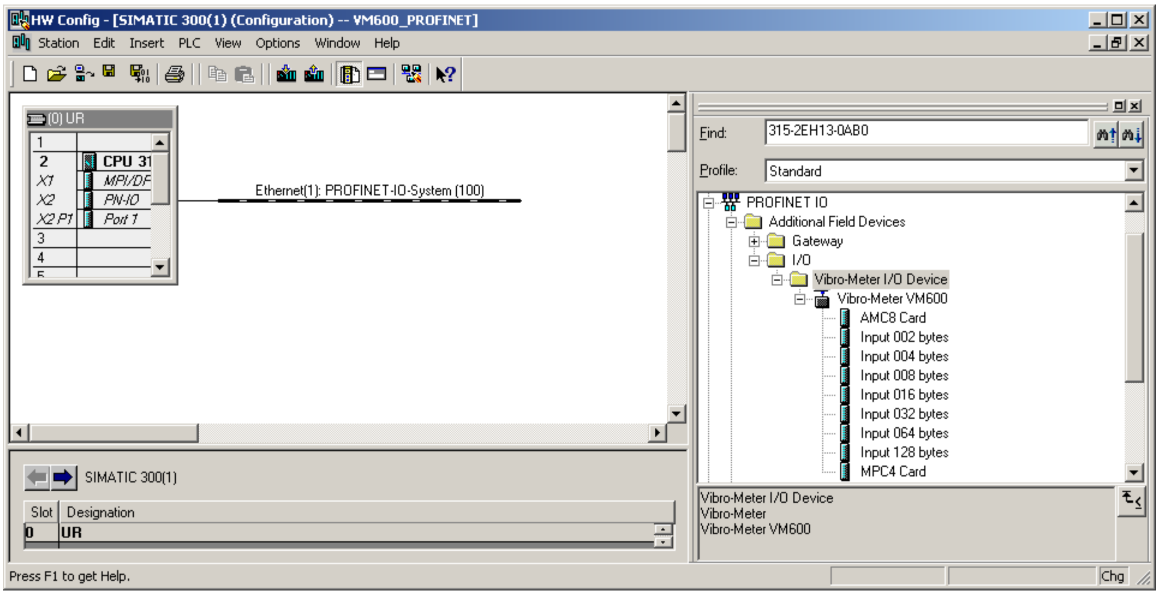
Install GSD Files: from the directory

C:\GSDMLs

File	Release	Version	Languages
GSDML-V2.1-VM600-CPUM-20100611.xml	06/11/2010 12:00:00 AM	V2.1	English
GSDML-V2.1-VM600-CPUM-20100624.xml	06/24/2010 12:00:00 AM	V2.1	English

Figure 10-12: HW Config – installing GSDML files

2- Select the required GSD file from the list and click **Install**. Then click **Yes** to confirm the installation of the GSD.



HW Config - [SIMATIC 300(1) (Configuration) -- VM600_PROFINET]

Slot	Designation
0	UR

Find: 315-2EH13-0AB0

Profile: Standard

- PROFINET IO
  - Additional Field Devices
    - Gateway
    - I/O
      - Vibro-Meter I/O Device
        - Vibro-Meter VM600
          - AMC8 Card
          - Input 002 bytes
          - Input 004 bytes
          - Input 008 bytes
          - Input 016 bytes
          - Input 032 bytes
          - Input 064 bytes
          - Input 128 bytes
          - MPC4 Card

Vibro-Meter I/O Device
Vibro-Meter
Vibro-Meter VM600

Press F1 to get Help.

Figure 10-13: HW Config – with VM600 rack and cards available

The vibro-meter® device is added to the SIMATIC STEP 7 software and can be found in the Hardware Catalog under: PROFINET IO – Additional Field Devices – I/O – Vibro-Meter I/O Device – Vibro-Meter VM600.

Installing a GSDML file (.xml) will add a copy of the GSDML file to the repository of GSD and GSDML files that the SIMATIC software maintains on the computer. The location of this folder will depend on where you have installed the SIMATIC software. For example, `C:\Program Files\Siemens\Step7\S7DATA\GSD`

NOTE: GSDML files cannot be added from this folder. It is a repository maintained by the SIMATIC software.

10.7.3.5 Adding a VM600 rack to the project

As the Vibro-Meter VM600 rack device now exists in the Hardware Catalog (see 10.7.3.4 Adding a vibro-meter IO-Device to SIMATIC STEP 7), it can be added to the PROFINET network (that was started in step 10.7.3.2 Adding objects to the project).

1- Click on the Vibro-Meter VM600 rack icon in the Hardware Catalog and drag it onto the Ethernet(1) object to add a rack to the PROFINET network.

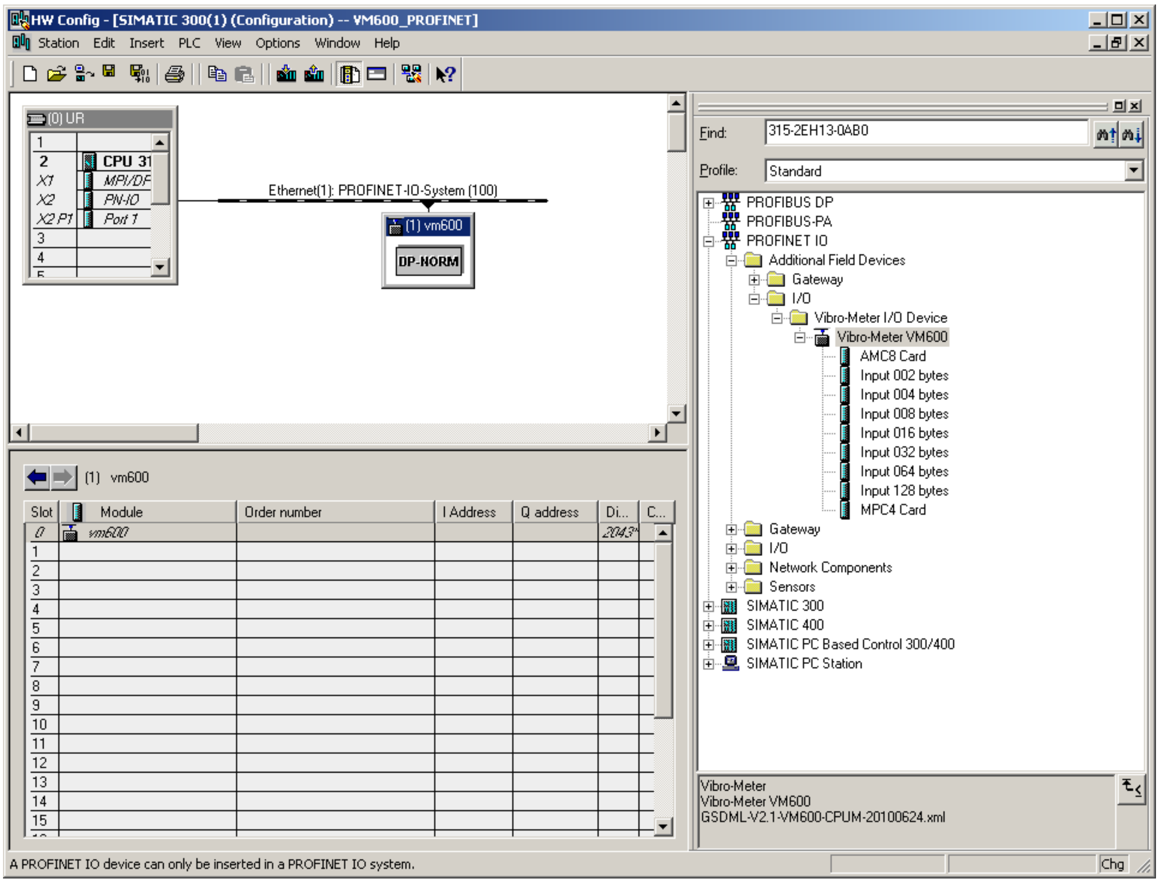


Figure 10-14: HW Config – with a VM600 rack added to the project

The Online window (bottom left) changes to show the slots that are available in the VM600 rack. For a newly added rack, slots 3 to 14 are available to accept cards (although slots 0 to 24 are shown).

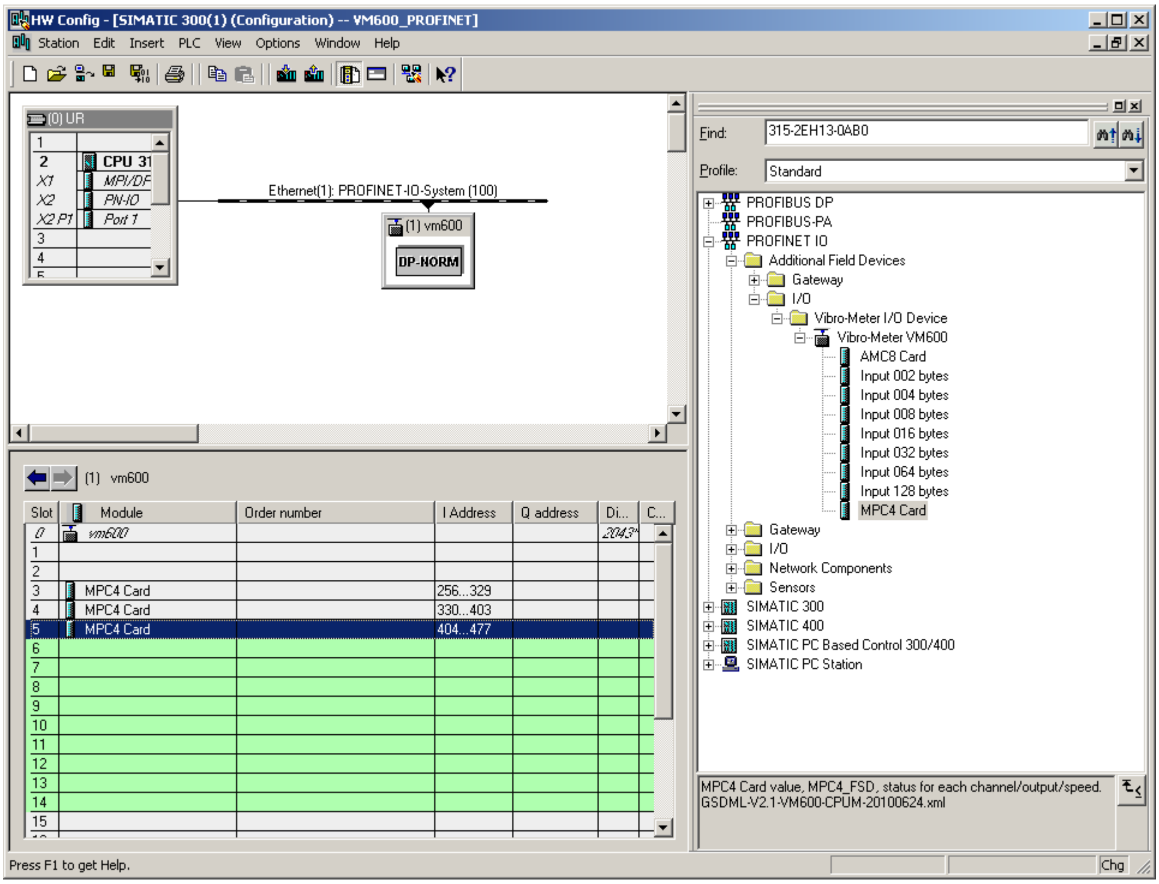
10.7.3.6 Adding a MPC4 or AMC8 card to the project

As the Vibro-Meter VM600 card devices now exist in the Hardware Catalog, they can be added to the PROFINET network (that was started in step 10.7.3.2 - Adding objects to the project).

NOTE: By default, a Vibro-Meter VM600 device does not have any cards installed, as it is not practical to manage GSD files for all possible configurations of VM600 racks and cards. It is therefore necessary to add individual MPC4 card and AMC8 card objects to the VM600 rack device to reflect the actual hardware used.

1- Click on the Vibro-Meter VM600 card icon ▯ – either MPC4 or AMC8 – in the Hardware Catalog and drag it onto the Slot of the Online Window that matches your VM600 rack hardware. (Slots that can accept a card will turn green in colour when a card is selected in the Hardware Catalog.)

This will add the card object to the VM600 rack object in the SIMATIC project.



1	
2	CPU 31
X1	MPI/DF
X2	PN-IO
X2 P1	Port 1
3	
4	

(1) vm600

Slot	Module	Order number	I Address	Q address	Di...	C...
0	vm600				2043*	
1						
2						
3	MPC4 Card		256...329			
4	MPC4 Card		330...403			
5	MPC4 Card		404...477			
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						

Find: 315-2EH13-0AB0

Profile: Standard

MPC4 Card value, MPC4_FSD, status for each channel/output/speed.
GSDML-V2.1-VM600-CPUM-20100624.xml

Press F1 to get Help.

Figure 10-15: HW Config – with VM600 cards added to the project

Repeat this step until the model of the VM600 hardware (rack and cards) created in the SIMATIC project corresponds to the actual VM600 hardware used.

NOTE: It is important that the Slot numbers used in the SIMATIC project match the real hardware as Slot numbers are used to calculate the addresses of variables.

You can right-click on any VM600 card row (Slot) that contains a card and click **Delete** to remove an unwanted card from a slot.

10.7.3.7 Checking the properties of a SIMATIC object

At anytime, the object property window can be used to view (or modify) the parameters of the objects added to a project.

1- To access an object properties window, either:
Double-click on the object
Or right-click on the object and click **Object Properties**.

For example, double-click on the VM600 rack object in the Station window (top left) to open the Properties – vm600 window and inspect the VM600 rack configuration settings.

For example, right-click on the VM600 card row in the Online Window (bottom left) and click **Object Properties** to open the Properties – MPC4 Card (or Properties – AMC8 Card) window and inspect the VM600 card configuration settings.

Although the Device Name given in the VM600 rack object can be modified using the SIMATIC software, it must be the same as the name defined in the `profinet.cfg` configuration file (see 10.6 PROFINET configuration files). This is also the name of the VM600 rack device stored on the CPUM card.

NOTE: Each device used on the network must have a unique name. The device name is restricted to be lower case characters only.

At anytime, the project being worked on in SIMATIC STEP 7 HW Config can be saved in one of the following ways:

Click **Station > Save**.

Or click the Save icon 💾 on the toolbar.

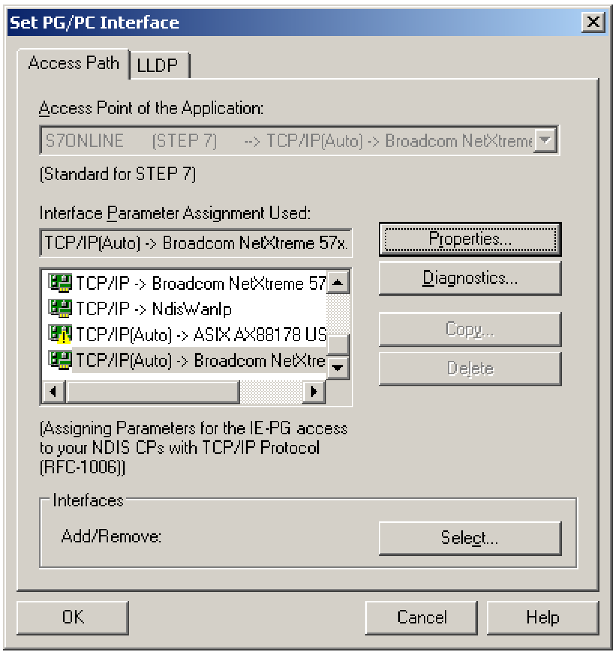
10.7.3.8 Configuring the PG/PC interface

The PG/PC interface is used by the SIMATIC STEP 7 software to program devices connected to the computer, for example, the SIMATIC S7-300 CPU (controller). This interface must be correctly set up before the project can be downloaded to the CPU.

The PG/PC interface is part of the SIMATIC Manager software, not the SIMATIC HW Config software.

1- In SIMATIC Manager, click **Options > Set PG/PC Interface**.

The Set PG/PC Interface window appears.



Set PG/PC Interface

Access Path | LLDP

Access Point of the Application:

S7ONLINE (STEP 7) --> TCP/IP(Auto) -> Broadcom NetXtreme

(Standard for STEP 7)

Interface Parameter Assignment Used:

TCP/IP(Auto) -> Broadcom NetXtreme 57x.

- TCP/IP -> Broadcom NetXtreme 57
- TCP/IP -> NdisWanIp
- TCP/IP(Auto) -> ASIX AX88178 US
- TCP/IP(Auto) -> Broadcom NetXtre

(Assigning Parameters for the IE-PG access to your NDIS CPs with TCP/IP Protocol (RFC-1006))

Interfaces

Add/Remove:

Figure 10-16: SIMATIC Manager – the Set PG/PC Interface window

2- Select the TCP/IP(Auto) row from the Interface Parameter Assignment Used list box that corresponds to the network interface card in your computer that you are using to program the SIMATIC CPU.

Choose and implement the Access Path that best suits your environment. In this example, the TCP/IP (Auto) access path is used.

3- In the Properties – TCP/IP(Auto) window that appears, ensure that Do not assign IP addresses automatically is selected. (Click OK and then OK again to return.)

10.7.3.9 Downloading the configuration to the SIMATIC CPU

The project configuration can now be sent to the SIMATIC CPU.

The download is started from the SIMATIC HW Config software, not the SIMATIC Manager software.

1- In the SIMATIC HW Config, click the Download to Module toolbar button.

2- In the project’s Message Number Assignment Selection dialog box that appears, select Assign CPU-oriented unique message numbers and click **OK**.

3- In the Select Target Module that appears, select the target module that corresponds to your hardware and click **OK**.
For example, CPU 315-2 PN/DP (see 10.7.1 Equipment used).

4- In the Select Node Address dialog box that appears, ensure that the IP address matches the IP address of the CPUM card in the VM600 rack. For example, 10.10.52.20.
(This IP address should be the same as that configured in 10.7.3.3 Configuring the SIMATIC CPU.)

NOTE: To send the configuration to a node other than the default, click **View** to find any other nodes present on the network.

