

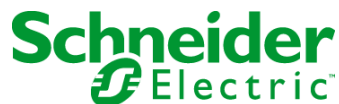
Modicon M580

Hardware Reference Manual

09/2017

EIO0000001578.07

www.schneider-electric.com



The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2017 Schneider Electric. All Rights Reserved.

Table of Contents



	Safety Information	9
	About the Book	13
Part I	Modicon M580 CPUs	17
Chapter 1	M580 CPUs	19
1.1	Functional Characteristics of M580 CPUs	20
	Introduction	21
	Performance Characteristics	23
	States for M580 CPUs	31
	Hot Standby System States	32
	Electrical Characteristics	35
	Real-Time Clock	36
	Addressing Field Buses	39
1.2	BMEP58xxx CPU Physical Characteristics	40
	Physical Description of M580 Standalone CPUs	41
	Physical Description of M580 Hot Standby CPUs	43
	LED Diagnostics for M580 Standalone CPUs	47
	LED Diagnostics for M580 Hot Standby CPUs	50
	USB Port	53
	Ethernet Ports	55
	SD Memory Card	59
	Memory Card Access LED	60
	Data Storage Elementary Functions	62
	Firmware Update	64
	Hardened Equipment	65
Chapter 2	Standards, Certifications, and Conformity Tests	67
	Standards and Certifications	68
	Service Conditions and Recommendations Relating to Environment	70
	Conformity Tests	72
Part II	Installing and Diagnosing Modules on the Local Rack	79
Chapter 3	Installing Modules in an M580 Rack	81
	Module Guidelines	82
	Installing the CPU	83
	Installing an SD Memory Card in a CPU	88

Chapter 4	M580 Diagnostics	91
	Blocking Conditions	92
	Non-blocking Conditions	94
	CPU or System Errors	95
	CPU Application Compatibility	96
Part III	Configuring the CPU in Unity Pro	97
Chapter 5	M580 CPU Configuration	99
5.1	Unity Pro Projects	100
	Creating a Project in Unity Pro	101
	Helping Secure a Project in Unity Pro	103
	Configuring the Size and Location of Inputs and Outputs	105
	Project Management	108
	DIO Scanner Functionality	110
5.2	Configuring the CPU with Unity Pro	112
	Unity Pro Configuration Tabs	113
	About Unity Pro Configuration	114
	Security Tab	115
	IPConfig Tab	119
	RSTP Tab	121
	SNMP Tab	123
	NTP Tab	125
	Switch Tab	128
	QoS Tab	129
	Service Port Tab	130
	Advanced Settings Tab	131
5.3	Configuring the M580 CPU with DTMs in Unity Pro	132
	About DTM Configuration in Unity Pro	133
	Accessing Channel Properties	134
	Configuring DHCP and FDR Address Servers	136
5.4	Diagnostics through the Unity Pro DTM Browser	139
	Introducing Diagnostics in the Unity Pro DTM	140
	Bandwidth Diagnostics	142
	RSTP Diagnostics	144
	Network Time Service Diagnostics	146
	Local Slave / Connection Diagnostics	148
	Local Slave or Connection I/O Value Diagnostics	152
	Logging DTM Events to a Unity Pro Logging Screen	153
	Logging DTM and Module Events to the SYSLOG Server	154