5- The download will start.

The SIMATIC software will prompt you with a dialog box before stopping the module (CPU 315-2 PN/DP/Controller) as part of the download process. Click OK to continue or Cancel to stop the process.

The SIMATIC software will also prompt you with a dialog box before restarting the module as part of the download process. Click Yes to continue or No to stop the process.

10.7.4 Procedures for using a PROFINET network with SIMATIC STEP 7

Now that the PROFINET (industrial Ethernet) network is configured, the SIMATIC STEP 7 software can be used to communicate with the SIMATIC CPU (that is, the CPU 315-2 PN/DP controller) and perform system operations such as:

- Managing the mode of the CPU (such as starting and stopping it)
- Viewing information about the controller.

10.7.4.1 Accessing the controller

These operations are performed using the SIMATIC Manager software.

1- Click **PLC > Diagnostic/Setting > Operating Mode** (or CTRL+I) to view the current operating mode of the CPU.

The Operating Mode window that appears can be used to stop and start the controller, by using the STOP and Warm Restart buttons (if supported by your particular controller).

2- Click **PLC > Diagnostic/Setting > Module Information** (or CTRL+D) to view information about the CPU.

The Module Information window that appears provides much information about the controller. For example, the Events list in the Diagnostic Buffer property sheet keeps a record of the events (with timestamp) that have occurred on this module, such as stopping and starting or module errors.

10.7.4.2 Monitoring the devices

These operations are performed using the SIMATIC HW Config.

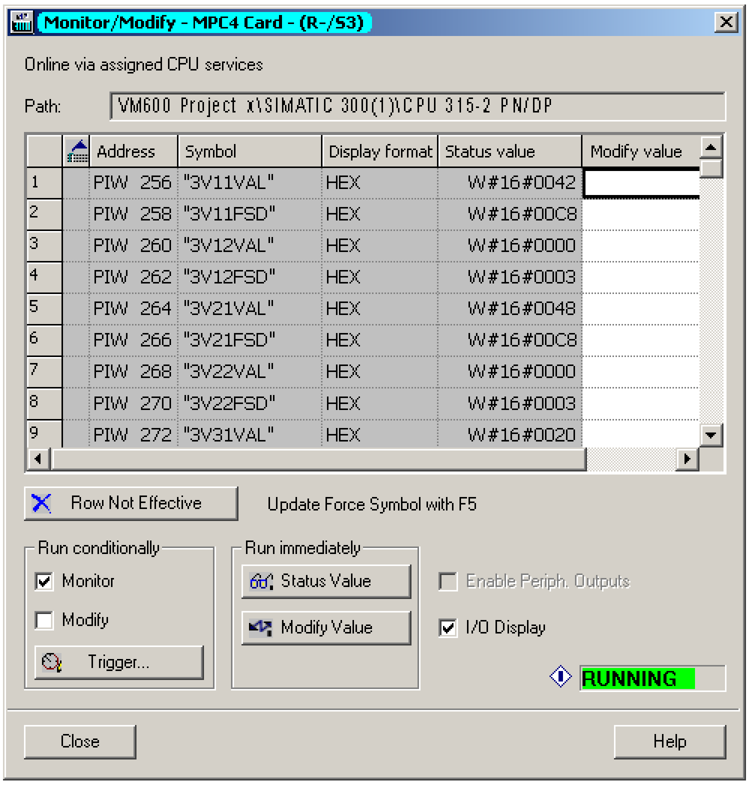
Selecting an object in the Station window (top left) updates the information displayed in the Online Window (bottom left).

1- In the SIMATIC HW Config, click on the (o)UR object in the Station window. The contents of the Online window updates to provide information about the SIMATIC Rail object.

2- In the SIMATIC HW Config, click on the (1)vm600 object in the Station window. The contents of the Online window updates to provide information about the VM600 rack object. Any cards that have been added during the configuration can also be seen.

3- Right-click on a MPC4 (or AMC8) card in a slot in the Online window and click **Monitor/Modify**.

The Monitor/Modify window for the card appears.



Monitor/Modify - MPC4 Card - (R-/S3)

Online via assigned CPU services

Path: VM600 Project x\SIMATIC 300(1)\CPU 315-2 PN/DP

	Address	Symbol	Display format	Status value	Modify value
1	PIW 256	"3V11VAL"	HEX	W#16#0042	
2	PIW 258	"3V11FSD"	HEX	W#16#00C8	
3	PIW 260	"3V12VAL"	HEX	W#16#0000	
4	PIW 262	"3V12FSD"	HEX	W#16#0003	
5	PIW 264	"3V21VAL"	HEX	W#16#0048	
6	PIW 266	"3V21FSD"	HEX	W#16#00C8	
7	PIW 268	"3V22VAL"	HEX	W#16#0000	
8	PIW 270	"3V22FSD"	HEX	W#16#0003	
9	PIW 272	"3V31VAL"	HEX	W#16#0020	

Row Not Effective   Update Force Symbol with F5

Run conditionally: ☑ Monitor, □ Modify, Trigger...

Run immediately: Status Value, Modify Value

□ Enable Periph. Outputs

☑ I/O Display

RUNNING

Close   Help

Figure 10-17: SIMATIC Manager – the Monitor/Modify window

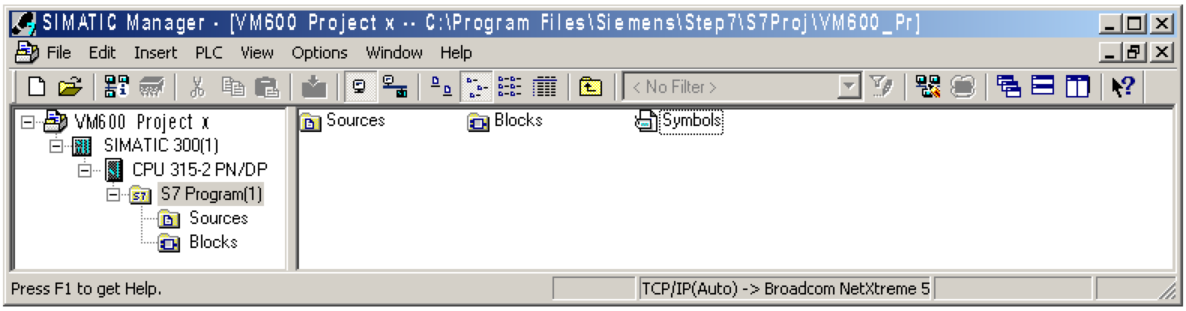
4- Click the **Status Value** button in the Run immediately area of the Monitor/Modify window and the window updates. That is, all status values will update at the same time.

5- Select the **Monitor** check box in the Run conditionally area of the Monitor/Modify window and the window updates continuously.

10.7.4.3 Aliasing device values

This operation is performed using the SIMATIC Symbol Editor, which is accessed using SIMATIC Manager.

1- Double-click on the objects, starting with the project object icon, in the left window of the SIMATIC Manager in order to navigate down to the S7 Program(1) level. (Alternatively, click on the ⊞ icons in the tree structure to navigate down to the S7 Program(1) level.)



SIMATIC Manager - [VM600 Project x -- C:\Program Files\Siemens\Step7\S7Proj\VM600_Pr]

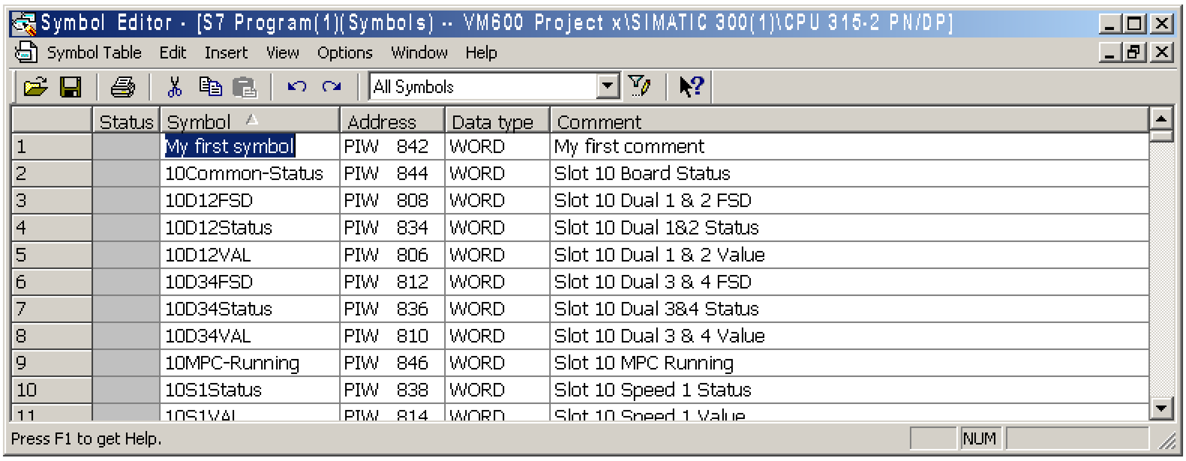
VM600 Project x
SIMATIC 300(1)
CPU 315-2 PN/DP
S7 Program(1)
Sources
Blocks

Sources   Blocks   Symbols

Press F1 to get Help.   TCP/IP(Auto) -> Broadcom NetXtreme 5

Figure 10-18: SIMATIC Manager – accessing the Symbol Editor

2- Double-click on the Symbol object icon in the right (main) window of the SIMATIC Manager.
The Symbol Editor window appears.



Symbol Editor - [S7 Program(1)(Symbols) -- VM600 Project x\SIMATIC 300(1)\CPU 315-2 PN/DP]
Symbol Table Edit Insert View Options Window Help
All Symbols

	Status	Symbol	Address	Data type	Comment
1		My first symbol	PIW 842	WORD	My first comment
2		10Common-Status	PIW 844	WORD	Slot 10 Board Status
3		10D12FSD	PIW 808	WORD	Slot 10 Dual 1 & 2 FSD
4		10D12Status	PIW 834	WORD	Slot 10 Dual 1&2 Status
5		10D12VAL	PIW 806	WORD	Slot 10 Dual 1 & 2 Value
6		10D34FSD	PIW 812	WORD	Slot 10 Dual 3 & 4 FSD
7		10D34Status	PIW 836	WORD	Slot 10 Dual 3&4 Status
8		10D34VAL	PIW 810	WORD	Slot 10 Dual 3 & 4 Value
9		10MPC-Running	PIW 846	WORD	Slot 10 MPC Running
10		10S1Status	PIW 838	WORD	Slot 10 Speed 1 Status
11		10S1VAL	PIW 814	WORD	Slot 10 Speed 1 Value

Figure 10-19: SIMATIC Manager – the Symbol Editor window

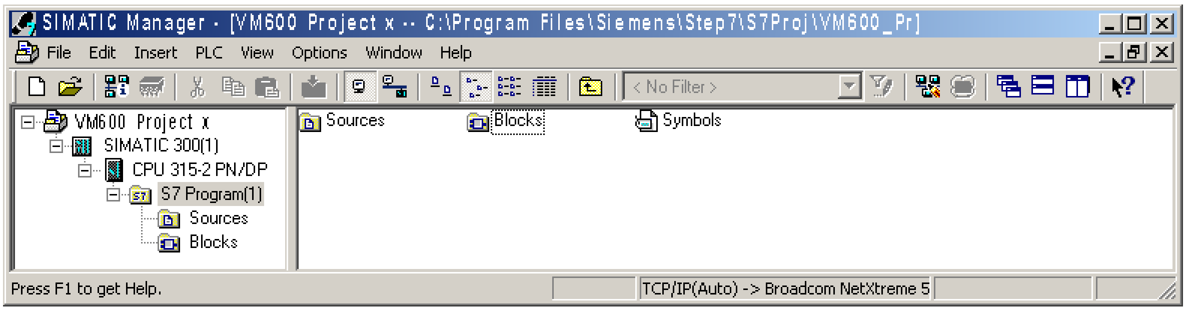
3- Click on the cell corresponding to the Symbol or Comment to edit and type the new information.

NOTE: The table in Appendix E: Symbol names includes a list of predefined symbol names for the MPC4 and AMC8 cards that can be used here.

10.7.4.4 Creating variable tables

This operation is performed using the SIMATIC Var editor, which is accessed using SIMATIC Manager.

1- Double-click on the objects, starting with the project object icon, in the left window of the SIMATIC Manager in order to navigate down to the S7 Program(1) level. (Alternatively, click on the ⊞ icons in the tree structure to navigate down to the S7 Program(1) level.)



SIMATIC Manager - [VM600 Project x -- C:\Program Files\Siemens\Step7\S7Proj\VM600_Pr]
File Edit Insert PLC View Options Window Help
< No Filter >
VM600 Project x
SIMATIC 300(1)
CPU 315-2 PN/DP
S7 Program(1)
Sources
Blocks
Sources
Blocks
Symbols
Press F1 to get Help.
TCP/IP(Auto) -> Broadcom NetXtreme 5

Figure 10-20: SIMATIC Manager – accessing the Variable Editor

2- Right-click on the Blocks object icon in the left window, click **Insert New Object > Variable Table**.
The Properties - Variable Table window appears.

3- Type a name for your variable table in the ‘Symbolic Name’ field and click **OK**.
The variable table is created.

4- Either right-click on the variable table just created and select **Open Object** or double-click on it.
The Var window appears.

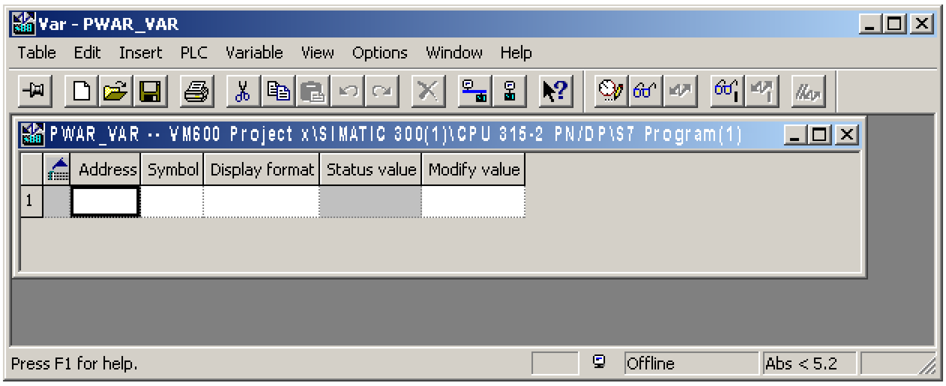

Var - PWAR_VAR
Table Edit Insert PLC Variable View Options Window Help
PWAR_VAR -- VM600 Project x\SIMATIC 300(1)\CPU 315-2 PN/DP\S7 Program(1)
Address Symbol Display format Status value Modify value
1
Press F1 for help.
Offline
Abs < 5.2

Figure 10-21: SIMATIC Manager – the Var window

5- Click on the cell corresponding to the Address or Symbol to edit and type the information for the variable that you want to monitor.

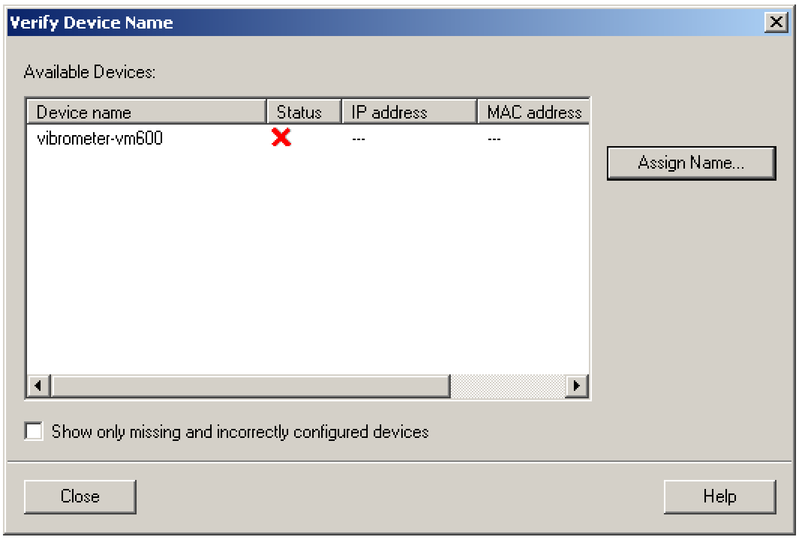
NOTE: Variable tables provide a quick way to easily monitor the most critical or important parameters in a project. (Variable tables can not be used to modify parameters.)

10.7.4.5 Verifying and assigning device names

If the device name configured in HW Config does not correspond to the actual device name, the PROFINET communication will not work. However, it is possible to check the device names that have been assigned and change the name of a device, even if it is already started.

1- In the SIMATIC HW Config, click on the (1)vm600 object to select it.

2- Click **PLC > Ethernet > Verify Device Name**.
The Verify Device Name window appears.



Device name	Status	IP address	MAC address
vibrometer-vm600	✗	---	---

☐ Show only missing and incorrectly configured devices

Figure 10-22: HW Config – the Verify Device Name window

3- To change the name of a device that is already started, click on the device in the list of Available Devices to select it and click the **Assign Name** button. (Alternatively, click **PLC > Ethernet > Assign Device Name**.)
The Assign device name window appears.

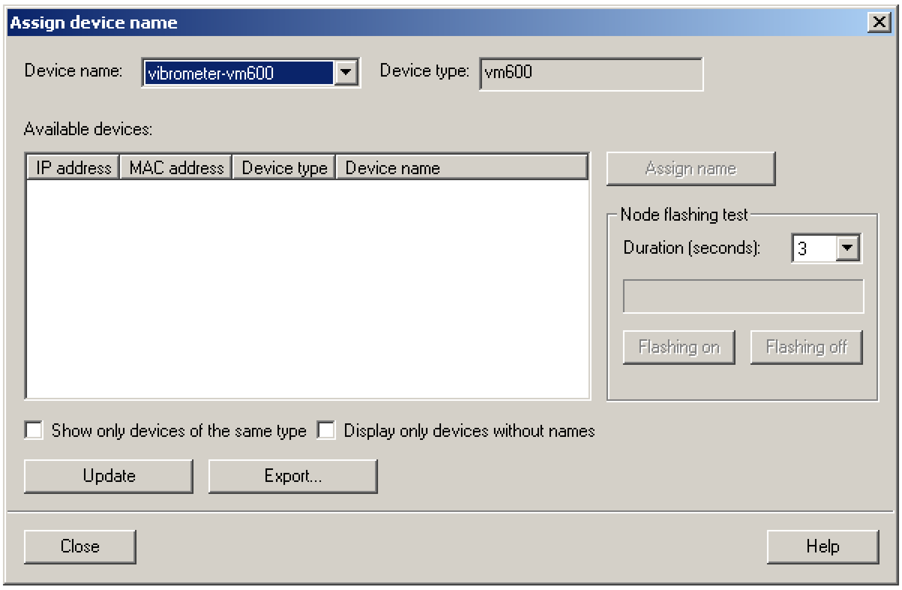


Figure 10-23: HW Config – the Assign device name window

4- Click on the device in the list of Available devices to select it, type the new device name in the Device name column and click the **Update** button.
The device name will be updated.

11 END-OF-LIFE PRODUCT DISPOSAL

A VM600-rack based monitoring system is an electrical/electronic product, therefore, it must be disposed of in a acceptable manner at the end of its useful life. This is important in order to reduce pollution and improve resource efficiency.

NOTE: For environmental and economic reasons, end-of-life electrical and electronic equipment must be collected and treated separately from other waste: it must not go into landfill (or tip, dump, rubbish dump, garbage dump or dumping ground).

In Europe (the European Union), end-of-life electrical/electronic products are classed as waste electrical and electronic equipment (WEEE), and are subject to the requirements of the European Union (EU) directive 2012/19/EU on waste electrical and electronic equipment (commonly referred to as the WEEE directive).

According to the WEEE regulations, all waste electrical and electronic equipment should be collected separately and then treated and disposed of in accordance with the best available and environmentally friendly techniques. This is because electronic waste (or e-waste) may contain substances harmful to the environment and/or to human health. In addition, electronic waste is also a valuable source of raw materials that can contribute to a circular economy.

The WEEE symbol (a “crossed-out wheeled bin”) is used on product labelling to indicate equipment that must be properly treated and disposed of at the end of its life (see Figure 11-1).

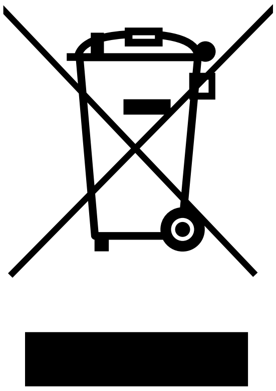


Figure 11-1: WEEE symbol

Although a number of non-EU countries have enacted WEEE regulations, different end-of-life product disposal laws and regulations apply in other countries and regions of the world. Accordingly, please consult your local authorities to obtain the information and guidance relevant to your country and region.

NOTE: At the end of its useful life, a VM600-rack based monitoring system must be disposed of in an environmentally friendly manner.
In European Union Member States, the WEEE directive is applicable.
In other countries and regions of the world, different laws and regulations may be applicable, so please consult your local authorities.

For additional end-of-life product disposal information and guidance, contact your local Meggitt representative. Alternatively, contact our main office:

Environment, health and safety department
Meggitt SA
Route de Moncor 4
Case postale
1701 Fribourg
Switzerland

Telephone: +41 26 407 11 11
Email: ehs@ch.meggitt.com
Website: www.meggittsensing.com/energy

12 SERVICE AND SUPPORT

12.1 Contacting us

Meggitt’s worldwide customer support network offers a range of support, including 12.2 Technical support and 12.3 Sales and repairs support. For customer support, contact your local Meggitt representative. Alternatively, contact our main office:

Customer support department
Meggitt SA
Route de Moncor 4
Case postale
1701 Fribourg
Switzerland

Telephone: +41 26 407 11 11
Email: energysupport@ch.meggitt.com
Website: www.meggittsensing.com/energy

12.2 Technical support

Meggitt’s technical support team provide both pre-sales and post-sales technical support, including:

- General advice
- Technical advice
- Troubleshooting
- Site visits.

NOTE: For further information, contact your local Meggitt representative or Meggitt SA (see 12.1 Contacting us).

12.3 Sales and repairs support

Meggitt's sales team provide both pre-sales and post-sales support, including advice on:

- New products
- Spare parts
- Repairs.

NOTE: If a product has to be returned for repairs, then it should be accompanied by a completed Energy product return form, included on page 12-4.

12.4 Customer feedback

As part of our continuing commitment to improving customer service, we warmly welcome your opinions. To provide feedback, complete the Energy customer feedback form on page 12-7 and return it to Meggitt SA's main office (see 12.1 Contacting us).

REPAIRS AND RETURNS

Energy product return procedure

If a Meggitt vibro-meter® Energy product needs to be returned to Meggitt Switzerland, please use the online product return procedure on the Meggitt vibro-meter® website at:

www.meggittsensing.com/energy/service-and-support/repair

As described on the website, the product return procedure is as follows:

1- Complete and submit online the **Energy product return form** that is available on the website (note: * indicates a required field).

For each Energy product to be returned, a separate Energy product return form must be completed and submitted online.
When an Energy product return form is submitted online, an acknowledgement email including an Energy product return reference number, will be sent by return to confirm that the form was received by Meggitt SA.
Please use the Energy product return reference number in all future communications regarding your product return.

2- Complete and include an end-user certificate.

For each Energy product to be returned, an associated end-user certificate is also required.
The single-use end-user certificate is recommended for smaller organisations that handle few products and the annual end-user certificate is recommended for larger organisations that handle many products.
Either end-user certificate can be used to cover multiple products.

NOTE: Visit the website or contact our Customer support department (see 12.1 Contacting us) to obtain the appropriate end-user certificate form.

3- Send the Energy product together with printed copies of the acknowledgement email (or emails) and the end-user certificate (or certificates) to Meggitt SA at:
Energy Repairs, Meggitt SA, Route de Moncor 4, Case postale, 1701 Fribourg, Switzerland.

A separate acknowledgement email (printed copy) is required for each product to be returned, although a single end-user certificate (printed copy) can be used for multiple products.

4- In addition, a purchase order (PO) with a value of CHF 0.00 must also be sent to Meggitt Switzerland, in order to support the initial problem diagnosis.

NOTE: The **Energy product return form** reproduced below is included to support the gathering of information required for completion and submission online.

Energy product return form

Contact information

First name:*

Last name:*

Job title:

Company:*

Address:*

Country:*

Email:*

Telephone:*

Fax:

Product information

Product type:*

Part number (PNR):*

Serial number (SER):

Note: Enter “Unknown” if the serial number (SER) is not known.

Ex product:

□ Yes

□ No

SIL product:*

□ Yes

□ No

Meggitt SA purchase order number:

Date of purchase (dd.mm.yyyy):

Product under warranty:

□ Yes

□ No

□ Don’t know

Site where installed:

End user:

Return information

Reason for return:*

☐ Repair

☐ Out-of-box failure

If the reason for return is “Repair”, please answer the following questions:*

Type of failure:

☐ Continuous

☐ Intermittent

☐ Temperature dependent

How long was the operating time before failure?

Description of failure:

Please provide a detailed description in order to help with problem diagnosis.

If the reason for return is “Out-of-box failure”, please answer the following questions:*

Type of out-of-box failure:

☐ Product damaged

☐ Incorrect product configuration

☐ Incorrect product delivered

☐ Problem with documentation / labelling

☐ Product dead-on-arrival

Additional information:

Please provide as much information as possible in order to help with problem diagnosis.

Ex product information – additional information required for Ex products only

Is the product installed in a hazardous area (potentially explosive atmosphere)?:

☐ Yes

☐ No

If the product is installed in a hazardous area, please answer the following questions:

How long was the operating time before failure?:

Additional information:

SIL product information – additional information required for SIL products only*

Note: For SIL products used in functional safety contexts/systems, this **SIL product information** section must be completed.

Is the product installed in a safety-related system?:*

☐ Yes

☐ No

If the product is installed in a safety-related system, please answer the following questions:*

Did the system fail** in a safe mode?:* (That is, the safety relay operated but the trip was spurious.)

☐ Yes

☐ No

☐ Not applicable

Did the system fail** in a dangerous state?:* (That is, the failure did not result in the safe state.)

☐ Yes

☐ No

☐ Not applicable

How long was the operating time before failure (in hours)?:*

Additional information:

** A faulty indicator LED is considered as a cosmetic failure.

FEEDBACK

Energy customer feedback form

Manual information

Title of manual:

VM600 networking manual

Reference:	MAVM600-NET/E	Version:	Edition 10
Date of issue:	February 2021		

Customer contact information

First name:*

Last name:*

Job title:

Company:*

Address:*

Country:*

Email:*

Telephone:*

Fax:

Feedback – general

Please answer the following questions:

Is the document well organised?	□ Yes	□ No
Is the information technically accurate?	□ Yes	□ No
Is more technical detail required?	□ Yes	□ No
Are the instructions clear and complete?	□ Yes	□ No
Are the descriptions easy to understand?	□ Yes	□ No
Are the examples and diagrams/photos helpful?	□ Yes	□ No
Are there enough examples and diagrams/photos?	□ Yes	□ No
Is the style/wording easy to read?	□ Yes	□ No
Is any information not included?	□ Yes	□ No

Please include any additional information in the “Feedback – additional” section below.

Feedback – additional

Additional information:

Please provide as much feedback as possible in order to help us improve our product documentation.
Continue on a separate sheet if necessary …

APPENDIX A: MPC4 MODBUS REGISTER DEFINITIONS

MPC4 card register definitions for Modbus can be found in the following address map tables:

- Table A-1: Read coil or discrete input registers (Modbus function codes 01 and 02)
- Table A-2: Read input or holding registers (Modbus function codes 03 and 04).

These tables include a column (CPUM **< 71**) to indicate if the Modbus register is available in a particular version of CPUM firmware. In general, most Modbus registers are available but CPUM cards running firmware version 071 or later have access to all Modbus registers.

These tables include the information required to calculate the Modbus starting addresses (MSAs). These addresses must be known to access the required registers and can be determined using the following technique:

1- Decide in which slot the card will be inserted (Snum).

2- Use the appropriate table (Table A-1 or Table A-2) to determine the Address Offset (Aoff). The choice of table depends on the Modbus function code to be used.

3- Use the formula given in the Modbus starting address (MSA) column of the table to calculate the MSA.

NOTE: To obtain the register number simply add 1 to the calculated MSA value.

Example:

Your MPC4 card is in slot 5 and you want to read the result of the logical combinations of alarms for the 8 basic functions you defined in the MPS software (they correspond to bits b0 to b7).

To do so, you have to use Modbus function 04 - Read Input Registers.

Looking at Table A-2, you can find that the Aoff = 1144.

Finally, using the equation given in the table, you obtain the Modbus starting address:

MSA = ((slot - 3) * 3) + Aoff = (5 - 3) * 3) + 1144 = 1150.

Therefore, the register number = 1150 + 1 = 1151.

NOTE: A complete list of Modbus starting addresses as an Excel® spreadsheet is available from Meggitt SA on request.

Table A-1: MPC4 register definitions
Read coil or discrete input registers (Modbus function codes 01 and 02)

Rack slot number (Snum)	Address offset (Aoff)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value Type	CPUM < 71
From 3 to 14	0	Channel 1, output 1	Value	Point is defined	((slot − 3) * 96) + Aoff	Bit	Same behaviour
	1			Alarm+			
	2			Alarm−			
	3			Danger+			
	4			Danger−			
	5			Sensor OK			
	8	Channel 1, output 2		Point is defined			
	9			Alarm+			
	10			Alarm−			
	11			Danger+			
	12			Danger−			
	13			Sensor OK			
	16	Channel 2, output 1		Point is defined			
	17			Alarm+			
	18			Alarm−			
	19			Danger+			
	20			Danger−			
	21			Sensor OK			
	24	Channel 2, output 2		Point is defined			
	25			Alarm+			
	26			Alarm−			
	27			Danger+			
	28			Danger−			
	29			Sensor OK			
	32	Channel 3, output 1		Point is defined			
	33			Alarm+			
	34			Alarm−			
	35			Danger+			
	36			Danger−			
	37			Sensor OK			
	40	Channel 3, output 2		Point is defined			
	41			Alarm+			
	42			Alarm−			
	43			Danger+			
	44			Danger−			
	45			Sensor OK			

Table A-1: MPC4 register definitions
Read coil or discrete input registers (Modbus function codes 01 and 02)

Rack slot number (Snum)	Address offset (Aoff)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value Type	CPUM < 71
From 3 to 14	48	Channel 4, output 1	Value	Point is defined	((slot – 3) * 96) + Aoff	Bit	Same behaviour
	49			Alarm+			
	50			Alarm−			
	51			Danger+			
	52			Danger−			
	53			Sensor OK			
	56	Channel 4, output 2		Point is defined			
	57			Alarm+			
	58			Alarm−			
	59			Danger+			
	60			Danger−			
	61			Sensor OK			
	64	Dual-channel 1 and 2		Point is defined			
	65			Alarm+			
	66			Alarm−			
	67			Danger+			
	68			Danger−			
	69			Sensor OK			
	72	Dual-channel 3 and 4		Point is defined			
	73			Alarm+			
	74			Alarm−			
	75			Danger+			
	76			Danger−			
	77			Sensor OK			
	80	Speed 1		Point is defined			
	81			Alarm+			
	82			Alarm−			
	85			Sensor OK			
	88	Speed 2		Point is defined			
	89			Alarm+			
	90			Alarm−			
	93			Sensor OK			

Table A-2: MPC4 register definitions
Read holding or input registers (Modbus function codes 03 and 04)

Rack slot number (Snum)	Address offset (Aoff)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value Type	CPUM < 71
From 3 to 14	0	Channel 1, output 1	Value	VAL	((slot – 3) * 22) + Aoff	Unsigned	Same behaviour
	1		Configuration	FSD			
	2	Channel 1, output 2	Value	VAL			
	3		Configuration	FSD			
	4	Channel 2, output 1	Value	VAL			
	5		Configuration	FSD			
	6	Channel 2, output 2	Value	VAL			
	7		Configuration	FSD			
	8	Channel 3, output 1	Value	VAL			
	9		Configuration	FSD			
	10	Channel 3, output 2	Value	VAL			
	11		Configuration	FSD			
	12	Channel 4, output 1	Value	VAL			
	13		Configuration	FSD			
	14	Channel 4, output 2	Value	VAL			
	15		Configuration	FSD			
	16	Dual-channel 1 and 2, output 1	Value	VAL			
	17		Configuration	FSD			
	18	Dual-channel 3 and 4, output 1	Value	VAL			
	19		Configuration	FSD			
	20	Speed 1	Value	VAL			
	21	Speed 2	Value	FSD			
	1000	Channel 1, output 1	Value	Alarm Status (see Table 8-6 and Table 9-10)	((slot – 3) * 12) + Aoff		Only bits b0 to b6 are available
	1001	Channel 1, output 2					
	1002	Channel 2, output 1					
	1003	Channel 2, output 2					
	1004	Channel 3, output 1					
	1005	Channel 3, output 2					
	1006	Channel 4, output 1					
	1007	Channel 4, output 2					
	1008	Dual-channel 1 and 2					
	1009	Dual-channel 3 and 4					
	1010	Speed 1					
	1011	Speed 2					
	1144	Basic and advanced function		Logic Result	((slot – 3) * 3) + Aoff		Not available
	1145	Board status #1		Board Status (see Table 9-12, Table 9-13 and Table 9-14)			
	1146	Board status #2					

APPENDIX B: AMC8 MODBUS REGISTER DEFINITIONS

AMC8 card register definitions for Modbus can be found in the following address map tables:

- Table B-1: Read coil or discrete input registers (Modbus function code 01)
- Table B-2: Read discrete input registers (Modbus function code 02)
- Table B-3: Read input registers (Modbus function code 03)
- Table B-4: Read holding registers (Modbus function code 04).

These tables include a column (CPUM **< 71**) to indicate if the Modbus register is available in a particular version of CPUM firmware. In general, most Modbus registers are available but CPUM cards running software version 071 or later have access to all Modbus registers.

These tables include the information required to calculate the Modbus starting addresses (MSAs). These addresses must be known to access the required registers and can be determined using the following technique:

1- Decide in which slot the card will be inserted (Snum).

2- Use the appropriate table (Table B-1, Table B-2, Table B-3 or Table B-4) to determine the Address Offset (Aoff). The choice of table depends on the Modbus function code to be used.

3- Use the following formula to calculate the Modbus starting address (MSA):

$$\textbf{MSA = 4096 + Aoff + (256 * Snum)} \qquad \text{(B.1)}$$

NOTE: To obtain the register number simply add 1 to the calculated MSA value.

Example:

Your AMC8 card is in slot 14 and you want to read the result of the logical combinations of alarms for the 16 basic functions you defined in the MPS software (they correspond to bits b0 to b15).

To do so, you have to use Modbus function 04 - Read Input Registers.

Looking at Table B-4, you can find that the Aoff = 24.

Finally, using equation (B.1), you obtain the Modbus starting address:

MSA = 4096 + 24 + (256 * 14) = 7704.

Therefore, the register number = 7704 + 1 = 7705.

NOTE: A complete list of Modbus starting addresses as an Excel® spreadsheet is available from Meggitt SA on request.