5.5	Online Action	155
	Online Action	156
	EtherNet/IP Objects Tab	158
	Service Port Tab	159
	Pinging a Network Device	160
5.6	Diagnostics Available through Modbus/TCP	162
	Modbus Diagnostic Codes	162
5.7	Diagnostics Available through EtherNet/IP CIP Objects	165
	About CIP Objects	166
	Identity Object	167
	Assembly Object	169
	Connection Manager Object	171
	Modbus Object	173
	Quality Of Service (QoS) Object	175
	TCP/IP Interface Object.	177
	Ethernet Link Object	179
	EtherNet/IP Interface Diagnostics Object	183
	EtherNet/IP IO Scanner Diagnostics Object	186
	IO Connection Diagnostics Object.	188
	EtherNet/IP Explicit Connection Diagnostics Object	192
	EtherNet/IP Explicit Connection Diagnostics List Object.	194
	RSTP Diagnostics Object	196
	Service Port Control Object.	200
5.8	DTM Device Lists	202
	Device List Configuration and Connection Summary	203
	Device List Parameters	206
	Standalone DDT Data Structure for M580 CPUs	211
	Hot Standby DDT Data Structure	219
5.9	Explicit Messaging.	226
	Configuring Explicit Messaging Using DATA_EXCH.	227
	Configuring the DATA_EXCH Management Parameter	229
	Explicit Messaging Services	231
	Configuring EtherNet/IP Explicit Messaging Using DATA_EXCH	233
	EtherNet/IP Explicit Message Example: Get_Attribute_Single	235
	EtherNet/IP Explicit Message Example: Read Modbus Object	238
	EtherNet/IP Explicit Message Example: Write Modbus Object	242
	Modbus TCP Explicit Messaging Function Codes.	246

	Configuring Modbus TCP Explicit Messaging Using DATA_EXCH . . .	247
	Modbus TCP Explicit Message Example: Read Register Request . . .	249
	Sending Explicit Messages to EtherNet/IP Devices	252
	Sending Explicit Messages to Modbus Devices	254
5.10	Explicit Messaging Using the MBP_MSTR Block in Quantum RIO Drops	256
	Configuring Explicit Messaging Using MBP_MSTR	257
	EtherNet/IP Explicit Messaging Services	259
	Configuring the CONTROL and DATABUF Parameters	261
	MBP_MSTR Example: Get_Attributes_Single	263
	Modbus TCP Explicit Messaging Function Codes	268
	Configuring the Control Parameter for Modbus TCP Explicit Messaging	269
5.11	Implicit Messaging	279
	Setting Up Your Network	280
	Adding an STB NIC 2212 Device	281
	Configuring STB NIC 2212 Properties	283
	Configuring EtherNet/IP Connections	285
	Configuring I/O Items	290
	EtherNet/IP Implicit Messaging	303
5.12	Configuring the M580 CPU as an EtherNet/IP Adapter	304
	Introducing the Local Slave	305
	Local Slave Configuration Example	307
	Enabling Local Slaves	308
	Accessing Local Slaves with a Scanner	309
	Local Slave Parameters	312
	Working with Device DDTs	314
5.13	Hardware Catalog	316
	Introduction to the Hardware Catalog	317
	Adding a DTM to the Unity Pro Hardware Catalog	318
	Adding an EDS File to the Hardware Catalog	319
	Removing an EDS File from the Hardware Catalog	322
5.14	M580 CPU Embedded Web Pages	324
	Introducing the Standalone Embedded Web Pages	325
	Status Summary (Standalone CPUs)	326
	Performance	328
	Port Statistics	329
	I/O Scanner	331
	Messaging	333

	QoS	334
	NTP	336
	Redundancy	338
	Alarm Viewer	339
	Rack Viewer	340
5.15	M580 Hot Standby CPU Web Pages	343
	Introducing the M580 Hot Standby CPU Web Pages	344
	Status Summary (Hot Standby CPUs)	346
	HSBY Status	348
	Rack Viewer	351
Chapter 6	M580 CPU Programming and Operating Modes	355
6.1	I/O and Task Management	356
	I/O Exchanges	357
	CPU Tasks	359
6.2	BMEP58xxx CPU Memory Structure	361
	Memory Structure	361
6.3	BMEP58xxx CPU Operating Modes	362
	Managing Run/Stop Input	363
	Power Cut and Restore	364
	Cold Start	366
	Warm Restart	369
Appendices	371
Appendix A	Function Blocks	373
	ETH_PORT_CTRL: Executing a Security Command in an Application	373
Glossary	377
Index	385

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book



At a Glance

Document Scope

PlantStruxure is a Schneider Electric program designed to address the key challenges of many different types of users, including plant managers, operations managers, engineers, maintenance teams, and operators, by delivering a system that is scalable, flexible, integrated, and collaborative.

This document provides detailed information about the M580 programmable automation controller (PAC). These topics are also discussed:

- Install a local rack in the M580 system.
- Configure the M580 CPU.
- The CPU performs Ethernet I/O scanning of both RIO and DIO logic without affecting network determinism.