Table B-1: AMC8 register definitions
Read coil or discrete input registers (Modbus function code 01)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	0	N/A	Channel 1	Configuration	Alert− Low Enable	See equation (B.1)	Bit	Same behaviour
	1		Channel 2					
	2		Channel 3					
	3		Channel 4					
	4		Channel 5					
	5		Channel 6					
	6		Channel 7					
	7		Channel 8					
	8		Multi-channel 1					
	9		Multi-channel 2					
	10		Multi-channel 3					
	11		Multi-channel 4					
	12		Channel 1		Alert+ High Enable			
	13		Channel 2					
	14		Channel 3					
	15		Channel 4					
	16		Channel 5					
	17		Channel 6					
	18		Channel 7					
	19		Channel 8					
	20		Multi-channel 1					
	21		Multi-channel 2					
	22		Multi-channel 3					
	23		Multi-channel 4					
	24		Channel 1		Danger− Low Enable			
	25		Channel 2					
	26		Channel 3					
	27		Channel 4					
	28		Channel 5					
	29		Channel 6					
	30		Channel 7					
	31		Channel 8					
	32		Multi-channel 1					
	33		Multi-channel 2					
	34		Multi-channel 3					
	35		Multi-channel 4					

Table B-1: AMC8 register definitions
Read coil or discrete input registers (Modbus function code 01)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	36	N/A	Channel 1	Configuration	Danger+ High Enable	See equation (B.1)	Bit	Same behaviour
	37		Channel 2					
	38		Channel 3					
	39		Channel 4					
	40		Channel 5					
	41		Channel 6					
	42		Channel 7					
	43		Channel 8					
	44		Multi-channel 1					
	45		Multi-channel 2					
	46		Multi-channel 3					
	47		Multi-channel 4					

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	0	N/A	Channel 1	Value	Alarm+	See equation (B.1)	Bit	Same behaviour
	1		Channel 2					
	2		Channel 3					
	3		Channel 4					
	4		Channel 5					
	5		Channel 6					
	6		Channel 7					
	7		Channel 8					
	8		Multi-channel 1					
	9		Multi-channel 2					
	10		Multi-channel 3					
	11		Multi-channel 4					
	12		Channel 1		Alarm−			
	13		Channel 2					
	14		Channel 3					
	15		Channel 4					
	16		Channel 5					
	17		Channel 6					
	18		Channel 7					
	19		Channel 8					
	20		Multi-channel 1					
	21		Multi-channel 2					
	22		Multi-channel 3					
	23		Multi-channel 4					
	24		Channel 1		Danger+			
	25		Channel 2					
	26		Channel 3					
	27		Channel 4					
	28		Channel 5					
	29		Channel 6					
	30		Channel 7					
	31		Channel 8					
	32		Multi-channel 1					
	33		Multi-channel 2					
	34		Multi-channel 3					
	35		Multi-channel 4					

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	36	N/A	Channel 1	Value	Danger−	See equation (B.1)	Bit	Same behaviour
	37		Channel 2					
	38		Channel 3					
	39		Channel 4					
	40		Channel 5					
	41		Channel 6					
	42		Channel 7					
	43		Channel 8					
	44		Multi-channel 1					
	45		Multi-channel 2					
	46		Multi-channel 3					
	47		Multi-channel 4					
	48		Channel 1		Global Channel OK Fail			
	49		Channel 2					
	50		Channel 3					
	51		Channel 4					
	52		Channel 5					
	53		Channel 6					
	54		Channel 7					
	55		Channel 8					
	56		Multi-channel 1					
	57		Multi-channel 2					
	58		Multi-channel 3					
	59		Multi-channel 4					
	60		Channel 1		ADC Error			
	61		Channel 2					
	62		Channel 3					
	63		Channel 4					
	64		Channel 5					
	65		Channel 6					
	66		Channel 7					
	67		Channel 8					
	68		Multi-channel 1					
	69		Multi-channel 2					
	70		Multi-channel 3					
	71		Multi-channel 4					
	72		Channel 1		ADC Standby			
	73		Channel 2					
	74		Channel 3					

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	75	N/A	Channel 4	Value	ADC Standby	See equation (B.1)	Bit	Same behaviour
	76		Channel 5					
	77		Channel 6					
	78		Channel 7					
	79		Channel 8					
	80		Multi-channel 1					
	81		Multi-channel 2					
	82		Multi-channel 3					
	83		Multi-channel 4					
	84		Channel 1		ADC PLL Lock Error			
	85		Channel 2					
	86		Channel 3					
	87		Channel 4					
	88		Channel 5					
	89		Channel 6					
	90		Channel 7					
	91		Channel 8					
	92		Multi-channel 1					
	93		Multi-channel 2					
	94		Multi-channel 3					
	95		Multi-channel 4					
	96		Channel 1		ADC Transmission Error			
	97		Channel 2					
	98		Channel 3					
	99		Channel 4					
	100		Channel 5					
	101		Channel 6					
	102		Channel 7					
	103		Channel 8					
	104		Multi-channel 1					
	105		Multi-channel 2					
	106		Multi-channel 3					
	107		Multi-channel 4					
	108		Channel 1		Dynamic Configuration Error			
	109		Channel 2					
	110		Channel 3					
	111		Channel 4					
	112		Channel 5					
	113		Channel 6					

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	114	N/A	Channel 7	Value	Dynamic Configuration Error	See equation (B.1)	Bit	Same behaviour
	115		Channel 8					
	116		Multi-channel 1					
	117		Multi-channel 2					
	118		Multi-channel 3					
	119		Multi-channel 4					
	120		Channel 1		BIT Result Fail			
	121		Channel 2					
	122		Channel 3					
	123		Channel 4					
	124		Channel 5					
	125		Channel 6					
	126		Channel 7					
	127		Channel 8					
	128		Multi-channel 1					
	129		Multi-channel 2					
	130		Multi-channel 3					
	131		Multi-channel 4					
	132		Channel 1		No Sample			
	133		Channel 2					
	134		Channel 3					
	135		Channel 4					
	136		Channel 5					
	137		Channel 6					
	138		Channel 7					
	139		Channel 8					
	140		Multi-channel 1					
	141		Multi-channel 2					
	142		Multi-channel 3					
	143		Multi-channel 4					
	144		Channel 1		OK Error Fail			
	145		Channel 2					
	146		Channel 3					
	147		Channel 4					
	148		Channel 5					
	149		Channel 6					
	150		Channel 7					
	151		Channel 8					
	152		Multi-channel 1					

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	153	N/A	Multi-channel 2	Value	OK Error Fail	See equation (B.1)	Bit	Same behaviour
	154		Multi-channel 3					
	155		Multi-channel 4					
	156		Channel 1		Linearization Error			
	157		Channel 2					
	158		Channel 3					
	159		Channel 4					
	160		Channel 5					
	161		Channel 6					
	162		Channel 7					
	163		Channel 8					
	164		Multi-channel 1					
	165		Multi-channel 2					
	166		Multi-channel 3					
	167		Multi-channel 4					
	168		Channel 1		Cold Junction Error			
	169		Channel 2					
	170		Channel 3					
	171		Channel 4					
	172		Channel 5					
	173		Channel 6					
	174		Channel 7					
	175		Channel 8					
	176		Multi-channel 1					
	177		Multi-channel 2					
	178		Multi-channel 3					
	179		Multi-channel 4					
	180		N/A		BasicFunction1			
	181				BasicFunction2			
	182				BasicFunction3			
	183				BasicFunction4			
	184				BasicFunction5			
	185				BasicFunction6			
	186				BasicFunction7			
	187				BasicFunction8			
	188				BasicFunction9			
	189				BasicFunction10			
	190				BasicFunction11			
	191				BasicFunction12			

Table B-2: AMC8 register definitions
Read discrete input registers (Modbus function code 02)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	192	N/A	N/A	Value	BasicFunction13	See equation (B.1)	Bit	Same behaviour
	193				BasicFunction14			
	194				BasicFunction15			
	195				BasicFunction16			
	196				AdvancedFunction1			
	197				AdvancedFunction2			
	198				AdvancedFunction3			
	199				AdvancedFunction4			
	200				AdvancedFunction5			
	201				AdvancedFunction6			
	202				AdvancedFunction7			
	203				AdvancedFunction8			

Table B-3: AMC8 register definitions
Read input registers (Modbus function code 03)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	0	0-15	Channel 1	Value	Processed Values (see Table 8-7)	See equation (B.1)	Float32	Same behaviour
	1	16-31						
	2	0-15	Channel 2					
	3	16-31						
	4	0-15	Channel 3					
	5	16-31						
	6	0-15	Channel 4					
	7	16-31						
	8	0-15	Channel 5					
	9	16-31						
	10	0-15	Channel 6					
	11	16-31						
	12	0-15	Channel 7					
	13	16-31						
	14	0-15	Channel 8					
	15	16-31						
	16	0-15	Multi-channel 1					
	17	16-31						
	18	0-15	Multi-channel 2					
	19	16-31						
	20	0-15	Multi-channel 3					
	21	16-31						
	22	0-15	Multi-channel 4					
	23	16-31						
	24	0-15	N/A		Logical Results (see Table 8-7)		U32	
	25	16-31						
	26	0-15	Channel 1		Alarm Status (see Table 8-7)		U16	
	27		Channel 2					
	28		Channel 3					
	29		Channel 4					
	30		Channel 5					
	31		Channel 6					
	32		Channel 7					

Table B-3: AMC8 register definitions
Read input registers (Modbus function code 03)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	33	0-15	Channel 8	Value	Alarm Status (see Table 8-7)	See equation (B.1)	U16	Same behaviour
	34		Multi-channel 1					
	35		Multi-channel 2					
	36		Multi-channel 3					
	37		Multi-channel 4					
From 3 to 14	38	0-15	Channel 1	Value	Alarm used (see Table 9-15)		U16	Not available
	39		Channel 2					
	40		Channel 3					
	41		Channel 4					
	42		Channel 5					
	43		Channel 6					
	44		Channel 7					
	45		Channel 8					
	46		Multi-channel 1					
	47		Multi-channel 2					
	48		Multi-channel 3					
	49		Multi-channel 4					
	50		N/A		Control status (Table 9-16)			

Table B-4: AMC8 register definitions
Read holding registers (Modbus function code 04)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	0	0-15	Channel 1	Configuration	Minimum Displayed Value	See equation (B.1)	Float32	Same behaviour
	1	16-31						
	2	0-15	Channel 2					
	3	16-31						
	4	0-15	Channel 3					
	5	16-31						
	6	0-15	Channel 4					
	7	16-31						
	8	0-15	Channel 5					
	9	16-31						
	10	0-15	Channel 6					
	11	16-31						
	12	0-15	Channel 7					
	13	16-31						
	14	0-15	Channel 8					
	15	16-31						
	16	0-15	Multi-channel 1					
	17	16-31						
	18	0-15	Multi-channel 2					
	19	16-31						
	20	0-15	Multi-channel 3					
	21	16-31						
	22	0-15	Multi-channel 4					
	23	16-31						
	24	0-15	Channel 1		Maximum Displayed Value			
	25	16-31						
	26	0-15	Channel 2					
	27	16-31						
	28	0-15	Channel 3					
	29	16-31						
	30	0-15	Channel 4					
	31	16-31						
	32	0-15	Channel 5					
	33	16-31						
	34	0-15	Channel 6					
	35	16-31						

Table B-4: AMC8 register definitions
Read holding registers (Modbus function code 04)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	36	0-15	Channel 7	Configuration	Maximum Displayed Value	See equation (B.1)	Float32	Same behaviour
	37	16-31						
	38	0-15	Channel 8					
	39	16-31						
	40	0-15	Multi-channel 1					
	41	16-31						
	42	0-15	Multi-channel 2					
	43	16-31						
	44	0-15	Multi-channel 3					
	45	16-31						
	46	0-15	Multi-channel 4					
	47	16-31						
	48	0-15	Channel 1		Alert− Low			
	49	16-31						
	50	0-15	Channel 2					
	51	16-31						
	52	0-15	Channel 3					
	53	16-31						
	54	0-15	Channel 4					
	55	16-31						
	56	0-15	Channel 5					
	57	16-31						
	58	0-15	Channel 6					
	59	16-31						
	60	0-15	Channel 7					
	61	16-31						
	62	0-15	Channel 8					
	63	16-31						
	64	0-15	Multi-channel 1					
	65	16-31						
	66	0-15	Multi-channel 2					
	67	16-31						
	68	0-15	Multi-channel 3					
	69	16-31						
	70	0-15	Multi-channel 4					
	71	16-31						
	72	0-15	Channel 1		Alert+ High			
	73	16-31						

Table B-4: AMC8 register definitions
Read holding registers (Modbus function code 04)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	74	0-15	Channel 2	Configuration	Alert+ High	See equation (B.1)	Float32	Same behaviour
	75	16-31						
	76	0-15	Channel 3					
	77	16-31						
	78	0-15	Channel 4					
	79	16-31						
	80	0-15	Channel 5					
	81	16-31						
	82	0-15	Channel 6					
	83	16-31						
	84	0-15	Channel 7					
	85	16-31						
	86	0-15	Channel 8					
	87	16-31						
	88	0-15	Multi-channel 1					
	89	16-31						
	90	0-15	Multi-channel 2					
	91	16-31						
	92	0-15	Multi-channel 3					
	93	16-31						
	94	0-15	Multi-channel 4					
	95	16-31						
	96	0-15	Channel 1		Danger− Low			
	97	16-31						
	98	0-15	Channel 2					
	99	16-31						
	100	0-15	Channel 3					
	101	16-31						
	102	0-15	Channel 4					
	103	16-31						
	104	0-15	Channel 5					
	105	16-31						
	106	0-15	Channel 6					
	107	16-31						
	108	0-15	Channel 7					
	109	16-31						
	110	0-15	Channel 8					
	111	16-31						

Table B-4: AMC8 register definitions
Read holding registers (Modbus function code 04)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	112	0-15	Multi-channel 1	Configuration	Danger− Low	See equation (B.1)	Float32	Same behaviour
	113	16-31						
	114	0-15	Multi-channel 2					
	115	16-31						
	116	0-15	Multi-channel 3					
	117	16-31						
	118	0-15	Multi-channel 4					
	119	16-31						
	120	0-15	Channel 1		Danger+ High			
	121	16-31						
	122	0-15	Channel 2					
	123	16-31						
	124	0-15	Channel 3					
	125	16-31						
	126	0-15	Channel 4					
	127	16-31						
	128	0-15	Channel 5					
	129	16-31						
	130	0-15	Channel 6					
	131	16-31						
	132	0-15	Channel 7					
	133	16-31						
	134	0-15	Channel 8					
	135	16-31						
	136	0-15	Multi-channel 1					
	137	16-31						
	138	0-15	Multi-channel 2					
	139	16-31						
	140	0-15	Multi-channel 3					
	141	16-31						
	142	0-15	Multi-channel 4					
	143	16-31						

Table B-4: AMC8 register definitions
Read holding registers (Modbus function code 04)

Rack slot number (Snum)	Address offset (Aoff)	Bit(s)	Channel	Internal value type	Value description	Modbus starting address (MSA)	Value type	CPUM < 71
From 3 to 14	144	0-15	Channel 1	Configuration	Output Unit	See equation (B.1)	U16	Same behaviour
	145	0-15	Channel 2					
	146	0-15	Channel 3					
	147	0-15	Channel 4					
	148	0-15	Channel 5					
	149	0-15	Channel 6					
	150	0-15	Channel 7					
	151	0-15	Channel 8					
	152	0-15	Multi-channel 1					
	153	0-15	Multi-channel 2					
	154	0-15	Multi-channel 3					
	155	0-15	Multi-channel 4					