Validity Note

This document is valid for Unity Pro 13.0 or later and BMEP58.... firmware version 2.10 or later.

The technical characteristics of the devices described in this document also appear online. To access this information online:

Step	Action
1	Go to the Schneider Electric home page www.schneider-electric.com .
2	In the Search box type the reference of a product or the name of a product range. <ul style="list-style-type: none">• Do not include blank spaces in the reference or product range.• To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the Product Datasheets search results and click on the reference that interests you. If you entered the name of a product range, go to the Product Ranges search results and click on the product range that interests you.
4	If more than one reference appears in the Products search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click Download XXX product datasheet .

The characteristics that are presented in this manual should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the manual and online information, use the online information as your reference.

Related Documents


Title of Documentation	Reference Number
Control Panel Technical Guide How to protect a machine from malfunctions due to electromagnetic disturbance	CPTG003_EN (English), CPTG003_FR (French)
Grounding and Electromagnetic Compatibility of PLC Systems (Basic Principles and Measures) User Manual	33002439 (English), 33002440 (French), 33002441 (German), 33003702 (Italian), 33002442 (Spanish), 33003703 (Chinese)
<i>Modicon M580 Standalone System Planning Guide for Frequently Used Architectures</i>	HRB62666 (English), HRB65318 (French), HRB65319 (German), HRB65320 (Italian), HRB65321 (Spanish), HRB65322 (Chinese)
<i>Modicon M580 System Planning Guide for Complex Topologies</i>	NHA58892 (English), NHA58893 (French), NHA58894 (German), NHA58895 (Italian), NHA58896 (Spanish), NHA58897 (Chinese)
Modicon M580 Hot Standby Installation and Configuration Guide	NHA58880 (English), NHA58881 (French), NHA58882 (German), NHA58883 (Italian), NHA58884 (Spanish), NHA58885 (Chinese)
Modicon M580 BMENOC0301/11 Ethernet Communication Module Installation and Configuration Guide	HRB62665 (English), HRB65311 (French), HRB65313 (German), HRB65314 (Italian), HRB65315 (Spanish), HRB65316 (Chinese)
Modicon M580 Remote I/O Modules Installation and Configuration Guide	EIO0000001584 (English), EIO0000001585 (French), EIO0000001586 (German), EIO0000001588 (Italian), EIO0000001587 (Spanish), EIO0000001589 (Chinese)

Title of Documentation	Reference Number
<i>Modicon M580 BMENOS0300 Network Option Switch Module Installation and Configuration Guide</i>	NHA89117 (English), NHA89119 (French), NHA89120 (German), NHA89121 (Italian), NHA89122 (Spanish), NHA89123 (Chinese)
Modicon eX80 BME AHI 0812 HART Analog Input Module & BME AHO 0412 HART Analog Output Module User Guide	EAV16400 (English), EAV28404 (French), EAV28384 (German), EAV28413 (Italian), EAV28360 (Spanish), EAV28417 (Chinese)
Unity Loader User Manual	33003805 (English), 33003806 (French), 33003807 (German), 33003809 (Italian), 33003808 (Spanish), 33003810 (Chinese)
Unity Pro Operating Modes	33003101 (English), 33003102 (French), 33003103 (German), 33003696 (Italian), 33003104 (Spanish), 33003697 (Chinese)
Unity Pro, Program Languages and Structure, Reference Manual	35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)
Modicon X80 Racks and Power Supplies, Hardware, Reference Manual	EIO0000002626 (English), EIO0000002627 (French), EIO0000002628 (German), EIO0000002630 (Italian), EIO0000002629 (Spanish), EIO0000002631 (Chinese)

Title of Documentation	Reference Number
Modicon Controllers Platform Cyber Security, Reference Manual	EIO0000001999 (English), EIO0000002001 (French), EIO0000002000 (German), EIO0000002002 (Italian), EIO0000002003 (Spanish), EIO0000002004 (Chinese)

You can download these technical publications and other technical information from our website at <http://www.schneider-electric.com/en/download>

Product Related Information

 WARNING
<p>UNINTENDED EQUIPMENT OPERATION</p> <p>The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise are allowed to program, install, alter, and apply this product.</p> <p>Follow all local and national safety codes and standards.</p> <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p>

Part I

Modicon M580 CPUs

Introduction

This part provides information about the Modicon M580 CPUs, including physical and operational characteristics.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	M580 CPUs	19
2	Standards, Certifications, and Conformity Tests	67

Chapter 1

M580 CPUs

Introduction

This chapter introduces you to the physical and functional characteristics of the M580 CPUs.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	Functional Characteristics of M580 CPUs	20
1.2	BMEP58xxxx CPU Physical Characteristics	40

Section 1.1

Functional Characteristics of M580 CPUs

Introduction

This section describes the functional characteristics of the M580 CPUs. Performance, electrical characteristics, and memory capacities of the different CPU modules are detailed.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction	21
Performance Characteristics	23
States for M580 CPUs	31
Hot Standby System States	32
Electrical Characteristics	35
Real-Time Clock	36
Addressing Field Buses	39

Introduction

Role of the CPU in a Control System

In a modular PAC, the CPU controls and processes the application. The local rack identifies the rack that contains the CPU. In addition to the CPU, the local rack contains a power supply module and may contain communication processing modules and input/output (I/O) modules.

The CPU is in charge of:

- configuring all modules and device present in the PAC configuration
- processing the application
- reading the inputs at the beginning of tasks and applying the outputs at the end of tasks
- managing explicit and implicit communications

Modules may reside in the local rack with the CPU or they may be installed in remote drops at a distance from the local rack. The CPU has built-in capabilities to act as the RIO processor that manages communications between the CPU and the Quantum and X80 EIO adapter modules that are installed in each remote drop.

Devices can be connected to the PAC network as either DIO clouds or DIO sub-rings.

For detailed information about the various architectures that the M580 network supports, refer to the *Modicon M580 System Planning Guide* (see *Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*). For a detailed description of the X80 EIO adapter modules and the options they provide for installing a remote drop, refer to the *Modicon M580, Remote I/O Modules Installation and Configuration Guide* (see *Modicon M580, RIO Modules, Installation and Configuration Guide*).

Functional Considerations

The CPU solves control logic for the I/O modules and distributed equipment in the system. Choose a CPU based on several operating characteristics:

- memory size
- processing power: the number of I/O points or channels that it can manage ([see page 24](#))
- the speed at which the CPU can execute the control logic ([see page 30](#))
- communication capabilities: the types of Ethernet ports on the CPU ([see page 55](#))
- the number of local I/O modules and RIO drops that it can support ([see page 24](#))
- the ability to function in harsh environments: (3 CPU modules are hardened to operate over extended temperature ranges and in dirty or corrosive environments ([see page 65](#)))
- network configuration (standalone or Hot Standby)

Standalone CPU Modules

This is a list of the available CPU modules. Some are available in both standard and industrially hardened modules. Industrially hardened modules have the letter H appended to the module name (*see page 65*). The letter C at the end of the module name indicates a conformal coating for harsh environments:

- BMEP581020, BMEP581020H
- BMEP582020, BMEP582020H
- BMEP582040, BMEP582040H
- BMEP583020
- BMEP583040
- BMEP584020, BMEP584020C
- BMEP584040, BMEP584040C
- BMEP585040, BMEP585040C
- BMEP586040, BMEP586040C

Hot Standby CPU Modules

These CPU modules are compatible with M580 Hot Standby systems:

- BMEH582040, BMEH582040C
- BMEH584040, BMEH584040C
- BMEH586040, BMEH586040C

NOTE: For detailed information about M580 Hot Standby configurations, refer to the *Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures* (*see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).

Performance Characteristics

Introduction

All M580 CPUs have an embedded DIO scanner service to manage distributed equipment on the M580 device network. Some M580 CPUs also have an embedded RIO scanner service to manage RIO drops.