APPENDIX C: MPC4 PROFINET SLOT DEFINITIONS

MPC4 card slot definitions for PROFINET are illustrated using the following extract from a typical `modbusDefault.cfg` configuration file for an MPC4 card in slot 3 of a VM600 rack.

```
...

[GLOBAL]
...
IS_PROFINET_ACTIVED = YES

[MAPPING]

/////////////////////////////////////////////////
///////////////// Slot3 ////////////////////////
/////////////////////////////////////////////////

///////////////// Alarm and Status //////////////

//  Slot3, Channel 1, Output 1
$MPC4S3C1O1Used = Slot3:MPC4:Config:C1:O1:Used
$MPC4S3C1O1Apos = Slot3:MPC4:C1:O1:A+
$MPC4S3C1O1Aneg = Slot3:MPC4:C1:O1:A-
$MPC4S3C1O1Dpos = Slot3:MPC4:C1:O1:D+
$MPC4S3C1O1Dneg = Slot3:MPC4:C1:O1:D-
$MPC4S3C1O1SOK  = Slot3:MPC4:Status:C1:SOK
$MPC4S3C1O1UsedAndApos = AND($MPC4S3C1O1Used, $MPC4S3C1O1Apos)
$MPC4S3C1O1UsedAndAneg = AND($MPC4S3C1O1Used, $MPC4S3C1O1Aneg)
$MPC4S3C1O1UsedAndDpos = AND($MPC4S3C1O1Used, $MPC4S3C1O1Dpos)
$MPC4S3C1O1UsedAndDneg = AND($MPC4S3C1O1Used, $MPC4S3C1O1Dneg)
$MPC4S3C1O1UsedAndSOK  = AND($MPC4S3C1O1Used, $MPC4S3C1O1SOK)

//  Slot3, Channel 1, Output 2
$MPC4S3C1O2Used = Slot3:MPC4:Config:C1:O2:Used
$MPC4S3C1O2Apos = Slot3:MPC4:C1:O2:A+
$MPC4S3C1O2Aneg = Slot3:MPC4:C1:O2:A-
$MPC4S3C1O2Dpos = Slot3:MPC4:C1:O2:D+
$MPC4S3C1O2Dneg = Slot3:MPC4:C1:O2:D-
$MPC4S3C1O2SOK  = Slot3:MPC4:Status:C1:SOK
$MPC4S3C1O2UsedAndApos = AND($MPC4S3C1O2Used, $MPC4S3C1O2Apos)
$MPC4S3C1O2UsedAndAneg = AND($MPC4S3C1O2Used, $MPC4S3C1O2Aneg)
$MPC4S3C1O2UsedAndDpos = AND($MPC4S3C1O2Used, $MPC4S3C1O2Dpos)
$MPC4S3C1O2UsedAndDneg = AND($MPC4S3C1O2Used, $MPC4S3C1O2Dneg)
$MPC4S3C1O2UsedAndSOK  = AND($MPC4S3C1O2Used, $MPC4S3C1O2SOK)
```

```
//  Slot3, Channel 2, Output 1
$MPC4S3C2O1Used = Slot3:MPC4:Config:C2:O1:Used
$MPC4S3C2O1Apos = Slot3:MPC4:C2:O1:A+
$MPC4S3C2O1Aneg = Slot3:MPC4:C2:O1:A-
$MPC4S3C2O1Dpos = Slot3:MPC4:C2:O1:D+
$MPC4S3C2O1Dneg = Slot3:MPC4:C2:O1:D-
$MPC4S3C2O1SOK  = Slot3:MPC4:Status:C2:SOK
$MPC4S3C2O1UsedAndApos = AND($MPC4S3C2O1Used, $MPC4S3C2O1Apos)
$MPC4S3C2O1UsedAndAneg = AND($MPC4S3C2O1Used, $MPC4S3C2O1Aneg)
$MPC4S3C2O1UsedAndDpos = AND($MPC4S3C2O1Used, $MPC4S3C2O1Dpos)
$MPC4S3C2O1UsedAndDneg = AND($MPC4S3C2O1Used, $MPC4S3C2O1Dneg)
$MPC4S3C2O1UsedAndSOK  = AND($MPC4S3C2O1Used, $MPC4S3C2O1SOK)

//  Slot3, Channel 2, Output 2
$MPC4S3C2O2Used = Slot3:MPC4:Config:C2:O2:Used
$MPC4S3C2O2Apos = Slot3:MPC4:C2:O2:A+
$MPC4S3C2O2Aneg = Slot3:MPC4:C2:O2:A-
$MPC4S3C2O2Dpos = Slot3:MPC4:C2:O2:D+
$MPC4S3C2O2Dneg = Slot3:MPC4:C2:O2:D-
$MPC4S3C2O2SOK  = Slot3:MPC4:Status:C2:SOK
$MPC4S3C2O2UsedAndApos = AND($MPC4S3C2O2Used, $MPC4S3C2O2Apos)
$MPC4S3C2O2UsedAndAneg = AND($MPC4S3C2O2Used, $MPC4S3C2O2Aneg)
$MPC4S3C2O2UsedAndDpos = AND($MPC4S3C2O2Used, $MPC4S3C2O2Dpos)
$MPC4S3C2O2UsedAndDneg = AND($MPC4S3C2O2Used, $MPC4S3C2O2Dneg)
$MPC4S3C2O2UsedAndSOK  = AND($MPC4S3C2O2Used, $MPC4S3C2O2SOK)

//  Slot3, Channel 3, Output 1
$MPC4S3C3O1Used = Slot3:MPC4:Config:C3:O1:Used
$MPC4S3C3O1Apos = Slot3:MPC4:C3:O1:A+
$MPC4S3C3O1Aneg = Slot3:MPC4:C3:O1:A-
$MPC4S3C3O1Dpos = Slot3:MPC4:C3:O1:D+
$MPC4S3C3O1Dneg = Slot3:MPC4:C3:O1:D-
$MPC4S3C3O1SOK  = Slot3:MPC4:Status:C3:SOK
$MPC4S3C3O1UsedAndApos = AND($MPC4S3C3O1Used, $MPC4S3C3O1Apos)
$MPC4S3C3O1UsedAndAneg = AND($MPC4S3C3O1Used, $MPC4S3C3O1Aneg)
$MPC4S3C3O1UsedAndDpos = AND($MPC4S3C3O1Used, $MPC4S3C3O1Dpos)
$MPC4S3C3O1UsedAndDneg = AND($MPC4S3C3O1Used, $MPC4S3C3O1Dneg)
$MPC4S3C3O1UsedAndSOK  = AND($MPC4S3C3O1Used, $MPC4S3C3O1SOK)

//  Slot3, Channel 3, Output 2
$MPC4S3C3O2Used = Slot3:MPC4:Config:C3:O2:Used
$MPC4S3C3O2Apos = Slot3:MPC4:C3:O2:A+
$MPC4S3C3O2Aneg = Slot3:MPC4:C3:O2:A-
$MPC4S3C3O2Dpos = Slot3:MPC4:C3:O2:D+
$MPC4S3C3O2Dneg = Slot3:MPC4:C3:O2:D-
```

```
$MPC4S3C3O2SOK  = Slot3:MPC4:Status:C3:SOK
$MPC4S3C3O2UsedAndApos = AND($MPC4S3C3O2Used, $MPC4S3C3O2Apos)
$MPC4S3C3O2UsedAndAneg = AND($MPC4S3C3O2Used, $MPC4S3C3O2Aneg)
$MPC4S3C3O2UsedAndDpos = AND($MPC4S3C3O2Used, $MPC4S3C3O2Dpos)
$MPC4S3C3O2UsedAndDneg = AND($MPC4S3C3O2Used, $MPC4S3C3O2Dneg)
$MPC4S3C3O2UsedAndSOK  = AND($MPC4S3C3O2Used, $MPC4S3C3O2SOK)

//  Slot3, Channel 4, Output 1
$MPC4S3C4O1Used = Slot3:MPC4:Config:C4:O1:Used
$MPC4S3C4O1Apos = Slot3:MPC4:C4:O1:A+
$MPC4S3C4O1Aneg = Slot3:MPC4:C4:O1:A-
$MPC4S3C4O1Dpos = Slot3:MPC4:C4:O1:D+
$MPC4S3C4O1Dneg = Slot3:MPC4:C4:O1:D-
$MPC4S3C4O1SOK  = Slot3:MPC4:Status:C4:SOK
$MPC4S3C4O1UsedAndApos = AND($MPC4S3C4O1Used, $MPC4S3C4O1Apos)
$MPC4S3C4O1UsedAndAneg = AND($MPC4S3C4O1Used, $MPC4S3C4O1Aneg)
$MPC4S3C4O1UsedAndDpos = AND($MPC4S3C4O1Used, $MPC4S3C4O1Dpos)
$MPC4S3C4O1UsedAndDneg = AND($MPC4S3C4O1Used, $MPC4S3C4O1Dneg)
$MPC4S3C4O1UsedAndSOK  = AND($MPC4S3C4O1Used, $MPC4S3C4O1SOK)

//  Slot3, Channel 4, Output 2
$MPC4S3C4O2Used = Slot3:MPC4:Config:C4:O2:Used
$MPC4S3C4O2Apos = Slot3:MPC4:C4:O2:A+
$MPC4S3C4O2Aneg = Slot3:MPC4:C4:O2:A-
$MPC4S3C4O2Dpos = Slot3:MPC4:C4:O2:D+
$MPC4S3C4O2Dneg = Slot3:MPC4:C4:O2:D-
$MPC4S3C4O2SOK  = Slot3:MPC4:Status:C4:SOK
$MPC4S3C4O2UsedAndApos = AND($MPC4S3C4O2Used, $MPC4S3C4O2Apos)
$MPC4S3C4O2UsedAndAneg = AND($MPC4S3C4O2Used, $MPC4S3C4O2Aneg)
$MPC4S3C4O2UsedAndDpos = AND($MPC4S3C4O2Used, $MPC4S3C4O2Dpos)
$MPC4S3C4O2UsedAndDneg = AND($MPC4S3C4O2Used, $MPC4S3C4O2Dneg)
$MPC4S3C4O2UsedAndSOK  = AND($MPC4S3C4O2Used, $MPC4S3C4O2SOK)

//  Slot3, Multi-Channel 1
$MPC4S3M1Used = Slot3:MPC4:Config:M1:Used
$MPC4S3M1Apos = Slot3:MPC4:M1:A+
$MPC4S3M1Aneg = Slot3:MPC4:M1:A-
$MPC4S3M1Dpos = Slot3:MPC4:M1:D+
$MPC4S3M1Dneg = Slot3:MPC4:M1:D-
$MPC4S3M1SOK  = Slot3:MPC4:Status:M1:SOK
$MPC4S3M1UsedAndApos = AND($MPC4S3M1Used, $MPC4S3M1Apos)
$MPC4S3M1UsedAndAneg = AND($MPC4S3M1Used, $MPC4S3M1Aneg)
$MPC4S3M1UsedAndDpos = AND($MPC4S3M1Used, $MPC4S3M1Dpos)
$MPC4S3M1UsedAndDneg = AND($MPC4S3M1Used, $MPC4S3M1Dneg)
$MPC4S3M1UsedAndSOK  = AND($MPC4S3M1Used, $MPC4S3M1SOK)
```

```
//  Slot3, Multi-Channel 2
$MPC4S3M2Used = Slot3:MPC4:Config:M2:Used
$MPC4S3M2Apos = Slot3:MPC4:M2:A+
$MPC4S3M2Aneg = Slot3:MPC4:M2:A-
$MPC4S3M2Dpos = Slot3:MPC4:M2:D+
$MPC4S3M2Dneg = Slot3:MPC4:M2:D-
$MPC4S3M2SOK  = Slot3:MPC4:Status:M1:SOK
$MPC4S3M2UsedAndApos = AND($MPC4S3M2Used, $MPC4S3M2Apos)
$MPC4S3M2UsedAndAneg = AND($MPC4S3M2Used, $MPC4S3M2Aneg)
$MPC4S3M2UsedAndDpos = AND($MPC4S3M2Used, $MPC4S3M2Dpos)
$MPC4S3M2UsedAndDneg = AND($MPC4S3M2Used, $MPC4S3M2Dneg)
$MPC4S3M2UsedAndSOK  = AND($MPC4S3M2Used, $MPC4S3M2SOK)

//  Slot3, Speed 1
$MPC4S3S1Used = Slot3:MPC4:Config:S1:Used
$MPC4S3S1Apos = Slot3:MPC4:S1:A+
$MPC4S3S1Aneg = Slot3:MPC4:S1:A-
$MPC4S3S1SOK  = Slot3:MPC4:S1:SOK
$MPC4S3S1UsedAndApos = AND($MPC4S3S1Used, $MPC4S3S1Apos)
$MPC4S3S1UsedAndAneg = AND($MPC4S3S1Used, $MPC4S3S1Aneg)
$MPC4S3S1UsedAndSOK  = AND($MPC4S3S1Used, $MPC4S3S1SOK)

//  Slot3, Speed 2
$MPC4S3S2Used = Slot3:MPC4:Config:S2:Used
$MPC4S3S2Apos = Slot3:MPC4:S2:A+
$MPC4S3S2Aneg = Slot3:MPC4:S2:A-
$MPC4S3S2SOK  = Slot3:MPC4:S2:SOK
$MPC4S3S2UsedAndApos = AND($MPC4S3S2Used, $MPC4S3S2Apos)
$MPC4S3S2UsedAndAneg = AND($MPC4S3S2Used, $MPC4S3S2Aneg)
$MPC4S3S2UsedAndSOK  = AND($MPC4S3S2Used, $MPC4S3S2SOK)

//  Slot3, Basic and Advanced Functions
$MPC4S3BF1 = Slot3:MPC4:Logical:BF1
$MPC4S3BF2 = Slot3:MPC4:Logical:BF2
$MPC4S3BF3 = Slot3:MPC4:Logical:BF3
$MPC4S3BF4 = Slot3:MPC4:Logical:BF4
$MPC4S3BF5 = Slot3:MPC4:Logical:BF5
$MPC4S3BF6 = Slot3:MPC4:Logical:BF6
$MPC4S3BF7 = Slot3:MPC4:Logical:BF7
$MPC4S3BF8 = Slot3:MPC4:Logical:BF8
$MPC4S3AF1 = Slot3:MPC4:Logical:AF1
$MPC4S3AF2 = Slot3:MPC4:Logical:AF2
$MPC4S3AF3 = Slot3:MPC4:Logical:AF3
$MPC4S3AF4 = Slot3:MPC4:Logical:AF4
```

```
//  Slot3, Common Monitoring Functions
$MPC4S3MCR  = Slot3:MPC4:CommonStatus:MCR
$MPC4S3CMF  = Slot3:MPC4:CommonStatus:CMF
$MPC4S3COF  = Slot3:MPC4:CommonStatus:COF
$MPC4S3CPE  = Slot3:MPC4:CommonStatus:CPE
$MPC4S3CIE  = Slot3:MPC4:CommonStatus:CIE
$MPC4S3CA   = Slot3:MPC4:CommonStatus:CA
$MPC4S3CD   = Slot3:MPC4:CommonStatus:CD
$MPC4S3CSOL = Slot3:MPC4:CommonStatus:CSOL
$MPC4S3CTL  = Slot3:MPC4:CommonStatus:CTL
$MPC4S3CTOR = Slot3:MPC4:CommonStatus:CTOR
$MPC4S3CDSE = Slot3:MPC4:CommonStatus:CDSE
$MPC4S3CISE = Slot3:MPC4:CommonStatus:CISE
$MPC4S3CCME = Slot3:MPC4:CommonStatus:CCME
$MPC4S3SL   = Slot3:MPC4:CommonStatus:SL
$MPC4S3TM   = Slot3:MPC4:CommonStatus:TM
$MPC4S3DBP  = Slot3:MPC4:CommonStatus:DBP
$MPC4S3AR   = Slot3:MPC4:CommonStatus:AR



00000:3:U = Slot3:MPC4:C1:O1:V
00001:3:U = Slot3:MPC4:C1:O1:FSD
00002:3:U = Slot3:MPC4:C1:O2:V
00003:3:U = Slot3:MPC4:C1:O2:FSD
00004:3:U = Slot3:MPC4:C2:O1:V
00005:3:U = Slot3:MPC4:C2:O1:FSD
00006:3:U = Slot3:MPC4:C2:O2:V
00007:3:U = Slot3:MPC4:C2:O2:FSD
00008:3:U = Slot3:MPC4:C3:O1:V
00009:3:U = Slot3:MPC4:C3:O1:FSD
00010:3:U = Slot3:MPC4:C3:O2:V
00011:3:U = Slot3:MPC4:C3:O2:FSD
00012:3:U = Slot3:MPC4:C4:O1:V
00013:3:U = Slot3:MPC4:C4:O1:FSD
00014:3:U = Slot3:MPC4:C4:O2:V
00015:3:U = Slot3:MPC4:C4:O2:FSD
00016:3:U = Slot3:MPC4:M1:V
00017:3:U = Slot3:MPC4:M1:FSD
00018:3:U = Slot3:MPC4:M2:V
00019:3:U = Slot3:MPC4:M2:FSD
00020:3:U = Slot3:MPC4:S1:SPEED
00021:3:U = Slot3:MPC4:S2:SPEED
00022:3:U = Pack($MPC4S3C1O1Used,$MPC4S3C1O1UsedAndApos,
$MPC4S3C1O1UsedAndAneg, $MPC4S3C1O1UsedAndDpos,$MPC4S3C1O1UsedAndDneg,
$MPC4S3C1O1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
```

```
00023:3:U = Pack($MPC4S3C1O2Used,$MPC4S3C1O2UsedAndApos,
$MPC4S3C1O2UsedAndAneg,$MPC4S3C1O2UsedAndDpos,$MPC4S3C1O2UsedAndDneg,
$MPC4S3C1O2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00024:3:U = Pack($MPC4S3C2O1Used,$MPC4S3C2O1UsedAndApos,
$MPC4S3C2O1UsedAndAneg,$MPC4S3C2O1UsedAndDpos,$MPC4S3C2O1UsedAndDneg,
$MPC4S3C2O1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00025:3:U = Pack($MPC4S3C2O2Used,$MPC4S3C2O2UsedAndApos,
$MPC4S3C2O2UsedAndAneg,$MPC4S3C2O2UsedAndDpos,$MPC4S3C2O2UsedAndDneg,
$MPC4S3C2O2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00026:3:U = Pack($MPC4S3C3O1Used,$MPC4S3C3O1UsedAndApos,
$MPC4S3C3O1UsedAndAneg,$MPC4S3C3O1UsedAndDpos,$MPC4S3C3O1UsedAndDneg,
$MPC4S3C3O1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00027:3:U = Pack($MPC4S3C3O2Used,$MPC4S3C3O2UsedAndApos,
$MPC4S3C3O2UsedAndAneg,$MPC4S3C3O2UsedAndDpos,$MPC4S3C3O2UsedAndDneg,
$MPC4S3C3O2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00028:3:U = Pack($MPC4S3C4O1Used,$MPC4S3C4O1UsedAndApos,
$MPC4S3C4O1UsedAndAneg,$MPC4S3C4O1UsedAndDpos,$MPC4S3C4O1UsedAndDneg,
$MPC4S3C4O1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00029:3:U = Pack($MPC4S3C4O2Used,$MPC4S3C4O2UsedAndApos,
$MPC4S3C4O2UsedAndAneg,$MPC4S3C4O2UsedAndDpos,$MPC4S3C4O2UsedAndDneg,
$MPC4S3C4O2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00030:3:U = Pack($MPC4S3M1Used,$MPC4S3M1UsedAndApos,
$MPC4S3M1UsedAndAneg,$MPC4S3M1UsedAndDpos,$MPC4S3M1UsedAndDneg,
$MPC4S3M1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00031:3:U = Pack($MPC4S3M2Used,$MPC4S3M2UsedAndApos,
$MPC4S3M2UsedAndAneg,$MPC4S3M2UsedAndDpos,$MPC4S3M2UsedAndDneg,
$MPC4S3M2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00032:3:U = Pack($MPC4S3S1Used,$MPC4S3S1UsedAndApos,
$MPC4S3S1UsedAndAneg,0,0,$MPC4S3S1UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00033:3:U = Pack($MPC4S3S2Used,$MPC4S3S2UsedAndApos,
$MPC4S3S2UsedAndAneg,0,0,$MPC4S3S2UsedAndSOK,0,0,0,0,0,0,0,0,0,0)
00034:3:U = PACK($MPC4S3BF1,$MPC4S3BF2,$MPC4S3BF3,$MPC4S3BF4,
$MPC4S3BF5,$MPC4S3BF6,$MPC4S3BF7,$MPC4S3BF8,$MPC4S3AF1,$MPC4S3AF2,
$MPC4S3AF3,$MPC4S3AF4,0,0,0,0)
00035:3:U = PACK($MPC4S3CMF,$MPC4S3COF,$MPC4S3CPE,$MPC4S3CIE,
$MPC4S3CA,$MPC4S3CD,$MPC4S3CSOL,$MPC4S3CTL,$MPC4S3CTOR,$MPC4S3CDSE,
$MPC4S3CISE,$MPC4S3CCME,$MPC4S3SL,$MPC4S3TM,$MPC4S3DBP,$MPC4S3AR)
00036:3:U = PACK($MPC4S3MCR,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

...

PROFINET_CREATE SLOT3 0 36
```

As shown above, in the extract for an MPC4 card in slot 3:

- PROFINET is enabled in the `[GLOBAL]` section using the `IS_PROFINET_ACTIVED` tag.
- MPC4 outputs are defined and organised in the `[MAPPING]` section, on a slot by slot (card) basis, using `$MPC4...` tags.

NOTE: `$MPC4...` tags can use logical operators to combine status flags (bits).

- MPC4 Modbus register mapping is defined in the `[MAPPING]` section, on a slot by slot (card) basis, using the Modbus register mapping syntax.

  For example: `00000:3:U = Slot3:MPC4:C1:01:V`
  See 9.2.4 The MAPPING section for further information.

NOTE: Modbus register mapping can use the pack command (`Pack(...)`) to combine `$MPC4...` tags.

- PROFINET slots are defined in the `[MAPPING]` section, at the end of the configuration file, using `PROFINET_CREATE` tags.

  For example: `PROFINET_CREATE SLOT3 0 36`, where

  - `SLOT3` is the PROFINET slot ID.
    The PROFINET slot ID number can correspond to the VM600 rack slot number or a different numbering scheme can be used.
  - `0` is the first Modbus register address to map to this PROFINET slot.
  - `36` is the last Modbus register address to map to this PROFINET slot.
    The first and last Modbus register addresses mapped to a PROFINET slot must contain a contiguous data block.

The underlying PROFINET slot data structures are defined in a GSDML file (see 10.5 GSDML files). The MPC4 Modbus register mapping and the PROFINET slot definitions configured in the `[MAPPING]` section of the configuration file must match the data structures defined in the GSDML file.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D: AMC8 PROFINET SLOT DEFINITIONS

AMC8 card slot definitions for PROFINET are illustrated using the following extract from a typical `modbusDefault.cfg` configuration file for an AMC8 card in slot 7 of a VM600 rack.

```
...

[GLOBAL]
...
IS_PROFINET_ACTIVED = YES

[MAPPING]

/////////////////////////////////////////////////
///////////////// Slot7 ////////////////////////
/////////////////////////////////////////////////

///////////////// Alarm and Status //////////////

//  Slot 7, Alarm+
$S7C1APos = Slot7:AMC8:C1:A+
$S7C2APos = Slot7:AMC8:C2:A+
$S7C3APos = Slot7:AMC8:C3:A+
$S7C4APos = Slot7:AMC8:C4:A+
$S7C5APos = Slot7:AMC8:C5:A+
$S7C6APos = Slot7:AMC8:C6:A+
$S7C7APos = Slot7:AMC8:C7:A+
$S7C8APos = Slot7:AMC8:C8:A+
$S7M1APos = Slot7:AMC8:M1:A+
$S7M2APos = Slot7:AMC8:M2:A+
$S7M3APos = Slot7:AMC8:M3:A+
$S7M4APos = Slot7:AMC8:M4:A+

//  Slot 7, Alarm-
$S7C1ANeg = Slot7:AMC8:C1:A-
$S7C2ANeg = Slot7:AMC8:C2:A-
$S7C3ANeg = Slot7:AMC8:C3:A-
$S7C4ANeg = Slot7:AMC8:C4:A-
$S7C5ANeg = Slot7:AMC8:C5:A-
$S7C6ANeg = Slot7:AMC8:C6:A-
$S7C7ANeg = Slot7:AMC8:C7:A-
$S7C8ANeg = Slot7:AMC8:C8:A-
$S7M1ANeg = Slot7:AMC8:M1:A-
$S7M2ANeg = Slot7:AMC8:M2:A-
```

```
$S7M3ANeg = Slot7:AMC8:M3:A-
$S7M4ANeg = Slot7:AMC8:M4:A-

//  Slot 7, Danger+
$S7C1DPos = Slot7:AMC8:C1:D+
$S7C2DPos = Slot7:AMC8:C2:D+
$S7C3DPos = Slot7:AMC8:C3:D+
$S7C4DPos = Slot7:AMC8:C4:D+
$S7C5DPos = Slot7:AMC8:C5:D+
$S7C6DPos = Slot7:AMC8:C6:D+
$S7C7DPos = Slot7:AMC8:C7:D+
$S7C8DPos = Slot7:AMC8:C8:D+
$S7M1DPos = Slot7:AMC8:M1:D+
$S7M2DPos = Slot7:AMC8:M2:D+
$S7M3DPos = Slot7:AMC8:M3:D+
$S7M4DPos = Slot7:AMC8:M4:D+

//  Slot 7, Danger-
$S7C1DNeg = Slot7:AMC8:C1:D-
$S7C2DNeg = Slot7:AMC8:C2:D-
$S7C3DNeg = Slot7:AMC8:C3:D-
$S7C4DNeg = Slot7:AMC8:C4:D-
$S7C5DNeg = Slot7:AMC8:C5:D-
$S7C6DNeg = Slot7:AMC8:C6:D-
$S7C7DNeg = Slot7:AMC8:C7:D-
$S7C8DNeg = Slot7:AMC8:C8:D-
$S7M1DNeg = Slot7:AMC8:M1:D-
$S7M2DNeg = Slot7:AMC8:M2:D-
$S7M3DNeg = Slot7:AMC8:M3:D-
$S7M4DNeg = Slot7:AMC8:M4:D-

//  Slot 7, Global Fail
$S7C1GOKF = Slot7:AMC8:C1:GOKF
$S7C2GOKF = Slot7:AMC8:C2:GOKF
$S7C3GOKF = Slot7:AMC8:C3:GOKF
$S7C4GOKF = Slot7:AMC8:C4:GOKF
$S7C5GOKF = Slot7:AMC8:C5:GOKF
$S7C6GOKF = Slot7:AMC8:C6:GOKF
$S7C7GOKF = Slot7:AMC8:C7:GOKF
$S7C8GOKF = Slot7:AMC8:C8:GOKF
$S7M1GOKF = Slot7:AMC8:M1:GOKF
$S7M2GOKF = Slot7:AMC8:M2:GOKF
$S7M3GOKF = Slot7:AMC8:M3:GOKF
$S7M4GOKF = Slot7:AMC8:M4:GOKF
```

```
//  Slot 7, ADC Error
$S7C1ADERR = Slot7:AMC8:C1:AD_ERR
$S7C2ADERR = Slot7:AMC8:C2:AD_ERR
$S7C3ADERR = Slot7:AMC8:C3:AD_ERR
$S7C4ADERR = Slot7:AMC8:C4:AD_ERR
$S7C5ADERR = Slot7:AMC8:C5:AD_ERR
$S7C6ADERR = Slot7:AMC8:C6:AD_ERR
$S7C7ADERR = Slot7:AMC8:C7:AD_ERR
$S7C8ADERR = Slot7:AMC8:C8:AD_ERR
$S7M1ADERR = Slot7:AMC8:M1:AD_ERR
$S7M2ADERR = Slot7:AMC8:M2:AD_ERR
$S7M3ADERR = Slot7:AMC8:M3:AD_ERR
$S7M4ADERR = Slot7:AMC8:M4:AD_ERR

//  Slot 7, ADC Stdby
$S7C1ADSTBY = Slot7:AMC8:C1:AD_STBY
$S7C2ADSTBY = Slot7:AMC8:C2:AD_STBY
$S7C3ADSTBY = Slot7:AMC8:C3:AD_STBY
$S7C4ADSTBY = Slot7:AMC8:C4:AD_STBY
$S7C5ADSTBY = Slot7:AMC8:C5:AD_STBY
$S7C6ADSTBY = Slot7:AMC8:C6:AD_STBY
$S7C7ADSTBY = Slot7:AMC8:C7:AD_STBY
$S7C8ADSTBY = Slot7:AMC8:C8:AD_STBY
$S7M1ADSTBY = Slot7:AMC8:M1:AD_STBY
$S7M2ADSTBY = Slot7:AMC8:M2:AD_STBY
$S7M3ADSTBY = Slot7:AMC8:M3:AD_STBY
$S7M4ADSTBY = Slot7:AMC8:M4:AD_STBY

//  Slot 7, ADC-PLL_LockErr
$S7C1ADLOCK = Slot7:AMC8:C1:AD_LOCK
$S7C2ADLOCK = Slot7:AMC8:C2:AD_LOCK
$S7C3ADLOCK = Slot7:AMC8:C3:AD_LOCK
$S7C4ADLOCK = Slot7:AMC8:C4:AD_LOCK
$S7C5ADLOCK = Slot7:AMC8:C5:AD_LOCK
$S7C6ADLOCK = Slot7:AMC8:C6:AD_LOCK
$S7C7ADLOCK = Slot7:AMC8:C7:AD_LOCK
$S7C8ADLOCK = Slot7:AMC8:C8:AD_LOCK
$S7M1ADLOCK = Slot7:AMC8:M1:AD_LOCK
$S7M2ADLOCK = Slot7:AMC8:M2:AD_LOCK
$S7M3ADLOCK = Slot7:AMC8:M3:AD_LOCK
$S7M4ADLOCK = Slot7:AMC8:M4:AD_LOCK

//  Slot 7, ADC_TxErr
$S7C1ADTXERR = Slot7:AMC8:C1:AD_TXERR
$S7C2ADTXERR = Slot7:AMC8:C2:AD_TXERR
```

```
$S7C3ADTXERR = Slot7:AMC8:C3:AD_TXERR
$S7C4ADTXERR = Slot7:AMC8:C4:AD_TXERR
$S7C5ADTXERR = Slot7:AMC8:C5:AD_TXERR
$S7C6ADTXERR = Slot7:AMC8:C6:AD_TXERR
$S7C7ADTXERR = Slot7:AMC8:C7:AD_TXERR
$S7C8ADTXERR = Slot7:AMC8:C8:AD_TXERR
$S7M1ADTXERR = Slot7:AMC8:M1:AD_TXERR
$S7M2ADTXERR = Slot7:AMC8:M2:AD_TXERR
$S7M3ADTXERR = Slot7:AMC8:M3:AD_TXERR
$S7M4ADTXERR = Slot7:AMC8:M4:AD_TXERR

//  Slot 7, ADC_DynConfErr
$S7C1ADCFGERR = Slot7:AMC8:C1:AD_CFG_ERR
$S7C2ADCFGERR = Slot7:AMC8:C2:AD_CFG_ERR
$S7C3ADCFGERR = Slot7:AMC8:C3:AD_CFG_ERR
$S7C4ADCFGERR = Slot7:AMC8:C4:AD_CFG_ERR
$S7C5ADCFGERR = Slot7:AMC8:C5:AD_CFG_ERR
$S7C6ADCFGERR = Slot7:AMC8:C6:AD_CFG_ERR
$S7C7ADCFGERR = Slot7:AMC8:C7:AD_CFG_ERR
$S7C8ADCFGERR = Slot7:AMC8:C8:AD_CFG_ERR
$S7M1ADCFGERR = Slot7:AMC8:M1:AD_CFG_ERR
$S7M2ADCFGERR = Slot7:AMC8:M2:AD_CFG_ERR
$S7M3ADCFGERR = Slot7:AMC8:M3:AD_CFG_ERR
$S7M4ADCFGERR = Slot7:AMC8:M4:AD_CFG_ERR

//  Slot 7, Bit_resultFail
$S7C1BITF = Slot7:AMC8:C1:BITF
$S7C2BITF = Slot7:AMC8:C2:BITF
$S7C3BITF = Slot7:AMC8:C3:BITF
$S7C4BITF = Slot7:AMC8:C4:BITF
$S7C5BITF = Slot7:AMC8:C5:BITF
$S7C6BITF = Slot7:AMC8:C6:BITF
$S7C7BITF = Slot7:AMC8:C7:BITF
$S7C8BITF = Slot7:AMC8:C8:BITF
$S7M1BITF = Slot7:AMC8:M1:BITF
$S7M2BITF = Slot7:AMC8:M2:BITF
$S7M3BITF = Slot7:AMC8:M3:BITF
$S7M4BITF = Slot7:AMC8:M4:BITF

//  Slot 7, NoSample
$S7C1NOSPL = Slot7:AMC8:C1:NOSPL
$S7C2NOSPL = Slot7:AMC8:C2:NOSPL
$S7C3NOSPL = Slot7:AMC8:C3:NOSPL
$S7C4NOSPL = Slot7:AMC8:C4:NOSPL
$S7C5NOSPL = Slot7:AMC8:C5:NOSPL
```

```
$S7C6NOSPL = Slot7:AMC8:C6:NOSPL
$S7C7NOSPL = Slot7:AMC8:C7:NOSPL
$S7C8NOSPL = Slot7:AMC8:C8:NOSPL
$S7M1NOSPL = Slot7:AMC8:M1:NOSPL
$S7M2NOSPL = Slot7:AMC8:M2:NOSPL
$S7M3NOSPL = Slot7:AMC8:M3:NOSPL
$S7M4NOSPL = Slot7:AMC8:M4:NOSPL

//  Slot 7, OKLevelFail
$S7C1OKF = Slot7:AMC8:C1:OKF
$S7C2OKF = Slot7:AMC8:C2:OKF
$S7C3OKF = Slot7:AMC8:C3:OKF
$S7C4OKF = Slot7:AMC8:C4:OKF
$S7C5OKF = Slot7:AMC8:C5:OKF
$S7C6OKF = Slot7:AMC8:C6:OKF
$S7C7OKF = Slot7:AMC8:C7:OKF
$S7C8OKF = Slot7:AMC8:C8:OKF
$S7M1OKF = Slot7:AMC8:M1:OKF
$S7M2OKF = Slot7:AMC8:M2:OKF
$S7M3OKF = Slot7:AMC8:M3:OKF
$S7M4OKF = Slot7:AMC8:M4:OKF

//  Slot 7, LinearizationErr
$S7C1LINERR = Slot7:AMC8:C1:LINERR
$S7C2LINERR = Slot7:AMC8:C2:LINERR
$S7C3LINERR = Slot7:AMC8:C3:LINERR
$S7C4LINERR = Slot7:AMC8:C4:LINERR
$S7C5LINERR = Slot7:AMC8:C5:LINERR
$S7C6LINERR = Slot7:AMC8:C6:LINERR
$S7C7LINERR = Slot7:AMC8:C7:LINERR
$S7C8LINERR = Slot7:AMC8:C8:LINERR
$S7M1LINERR = Slot7:AMC8:M1:LINERR
$S7M2LINERR = Slot7:AMC8:M2:LINERR
$S7M3LINERR = Slot7:AMC8:M3:LINERR
$S7M4LINERR = Slot7:AMC8:M4:LINERR

//  Slot 7, CJT_Err
$S7C1CJERR = Slot7:AMC8:C1:CJERR
$S7C2CJERR = Slot7:AMC8:C2:CJERR
$S7C3CJERR = Slot7:AMC8:C3:CJERR
$S7C4CJERR = Slot7:AMC8:C4:CJERR
$S7C5CJERR = Slot7:AMC8:C5:CJERR
$S7C6CJERR = Slot7:AMC8:C6:CJERR
$S7C7CJERR = Slot7:AMC8:C7:CJERR
$S7C8CJERR = Slot7:AMC8:C8:CJERR
```

```
$S7M1CJERR = Slot7:AMC8:M1:CJERR
$S7M2CJERR = Slot7:AMC8:M2:CJERR
$S7M3CJERR = Slot7:AMC8:M3:CJERR
$S7M4CJERR = Slot7:AMC8:M4:CJERR

//  Basic function
$S7BF1 = Slot7:AMC8:Logical:BF1
$S7BF2 = Slot7:AMC8:Logical:BF2
$S7BF3 = Slot7:AMC8:Logical:BF3
$S7BF4 = Slot7:AMC8:Logical:BF4
$S7BF5 = Slot7:AMC8:Logical:BF5
$S7BF6 = Slot7:AMC8:Logical:BF6
$S7BF7 = Slot7:AMC8:Logical:BF7
$S7BF8 = Slot7:AMC8:Logical:BF8
$S7BF9 = Slot7:AMC8:Logical:BF9
$S7BF10 = Slot7:AMC8:Logical:BF10
$S7BF11 = Slot7:AMC8:Logical:BF11
$S7BF12 = Slot7:AMC8:Logical:BF12
$S7BF13 = Slot7:AMC8:Logical:BF13
$S7BF14 = Slot7:AMC8:Logical:BF14
$S7BF15 = Slot7:AMC8:Logical:BF15
$S7BF16 = Slot7:AMC8:Logical:BF16

//  Advanced function
$S7AF1 = Slot7:AMC8:Logical:AF1
$S7AF2 = Slot7:AMC8:Logical:AF2
$S7AF3 = Slot7:AMC8:Logical:AF3
$S7AF4 = Slot7:AMC8:Logical:AF4
$S7AF5 = Slot7:AMC8:Logical:AF5
$S7AF6 = Slot7:AMC8:Logical:AF6
$S7AF7 = Slot7:AMC8:Logical:AF7
$S7AF8 = Slot7:AMC8:Logical:AF8

// Alarms and Dangers used //
$S7C1ALOWUSED = Slot7:AMC8:Config:C1:A-:Used
$S7C2ALOWUSED = Slot7:AMC8:Config:C2:A-:Used
$S7C3ALOWUSED = Slot7:AMC8:Config:C3:A-:Used
$S7C4ALOWUSED = Slot7:AMC8:Config:C4:A-:Used
$S7C5ALOWUSED = Slot7:AMC8:Config:C5:A-:Used
$S7C6ALOWUSED = Slot7:AMC8:Config:C6:A-:Used
$S7C7ALOWUSED = Slot7:AMC8:Config:C7:A-:Used
$S7C8ALOWUSED = Slot7:AMC8:Config:C8:A-:Used
$S7M1ALOWUSED = Slot7:AMC8:Config:M1:A-:Used
$S7M2ALOWUSED = Slot7:AMC8:Config:M2:A-:Used
$S7M3ALOWUSED = Slot7:AMC8:Config:M3:A-:Used
```

```
$S7M4ALOWUSED = Slot7:AMC8:Config:M4:A-:Used
$S7C1AHIGHUSED = Slot7:AMC8:Config:C1:A+:Used
$S7C2AHIGHUSED = Slot7:AMC8:Config:C2:A+:Used
$S7C3AHIGHUSED = Slot7:AMC8:Config:C3:A+:Used
$S7C4AHIGHUSED = Slot7:AMC8:Config:C4:A+:Used
$S7C5AHIGHUSED = Slot7:AMC8:Config:C5:A+:Used
$S7C6AHIGHUSED = Slot7:AMC8:Config:C6:A+:Used
$S7C7AHIGHUSED = Slot7:AMC8:Config:C7:A+:Used
$S7C8AHIGHUSED = Slot7:AMC8:Config:C8:A+:Used
$S7M1AHIGHUSED = Slot7:AMC8:Config:M1:A+:Used
$S7M2AHIGHUSED = Slot7:AMC8:Config:M2:A+:Used
$S7M3AHIGHUSED = Slot7:AMC8:Config:M3:A+:Used
$S7M4AHIGHUSED = Slot7:AMC8:Config:M4:A+:Used
$S7C1DLOWUSED = Slot7:AMC8:Config:C1:D-:Used
$S7C2DLOWUSED = Slot7:AMC8:Config:C2:D-:Used
$S7C3DLOWUSED = Slot7:AMC8:Config:C3:D-:Used
$S7C4DLOWUSED = Slot7:AMC8:Config:C4:D-:Used
$S7C5DLOWUSED = Slot7:AMC8:Config:C5:D-:Used
$S7C6DLOWUSED = Slot7:AMC8:Config:C6:D-:Used
$S7C7DLOWUSED = Slot7:AMC8:Config:C7:D-:Used
$S7C8DLOWUSED = Slot7:AMC8:Config:C8:D-:Used
$S7M1DLOWUSED = Slot7:AMC8:Config:M1:D-:Used
$S7M2DLOWUSED = Slot7:AMC8:Config:M2:D-:Used
$S7M3DLOWUSED = Slot7:AMC8:Config:M3:D-:Used
$S7M4DLOWUSED = Slot7:AMC8:Config:M4:D-:Used
$S7C1DHIGHUSED = Slot7:AMC8:Config:C1:D+:Used
$S7C2DHIGHUSED = Slot7:AMC8:Config:C2:D+:Used
$S7C3DHIGHUSED = Slot7:AMC8:Config:C3:D+:Used
$S7C4DHIGHUSED = Slot7:AMC8:Config:C4:D+:Used
$S7C5DHIGHUSED = Slot7:AMC8:Config:C5:D+:Used
$S7C6DHIGHUSED = Slot7:AMC8:Config:C6:D+:Used
$S7C7DHIGHUSED = Slot7:AMC8:Config:C7:D+:Used
$S7C8DHIGHUSED = Slot7:AMC8:Config:C8:D+:Used
$S7M1DHIGHUSED = Slot7:AMC8:Config:M1:D+:Used
$S7M2DHIGHUSED = Slot7:AMC8:Config:M2:D+:Used
$S7M3DHIGHUSED = Slot7:AMC8:Config:M3:D+:Used
$S7M4DHIGHUSED = Slot7:AMC8:Config:M4:A+:Used

//  Slot 7, Common Channel Status //
$S7ACNR = Slot7:AMC8:CommonStatus:ACNR
$S7AR   = Slot7:AMC8:CommonStatus:AR
$S7DBP  = Slot7:AMC8:CommonStatus:DBP
$S7SL   = Slot7:AMC8:CommonStatus:SL
```

```
00064:3:F = Slot7:AMC8:C1:V
00066:3:F = Slot7:AMC8:C2:V
00068:3:F = Slot7:AMC8:C3:V
00070:3:F = Slot7:AMC8:C4:V
00072:3:F = Slot7:AMC8:C5:V
00074:3:F = Slot7:AMC8:C6:V
00076:3:F = Slot7:AMC8:C7:V
00078:3:F = Slot7:AMC8:C8:V
00080:3:F = Slot7:AMC8:M1:V
00082:3:F = Slot7:AMC8:M2:V
00084:3:F = Slot7:AMC8:M3:V
00086:3:F = Slot7:AMC8:M4:V
00088:3:U = Pack($S7BF1,$S7BF2,$S7BF3,$S7BF4,$S7BF5,$S7BF6,$S7BF7,$S7BF8,
$S7BF9,$S7BF10,$S7BF11,$S7BF12,$S7BF13,$S7BF14,$S7BF15,$S7BF16)
00089:3:U = Pack($S7AF1,$S7AF2,$S7AF3,$S7AF4,$S7AF5,$S7AF6,$S7AF7,$S7AF8,
0,0,0,0,0,0,0,0)
00090:3:U = Pack($S7C1APos,$S7C1ANeg,$S7C1DPos,$S7C1DNeg,$S7C1GOKF,
$S7C1ADERR,$S7C1ADSTBY,$S7C1ADLOCK,$S7C1ADTXERR,$S7C1ADCFGERR,$S7C1BITF,
$S7C1NOSPL,$S7C1OKF,$S7C1LINERR,$S7C1CJERR,0)
00091:3:U = Pack($S7C2APos,$S7C2ANeg,$S7C2DPos,$S7C2DNeg,$S7C2GOKF,
$S7C2ADERR,$S7C2ADSTBY,$S7C2ADLOCK,$S7C2ADTXERR,$S7C2ADCFGERR,$S7C2BITF,
$S7C2NOSPL,$S7C2OKF,$S7C2LINERR,$S7C2CJERR,0)
00092:3:U = Pack($S7C3APos,$S7C3ANeg,$S7C3DPos,$S7C3DNeg,$S7C3GOKF,
$S7C3ADERR,$S7C3ADSTBY,$S7C3ADLOCK,$S7C3ADTXERR,$S7C3ADCFGERR,$S7C3BITF,
$S7C3NOSPL,$S7C3OKF,$S7C3LINERR,$S7C3CJERR,0)
00093:3:U = Pack($S7C4APos,$S7C4ANeg,$S7C4DPos,$S7C4DNeg,$S7C4GOKF,
$S7C4ADERR,$S7C4ADSTBY,$S7C4ADLOCK,$S7C4ADTXERR,$S7C4ADCFGERR,$S7C4BITF,
$S7C4NOSPL,$S7C4OKF,$S7C4LINERR,$S7C4CJERR,0)
00094:3:U = Pack($S7C5APos,$S7C5ANeg,$S7C5DPos,$S7C5DNeg,$S7C5GOKF,
$S7C5ADERR,$S7C5ADSTBY,$S7C5ADLOCK,$S7C5ADTXERR,$S7C5ADCFGERR,$S7C5BITF,
$S7C5NOSPL,$S7C5OKF,$S7C5LINERR,$S7C5CJERR,0)
00095:3:U = Pack($S7C6APos,$S7C6ANeg,$S7C6DPos,$S7C6DNeg,$S7C6GOKF,
$S7C6ADERR,$S7C6ADSTBY,$S7C6ADLOCK,$S7C6ADTXERR,$S7C6ADCFGERR,$S7C6BITF,
$S7C6NOSPL,$S7C6OKF,$S7C6LINERR,$S7C6CJERR,0)
00096:3:U = Pack($S7C7APos,$S7C7ANeg,$S7C7DPos,$S7C7DNeg,$S7C7GOKF,
$S7C7ADERR,$S7C7ADSTBY,$S7C7ADLOCK,$S7C7ADTXERR,$S7C7ADCFGERR,$S7C7BITF,
$S7C7NOSPL,$S7C7OKF,$S7C7LINERR,$S7C7CJERR,0)
00097:3:U = Pack($S7C8APos,$S7C8ANeg,$S7C8DPos,$S7C8DNeg,$S7C8GOKF,
$S7C8ADERR,$S7C8ADSTBY,$S7C8ADLOCK,$S7C8ADTXERR,$S7C8ADCFGERR,$S7C8BITF,
$S7C8NOSPL,$S7C8OKF,$S7C8LINERR,$S7C8CJERR,0)
00098:3:U = Pack($S7M1APos,$S7M1ANeg,$S7M1DPos,$S7M1DNeg,$S7M1GOKF,
$S7M1ADERR,$S7M1ADSTBY,$S7M1ADLOCK,$S7M1ADTXERR,$S7M1ADCFGERR,$S7M1BITF,
$S7M1NOSPL,$S7M1OKF,$S7M1LINERR,$S7M1CJERR,0)
00099:3:U = Pack($S7M2APos,$S7M2ANeg,$S7M2DPos,$S7M2DNeg,$S7M2GOKF,
$S7M2ADERR,$S7M2ADSTBY,$S7M2ADLOCK,$S7M2ADTXERR,$S7M2ADCFGERR,$S7M2BITF,
$S7M2NOSPL,$S7M2OKF,$S7M2LINERR,$S7M2CJERR,0)
00100:3:U = Pack($S7M3APos,$S7M3ANeg,$S7M3DPos,$S7M3DNeg,$S7M3GOKF,
$S7M3ADERR,$S7M3ADSTBY,$S7M3ADLOCK,$S7M3ADTXERR,$S7M3ADCFGERR,$S7M3BITF,
$S7M3NOSPL,$S7M3OKF,$S7M3LINERR,$S7M3CJERR,0)
```

```
00101:3:U = Pack($S7M4APos,$S7M4ANeg,$S7M4DPos,$S7M4DNeg,$S7M4GOKF,
$S7M4ADERR,$S7M4ADSTBY,$S7M4ADLOCK,$S7M4ADTXERR,$S7M4ADCFGERR,$S7M4BITF,
$S7M4NOSPL,$S7M4OKF,$S7M4LINERR,$S7M4CJERR,0)
00102:3:U = Pack($S7C1ALOWUSED,$S7C1AHIGHUSED,$S7C1DLOWUSED,
$S7C1DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00103:3:U = Pack($S7C2ALOWUSED,$S7C2AHIGHUSED,$S7C2DLOWUSED,
$S7C2DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00104:3:U = Pack($S7C3ALOWUSED,$S7C3AHIGHUSED,$S7C3DLOWUSED,
$S7C3DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00105:3:U = Pack($S7C4ALOWUSED,$S7C4AHIGHUSED,$S7C4DLOWUSED,
$S7C4DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00106:3:U = Pack($S7C5ALOWUSED,$S7C5AHIGHUSED,$S7C5DLOWUSED,
$S7C5DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00107:3:U = Pack($S7C6ALOWUSED,$S7C6AHIGHUSED,$S7C6DLOWUSED,
$S7C6DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00108:3:U = Pack($S7C7ALOWUSED,$S7C7AHIGHUSED,$S7C7DLOWUSED,
$S7C7DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00109:3:U = Pack($S7C8ALOWUSED,$S7C8AHIGHUSED,$S7C8DLOWUSED,
$S7C8DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00110:3:U = Pack($S7M1ALOWUSED,$S7M1AHIGHUSED,$S7M1DLOWUSED,
$S7M1DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00111:3:U = Pack($S7M2ALOWUSED,$S7M2AHIGHUSED,$S7M2DLOWUSED,
$S7M2DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00112:3:U = Pack($S7M3ALOWUSED,$S7M3AHIGHUSED,$S7M3DLOWUSED,
$S7M3DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00113:3:U = Pack($S7M4ALOWUSED,$S7M4AHIGHUSED,$S7M4DLOWUSED,
$S7M4DHIGHUSED,0,0,0,0,0,0,0,0,0,0,0,0)
00114:3:U = Pack($S7ACNR,$S7AR,$S7DBP,$S7SL,
0,0,0,0,0,0,0,0,0,0,0,0)

...

PROFINET_CREATE SLOT7 64 114
```