To manage RIO drops on the device network, select one of these CPUs with Ethernet I/O scanner service (both RIO and DIO scanner service):

- BMEP582040, BMEP582040H
- BMEP583040
- BMEP584040
- BMEP585040, BMEP585040C
- BMEP586040, BMEP586040C
- BMEH582040
- BMEH584040, BMEH584040C
- BMEH586040, BMEH586040C

Embedded Ethernet I/O scanner services are configured via CPU IP configuration ([see page 119](#)).

NOTE: Some of this information applies to M580 Hot Standby configurations. For more information, refer to the *Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures* (see *Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).

CPU Characteristics

These tables show the key characteristics of the M580 standalone and Hot Standby CPUs. These characteristics represent the maximum values that a specific CPU can manage in the M580 system.

NOTE: The values in these tables may not be achieved depending on the I/O density and the number of available rack slots.

Standalone CPUs:

Maximum number of ...	Reference (BMEP58 ...)								
	1020(H)	2020(H)	2040(H)	3020	3040	4020	4040	5040(C)	6040(C)
discrete I/O channels	1024	2048	2048	3072	3072	4096	4096	5120	6144
analog I/O channels	256	512	512	768	768	1024	1024	1280	1536
expert channels	36	72	72	108	108	144	144	180	216
distributed devices ⁴	64	128	64	128	64	128	64	64	64
memory size In+Out (KB)	2+2	4+4	2+2	4+4	2+2	4+4	2+2	2+2	2+2
Ethernet communication modules (including BMENOC0301/11 modules, but not the CPU)	2	2	2	3	3	4 ⁽⁵⁾	4 ⁽⁵⁾	6 ⁽¹⁾⁽⁵⁾	6 ⁽¹⁾⁽⁵⁾
local racks (main rack + extended rack)	4	4	4	8	8	8	8	8	8
RIO drops (see page 25) (maximum of 2 racks per drop) (main rack + extended rack)	–	–	8 ²	–	16 ²	–	16 ³	31 ³	31 ³
Ethernet ports:									
• service	1	1	1	1	1	1	1	1	1
• RIO or distributed equipment	–	–	2	–	2	–	2	2	2
• distributed equipment	2	2	–	2	–	2	–	–	–
– (not available)									
H (hardened)									
C (coated version)									
1. Only four of these six modules can be BMENOC03•1 modules.									
2. Supports BM•CRA312•0 adapter modules.									
3. Supports BM•CRA312•0 and 140CRA31200 adapter modules.									
4. Of these connections: 3 are reserved for local slaves; the remainder are available for scanning distributed equipment.									
5. A maximum of 3 BME Ethernet modules, all the other are BMX Ethernet modules.									

Hot Standby CPUs:

Maximum number of ...	Reference (BMEH58 ...)		
	2040	4040(C)	6040(C)
distributed devices	64	64	64
memory In+Out (KB)	2+2	2+2	2+2
Ethernet communication modules (including BMENOC0301/11 modules, but not the CPU)	2	4	6 ⁽¹⁾
local racks (main rack + extended rack)	1	1	1
RIO drops (<i>see page 25</i>) (maximum of 2 racks per drop) (main rack + extended rack)	8 ²	16 ³	31 ³
Ethernet ports:			
• service	1	1	1
• RIO or distributed equipment	2	2	2
• distributed equipment	0	0	0
1. Only four of these six communication modules can be BMENOC0301/11 modules. 2. Supports BM•CRA312•0 adapter modules. 3. Supports BM•CRA312•0 and 140CRA31200 adapter modules.			

RIO Drop Maximum Configuration

The maximum number of channels in an RIO drop depends on the eX80 EIO adapter module:

EIO adapter	Maximum number of Channels			
	Discrete	Analog	Expert	Sensor bus
BMXCRA31200	128	16	–	–
BMXCRA31210	1024	256	36	2
BMECRA31210	1024	256	36	2

NOTE: The number of available channels could differ from the maximum values shown because the values depend on the CPU reference and the other modules in the same drop. More information is given in Modicon X80 I/O Modules (*see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).

To configure Quantum RIO drops, refer to the Quantum EIO installation and configuration guide (*see Quantum EIO, Remote I/O Modules, Installation and Configuration Guide*).