As shown above, in the extract for an AMC8 card in slot 7:

- PROFINET is enabled in the [GLOBAL] section using the IS_PROFINET_ACTIVED tag.
- AMC8 outputs are defined and organised in the [MAPPING] section, on a slot by slot (card) basis, using $S7... tags (for a card in slot 7).
- AMC8 Modbus register mapping is defined in the [MAPPING] section, on a slot by slot (card) basis, using the Modbus register mapping syntax.

  For example: `00064:3:F = Slot7:AMC8:C1:V`
  See 9.2.4 The MAPPING section for further information.

  NOTE: Modbus register mapping can use the pack command (Pack(...)) to combine $S7... tags.

- PROFINET slots are defined in the [MAPPING] section, at the end of the configuration file, using PROFINET_CREATE tags.

  For example: `PROFINET_CREATE SLOT7 64 114`, where

  - SLOT7 is the PROFINET slot ID.
    The PROFINET slot ID number can correspond to the VM600 rack slot number or a different numbering scheme can be used.
  - 64 is the first Modbus register address to map to this PROFINET slot.
  - 114 is the last Modbus register address to map to this PROFINET slot.
    The first and last Modbus register addresses mapped to a PROFINET slot must contain a contiguous data block.

The underlying PROFINET slot data structures are defined in a GSDML file (see 10.5 GSDML files). The AMC8 Modbus register mapping and the PROFINET slot definitions configured in the [MAPPING] section of the configuration file must match the data structures defined in the GSDML file.

APPENDIX E: SYMBOL NAMES

A list of symbol names (that is, text description tags) has been defined for the MPC4 and AMC8 cards. This list is available from Meggitt SA as a text file, on request.

This information can be used with the SIMATIC STEP 7 software, as Symbols, to make the data available at each PROFINET address easier to identify and understand.

NOTE: In the following table, n (in the Symbol name column) defines the VM600 rack slot number for a card, which can be from 3 to 14.

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	**MPS backplane configuration extractor tools syntax**	**Logical inv.**	**Normal state**	**Function, as described in the VM600 MPSx software**
MPC	Common	nMCR	Slot n (MCR)	0	1	Common MPC Diagnostics MPC Card Running
		nCMF	Slot n (CMF)	0	0	Common MPC Diagnostics Monitoring Failure
		nCOF	Slot n (COF)	0	0	Common Channels Sensor OK Failure
		nCPE	Slot n (CPE)	0	0	Common MPC Diagnostics Processing Error
		nCIE	Slot n (CIE)	0	0	Common Signal Diagnostics Input Signal Error
		nCA	Slot n (CA)	0	0	Common Channels Alarm
		nCD	Slot n (CD)	0	0	Common Channels Danger
		nCSOL	Slot n (CSOL)	0	0	Common Trk/Spd Diagnostics Speed Out Of Limit
		nCTL	Slot n (CTL)	0	0	Common Trk/Spd Diagnostics Track Lost
		nCTOR	Slot n (CTOR)	0	0	Common Trk/Spd Diagnostics Track Out Of Range
		nCDSE	Slot n (CDSE)	0	0	Common MPC Diagnostics DSP Saturation Error
		nCISE	Slot n (CISE)	0	0	Common Signal Diagnostics Input Saturation Error
		nCCME	Slot n (CCME)	0	0	Common Signal Diagnostics Common Mode Overflow
		nSL	Slot n (SL)	0	0	Common MPC Diagnostics Status Latch/Err.
		nDTM	Slot n (DTM)	0	0	Common Controls Direct Trip Multiply
		nDBP	Slot n (DBP)	0	0	Common Controls Danger Bypass
		nAR	Slot n (AR)	0	0	Common Controls Alarm Reset
MPC	Speed channels	nS1A+	Slot n (S1A+)	0	0	Speed Channel 1 Alarm + High
		nS1A-	Slot n (nS1A-)	0	0	Speed Channel 1 Alarm – Low
		nS1ERR	Slot n (nS1ERR)	0	0	Speed Channel 1 Invalid
		nS1SOK	Slot n (nS1SOK)	0	1	Speed Channel 1 Sensor OK check successful (SOK Level)
		nS2A+	Slot n (nS2A+)	0	0	Speed Channel 2 Alarm + High
		nS2A-	Slot n (nS2A-)	0	0	Speed Channel 2 Alarm - Low
		nS2ERR	Slot n (nS2ERR)	0	0	Speed Channel 2 Invalid
		nS2SOK	Slot n (nS2SOK)	0	1	Speed Channel 2 Sensor OK check successful (SOK Level)

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	**MPS backplane configuration extractor tools syntax**	**Logical inv.**	**Normal state**	**Function, as described in the VM600 MPSx software**
MPC	Measurement Channel 1	nV11A+	Slot n (V11A+)	0	0	Measurement Channel 1 Process Output 1 A+
		nV11A-	Slot n (V11A-)	0	0	Measurement Channel 1 Process Output 1 A-
		nV11D+	Slot n (V11D+)	0	0	Measurement Channel 1 Process Output 1 D+
		nV11D-	Slot n (V11D-)	0	0	Measurement Channel 1 Process Output 1 D-
		nV11ERR	Slot n (V11ERR)	0	0	Measurement Channel 1 Process Output 1 Error bit
		nV12A+	Slot n (V12A+)	0	0	Measurement Channel 1 Process Output 2 A+
		nV12A-	Slot n (V12A-)	0	0	Measurement Channel 1 Process Output 2 A-
		nV12D+	Slot n (V12D+)	0	0	Measurement Channel 1 Process Output 2 D+
		nV12D-	Slot n (V12D-)	0	0	Measurement Channel 1 Process Output 2 D-
		nV12ERR	Slot n (V12ERR)	0	0	Measurement Channel 1 Process Output 2 Error bit
		nV1PGA	Slot n (V1PGA)	0	0	Measurement Channel 1 PGA Saturation Error
		nV1SOK	Slot n (V1SOK)	0	1	Measurement Channel 1 Sensor OK check successful
MPC	Measurement Channel 2	nV21A+	Slot n (V21A+)	0	0	Measurement Channel 2 Process Output 1 A+
		nV21A-	Slot n (V21A-)	0	0	Measurement Channel 2 Process Output 1 A-
		nV21D+	Slot n (V21D+)	0	0	Measurement Channel 2 Process Output 1 D+
		nV21D-	Slot n (V21D-)	0	0	Measurement Channel 2 Process Output 1 D-
		nV21ERR	Slot n (V21ERR)	0	0	Measurement Channel 2 Process Output 1 Error bit
		nV22A+	Slot n (V22A+)	0	0	Measurement Channel 2 Process Output 2 A+
		nV22A-	Slot n (V22A-)	0	0	Measurement Channel 2 Process Output 2 A-
		nV22D+	Slot n (V22D+)	0	0	Measurement Channel 2 Process Output 2 D+
		nV22D-	Slot n (V22D-)	0	0	Measurement Channel 2 Process Output 2 D-
		nV22ERR	Slot n (V22ERR)	0	0	Measurement Channel 2 Process Output 2 Error bit
		nV2PGA	Slot n (V2PGA)	0	0	Measurement Channel 2 PGA Saturation Error
		nV2SOK	Slot n (V2SOK)	0	1	Measurement Channel 2 Sensor OK check successful
MPC	Measurement Channel 3	nV31A+	Slot n (V31A+)	0	0	Measurement Channel 3 Process Output 1 A+
		nV31A-	Slot n (V31A-)	0	0	Measurement Channel 3 Process Output 1 A-
		nV31D+	Slot n (V31D+)	0	0	Measurement Channel 3 Process Output 1 D+
		nV31D-	Slot n (V31D-)	0	0	Measurement Channel 3 Process Output 1 D-
		nV31ERR	Slot n (V31ERR)	0	0	Measurement Channel 3 Process Output 1 Error bit
		nV32A+	Slot n (V32A+)	0	0	Measurement Channel 3 Process Output 2 A+
		nV32A-	Slot n (V32A-)	0	0	Measurement Channel 3 Process Output 2 A-
		nV32D+	Slot n (V32D+)	0	0	Measurement Channel 3 Process Output 2 D+
		nV32D-	Slot n (V32D-)	0	0	Measurement Channel 3 Process Output 2 D-
		nV32ERR	Slot n (V32ERR)	0	0	Measurement Channel 3 Process Output 2 Error bit
		nV3PGA	Slot n (V3PGA)	0	0	Measurement Channel 3 PGA Saturation Error
		nV3SOK	Slot n (V3SOK)	0	1	Measurement Channel 3 Sensor OK check successful

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	**MPS backplane configuration extractor tools syntax**	**Logical inv.**	**Normal state**	**Function, as described in the VM600 MPSx software**
MPC	Measurement Channel 4	nV41A+	Slot n (V41A+)	0	0	Measurement Channel 4 Process Output 1 A+
		nV41A-	Slot n (V41A-)	0	0	Measurement Channel 4 Process Output 1 A-
		nV41D+	Slot n (V41D+)	0	0	Measurement Channel 4 Process Output 1 D+
		nV41D-	Slot n (V41D-)	0	0	Measurement Channel 4 Process Output 1 D-
		nV41ERR	Slot n (V41ERR)	0	0	Measurement Channel 4 Process Output 1 Error bit
		nV42A+	Slot n (V42A+)	0	0	Measurement Channel 4 Process Output 2 A+
		nV42A-	Slot n (V42A-)	0	0	Measurement Channel 4 Process Output 2 A-
		nV42D+	Slot n (V42D+)	0	0	Measurement Channel 4 Process Output 2 D+
		nV42D-	Slot n (V42D-)	0	0	Measurement Channel 4 Process Output 2 D-
		nV42ERR	Slot n (V42ERR)	0	0	Measurement Channel 4 Process Output 2 Error bit
		nV4PGA	Slot n (V4PGA)	0	0	Measurement Channel 4 PGA Saturation Error
		nV4SOK	Slot n (V4SOK)	0	1	Measurement Channel 4 Sensor OK check successful
MPC	Measurement Channel 1 & 2	nD12A+	Slot n (D12A+)	0	0	Measurement Channel 1 & 2 Process Output 1 A+
		nD12A-	Slot n (D12A-)	0	0	Measurement Channel 1 & 2 Process Output 1 A-
		nD12D+	Slot n (D12D+)	0	0	Measurement Channel 1 & 2 Process Output 1 D+
		nD12D-	Slot n (D12D-)	0	0	Measurement Channel 1 & 2 Process Output 1 D-
		nD12ERR	Slot n (D12ERR)	0	0	Measurement Channel 1 & 2 Process Output Error bit
		nD12PGA	Slot n (D12PGA)	0	0	Measurement Channel 1 & 2 PGA Saturation Error
		nD12SOK	Slot n (D12SOK)	0	1	Measurement Channel 1 & 2 Sensor OK check successful
MPC	Measurement Channel 3 & 4	nD34A+	Slot n (D34A+)	0	0	Measurement Channel 3 & 4 Process Output 1 A+
		nD34A-	Slot n (D34A-)	0	0	Measurement Channel 3 & 4 Process Output 1 A-
		nD34D+	Slot n (D34D+)	0	0	Measurement Channel 3 & 4 Process Output 1 D+
		nD34D-	Slot n (D34D-)	0	0	Measurement Channel 3 & 4 Process Output 1 D-
		nD34ERR	Slot n (D34ERR)	0	0	Measurement Channel 3 & 4 Process Output Error bit
		nD34PGA	Slot n (D34PGA)	0	0	Measurement Channel 3 & 4 PGA Saturation Error
		nD34SOK	Slot n (D34SOK)	0	1	Measurement Channel 3 & 4 Sensor OK check successful
MPC	Basic & Advanced Functions	nBF1	Slot n (BF1)	0	0	Combination Basic 1
		nBF2	Slot n (BF2)	0	0	Combination Basic 2
		nBF3	Slot n (BF3)	0	0	Combination Basic 3
		nBF4	Slot n (BF4)	0	0	Combination Basic 4
		nBF5	Slot n (BF5)	0	0	Combination Basic 5
		nBF6	Slot n (BF6)	0	0	Combination Basic 6
		nBF7	Slot n (BF7)	0	0	Combination Basic 7
		nBF8	Slot n (BF8)	0	0	Combination Basic 8
		nAF1	Slot n (AF1)	0	0	Combination Advanced 1
		nAF2	Slot n (AF2)	0	0	Combination Advanced 2
		nAF3	Slot n (AF3)	0	0	Combination Advanced 3
		nAF4	Slot n (AF4)	0	0	Combination Advanced 4

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	MPS backplane configuration extractor tools syntax	Logical inv.	Normal state	Function, as described in the VM600 MPSx software
MPC	Basic & Advanced Functions	RLY1A	-	0	0	RLY1 is in Alarm state (RLY1 LED is red)
		RLY1ERR	-	0	0	RLY1 is in Error state (RLY1 LED is Yellow blinking)
		RLY2A	-	0	0	RLY2 is in Alarm state (RLY2 LED is red)
		RLY2ERR	-	0	0	RLY2 is in Error state (RLY2 LED is Yellow blinking)
		RLY3A	-	0	0	RLY3 is in Alarm state (RLY3 LED is red)
		RLY3ERR	-	0	0	RLY3 is in Error state (RLY3 LED is Yellow blinking)
		RLY4A	-	0	0	RLY4 is in Alarm state (RLY4 LED is red)
		RLY4ERR	-	0	0	RLY4 is in Error state (RLY4 LED is Yellow blinking)
		RLY5A	-	0	0	RLY5 is in Alarm state (RLY5 LED is red)
		RLY5ERR	-	0	0	RLY5 is in Error state (RLY5 LED is Yellow blinking)
		RLY6A	-	0	0	RLY6 is in Alarm state (RLY6 LED is red)
		RLY6ERR	-	0	0	RLY6 is in Error state (RLY6 LED is Yellow blinking)
		RLY7A	-	0	0	RLY7 is in Alarm state (RLY7 LED is red)
		RLY7ERR	-	0	0	RLY7 is in Error state (RLY7 LED is Yellow blinking)
		RLY8A	-	0	0	RLY8 is in Alarm state (RLY8 LED is red)
		RLY8ERR	-	0	0	RLY8 is in Error state (RLY8 LED is Yellow blinking)
		DSI1	-	0	0	Discrete Input #1 (‘0’=Open, ‘1’=Short to GND)
		DSI2	-	0	0	Discrete Input #2 (‘0’=Open, ‘1’=Short to GND)
		DSI3	-	0	0	Discrete Input #3 (‘0’=Open, ‘1’=Short to GND)
		DSI4	-	0	0	Discrete Input #4 (‘0’=Open, ‘1’=Short to GND)
		AR	-	0	0	Alarm Reset (‘0’=Open, ‘1’=Short to GND)
		DB	-	0	0	Danger Bypass (‘0’=Open, ‘1’=Short to GND)
		IRCOK	-	0	0	IRC4 Common OK
AMC	Input Channel 1	nCH1A+	Slot n (CH1 A+)	0	0	Input Channel 1 Alarm+
		nCH1A-	Slot n (CH1 A-)	0	0	Input Channel 1 Alarm-
		nCH1D+	Slot n (CH1 D+)	0	0	Input Channel 1 Danger+
		nCH1D-	Slot n (CH1 D-)	0	0	Input Channel 1 Danger-
		nCH1GOKF	Slot n (CH1 Global OK Fail)	0	0	Input Channel 1 Global Channel OK Fail
AMC	Input Channel 2	nCH2A+	Slot n (CH2 A+)	0	0	Input Channel 2 Alarm+
		nCH2A-	Slot n (CH2 A-)	0	0	Input Channel 2 Alarm-
		nCH2D+	Slot n (CH2 D+)	0	0	Input Channel 2 Danger+
		nCH2D-	Slot n (CH2 D-)	0	0	Input Channel 2 Danger-
		nCH2GOKF	Slot n (CH2 Global OK Fail)	0	0	Input Channel 2 Global Channel OK Fail
AMC	Input Channel 3	nCH3A+	Slot n (CH3 A+)	0	0	Input Channel 3 Alarm+
		nCH3A-	Slot n (CH3 A-)	0	0	Input Channel 3 Alarm-
		nCH3D+	Slot n (CH3 D+)	0	0	Input Channel 3 Danger+
		nCH3D-	Slot n (CH3 D-)	0	0	Input Channel 3 Danger-
		nCH3GOKF	Slot n (CH3 Global OK Fail)	0	0	Input Channel 3 Global Channel OK Fail
AMC	Input Channel 4	nCH4A+	Slot n (CH4 A+)	0	0	Input Channel 4 Alarm+
		nCH4A-	Slot n (CH4 A-)	0	0	Input Channel 4 Alarm-
		nCH4D+	Slot n (CH4 D+)	0	0	Input Channel 4 Danger+
		nCH4D-	Slot n (CH4 D-)	0	0	Input Channel 4 Danger-
		nCH4GOKF	Slot n (CH4 Global OK Fail)	0	0	Input Channel 4 Global Channel OK Fail

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	**MPS backplane configuration extractor tools syntax**	**Logical inv.**	**Normal state**	**Function, as described in the VM600 MPSx software**
AMC	Input Channel 5	nCH5A+	Slot n (CH5 A+)	0	0	Input Channel 5 Alarm+
		nCH5A-	Slot n (CH5 A-)	0	0	Input Channel 5 Alarm-
		nCH5D+	Slot n (CH5 D+)	0	0	Input Channel 5 Danger+
		nCH5D-	Slot n (CH5 D-)	0	0	Input Channel 5 Danger-
		nCH5GOKF	Slot n (CH5 Global OK Fail)	0	0	Input Channel 5 Global Channel OK Fail
AMC	Input Channel 6	nCH6A+	Slot n (CH2 A+)	0	0	Input Channel 2 Alarm+
		nCH6A-	Slot n (CH2 A-)	0	0	Input Channel 2 Alarm-
		nCH6D+	Slot n (CH2 D+)	0	0	Input Channel 2 Danger+
		nCH6D-	Slot n (CH2 D-)	0	0	Input Channel 2 Danger-
		nCH6GOKF	Slot n (CH2 Global OK Fail)	0	0	Input Channel 2 Global Channel OK Fail
AMC	Input Channel 7	nCH7A+	Slot n (CH7 A+)	0	0	Input Channel 7 Alarm+
		nCH7A-	Slot n (CH7 A-)	0	0	Input Channel 7 Alarm-
		nCH7D+	Slot n (CH7 D+)	0	0	Input Channel 7 Danger+
		nCH7D-	Slot n (CH7 D-)	0	0	Input Channel 7 Danger-
		nCH7GOKF	Slot n (CH7 Global OK Fail)	0	0	Input Channel 7 Global Channel OK Fail
AMC	Input Channel 8	nCH8A+	Slot n (CH8 A+)	0	0	Input Channel 8 Alarm+
		nCH8A-	Slot n (CH8 A-)	0	0	Input Channel 8 Alarm-
		nCH8D+	Slot n (CH8 D+)	0	0	Input Channel 8 Danger+
		nCH8D-	Slot n (CH8 D-)	0	0	Input Channel 8 Danger-
		nCH8GOKF	Slot n (CH8 Global OK Fail)	0	0	Input Channel 8 Global Channel OK Fail
AMC	Input Channel 1	nAF1	Slot n (AF1)	0 / 1	0 / 1	Advanced Logic Combination Function 1
		nAF2	Slot n (AF2)	0 / 1	0 / 1	Advanced Logic Combination Function 2
		nAF3	Slot n (AF3)	0 / 1	0 / 1	Advanced Logic Combination Function 3
		nAF4	Slot n (AF4)	0 / 1	0 / 1	Advanced Logic Combination Function 4
		nAF5	Slot n (AF5)	0 / 1	0 / 1	Advanced Logic Combination Function 5
		nAF6	Slot n (AF6)	0 / 1	0 / 1	Advanced Logic Combination Function 6
		nAF7	Slot n (AF7)	0 / 1	0 / 1	Advanced Logic Combination Function 7
		nAF8	Slot n (AF8)	0 / 1	0 / 1	Advanced Logic Combination Function 8
AMC	Common	nCSANR	Slot n (Common Status AMC Not Running)	0	0	Common Status AMC Not Running
		nCSSL	Slot n (Common Status Status Latched)	0	0	Common Status Status latched
		nCSDB	Slot n (Common Status Danger Bypass)	0	0	Common Status Danger Bypass
		nCSAR	Slot n (Common Status Alarm Reset)	0	0	Common Status Alarm Reset
AMC	Input Channel 1	nMCH1A+	Slot n (MCH1 A+)	0	0	Multi Channel 1 Alarm+
		nMCH1A-	Slot n (MCH1 A-)	0	0	Multi Channel 1 Alarm-
		nMCH1D+	Slot n (MCH1 D+)	0	0	Multi Channel 1 Danger+
		nMCH1D-	Slot n (MCH1 D-)	0	0	Multi Channel 1 Danger-
		nMCH1GOKF	Slot n (MCH1 Global OK Fail)	0	0	Multi Channel 1 Global Channel OK Fail

Table E-1: Reserved words (Symbol names)

		Symbol name (n is a slot number from 3 to 14)	**MPS backplane configuration extractor tools syntax**	**Logical inv.**	**Normal state**	**Function, as described in the VM600 MPSx software**
AMC	Input Channel 2	nMCH2A+	Slot n (MCH2 A+)	0	0	Multi Channel 2 Alarm+
		nMCH2A-	Slot n (MCH2 A-)	0	0	Multi Channel 2 Alarm-
		nMCH2D+	Slot n (MCH2 D+)	0	0	Multi Channel 2 Danger+
		nMCH2D-	Slot n (MCH2 D-)	0	0	Multi Channel 2 Danger-
		nMCH2GOKF	Slot n (MCH2 Global OK Fail)	0	0	Multi Channel 2 Global Channel OK Fail
AMC	Input Channel 3	nMCH3A+	Slot n (MCH3 A+)	0	0	Multi Channel 3 Alarm+
		nMCH3A-	Slot n (MCH3 A-)	0	0	Multi Channel 3 Alarm-
		nMCH3D+	Slot n (MCH3 D+)	0	0	Multi Channel 3 Danger+
		nMCH3D-	Slot n (MCH3 D-)	0	0	Multi Channel 3 Danger-
		nMCH3GOKF	Slot n (MCH3 Global OK Fail)	0	0	Multi Channel 3 Global Channel OK Fail
AMC	Input Channel 4	nMCH4A+	Slot n (MCH4 A+)	0	0	Multi Channel 4 Alarm+
		nMCH4A-	Slot n (MCH4 A-)	0	0	Multi Channel 4 Alarm-
		nMCH4D+	Slot n (MCH4 D+)	0	0	Multi Channel 4 Danger+
		nMCH4D-	Slot n (MCH4 D-)	0	0	Multi Channel 4 Danger-
		nMCH4GOKF	Slot n (MCH4 Global OK Fail)	0	0	Multi Channel 4 Global Channel OK Fail
AMC	Basic Logical Combination Function	nBF1	Slot n (BF1)	0 / 1	0 / 1	Basic Logic Combination Function 1
		nBF2	Slot n (BF2)	0 / 1	0 / 1	Basic Logic Combination Function 2
		nBF3	Slot n (BF3)	0 / 1	0 / 1	Basic Logic Combination Function 3
		nBF4	Slot n (BF4)	0 / 1	0 / 1	Basic Logic Combination Function 4
		nBF5	Slot n (BF5)	0 / 1	0 / 1	Basic Logic Combination Function 5
		nBF6	Slot n (BF6)	0 / 1	0 / 1	Basic Logic Combination Function 6
		nBF7	Slot n (BF7)	0 / 1	0 / 1	Basic Logic Combination Function 7
		nBF8	Slot n (BF8)	0 / 1	0 / 1	Basic Logic Combination Function 8
		nBF9	Slot n (BF9)	0 / 1	0 / 1	Basic Logic Combination Function 9
		nBF10	Slot n (BF10)	0 / 1	0 / 1	Basic Logic Combination Function 10
		nBF11	Slot n (BF11)	0 / 1	0 / 1	Basic Logic Combination Function 11
		nBF12	Slot n (BF12)	0 / 1	0 / 1	Basic Logic Combination Function 12
		nBF13	Slot n (BF13)	0 / 1	0 / 1	Basic Logic Combination Function 13
		nBF14	Slot n (BF14)	0 / 1	0 / 1	Basic Logic Combination Function 14
		nBF15	Slot n (BF15)	0 / 1	0 / 1	Basic Logic Combination Function 15
		nBF16	Slot n (BF16)	0 / 1	0 / 1	Basic Logic Combination Function 16