Maximum Internal Memory Size

Program and Data Memory (Standalone). This table shows the program and data memory capacity for M580 standalone CPUs:

Memory Size	Reference (BMEP58 ...)								
	1020(H)	2020(H)	2040(H)	3020	3040	4020(C)	4040(C)	5040(C)	6040(C)
internal memory size (KB)	4598	9048	9048	13558	13558	18678	18678	29174	65535 ⁽¹⁾
1. The sum of saved data, unsaved data, and program data is limited to 65535 KB.									

Program and Data Memory (Hot Standby). This table shows the program and data memory capacity for M580 Hot Standby CPUs:

Memory Size	Reference (BMEH58 ...)		
	2040	4040(C)	6040(C)
internal memory size (KB)	9462	18934	65536 ⁽¹⁾
1. The sum of saved data, unsaved data, and program data is limited to 65536 KB.			

Memory Areas (Standalone). This table shows the maximum memory size per area for M580 standalone CPUs:

Maximum Memory Size	Reference (BMEP58 ...)								
	1020(H)	2020(H)	2040(H)	3020	3040	4020(C)	4040(C)	5040(C)	6040(C)
saved data (KB) ⁽¹⁾	384	768	768	1024	1024	2048	2048	4096	4096
program (KB)	4096	8162	8162	12288	12288	16384	16384	24576	65536 ⁽²⁾
1. 10 KB are reserved for the system									
2. The sum of saved data, unsaved data, and program data is limited to 65536 KB.									

Memory Areas (Hot Standby). This table shows the maximum memory size per area for M580 Hot Standby CPUs:

Maximum Memory Size	Reference (BMEH58 ...)		
	2040	4040(C)	6040(C)
saved data (KB) ⁽¹⁾	768	2048	4096
Hot Standby data exchanged (KB)	768	2048	4096
program (KB)	4096	16384	65536 ⁽²⁾
1. 10 KB are reserved for the system			
2. The sum of saved data, unsaved data, and program data is limited to 65536 KB.			

Located Data (Standalone). This table shows the maximum and default size of located data (in KB) for each M580 standalone CPU:

Object Types	Address	Reference (BMEP58 ...)								
		1020(H)	2020(H)	2040(H)	3020	3040	4020(C)	4040(C)	5040(C)	6040(C)
internal bits	%Mi maximum	32634	32634	32634	32634	32634	32634	65280 ⁽²⁾	65280 ⁽²⁾	65280 ⁽²⁾
	%Mi default	512	512	512	512	512	512	512	512	512
input/output bits	%Ir.m.c %Qr.m.c	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
system bits	%Si	128	128	128	128	128	128	128	128	128
internal words	%MWi maximum	32464	32464	32464	65232	65232	65232	64968 ⁽³⁾	64968 ⁽³⁾	64968 ⁽³⁾
	%MWi default	1024	1024	1024	2048	2048	2048	2048	2048	2048
1 Memory size depends on the equipment configuration declared (I/O modules). 2 32634 for versions before 2.30. 3 65232 for versions before 2.30.										

Located Data (Hot Standby). This table shows the maximum and default size of located data (in KB) for each M580 Hot Standby CPU:

Object Types	Address	Reference (BMEH58 ...)		
		2040	4040(C)	6040(C)
internal bits	%Mi maximum	32634	65280 ⁽²⁾	65280 ⁽²⁾
	%Mi default	512	512	512
input/output bits	%Ir.m.c %Qr.m.c	(1)	(1)	(1)
system bits	%Si	128	128	128
internal words	%MWi maximum	32464	64968 ⁽³⁾	64968 ⁽³⁾
	%MWi default	1024	1024	2048
1 Memory size depends on the equipment configuration declared (I/O modules). 2 32634 for versions before 2.30. 3 65232 for versions before 2.30.				

Size of Non-Located Data Memory

This list contains non-located data types:

- elementary data type (EDT)
- derived data type (DDT)
- derived function block (DFB) and elementary function block (EFB)

The size limit of non-located data is the global maximum memory size for data (*see page 26*) minus the size consumed by located data.

Client and Server Requests per Scan

The communication performance of standalone (BMEP58•0•0) and Hot Standby (BMEH58•0•0) CPUs is described in terms of the number of client and server requests per scan.

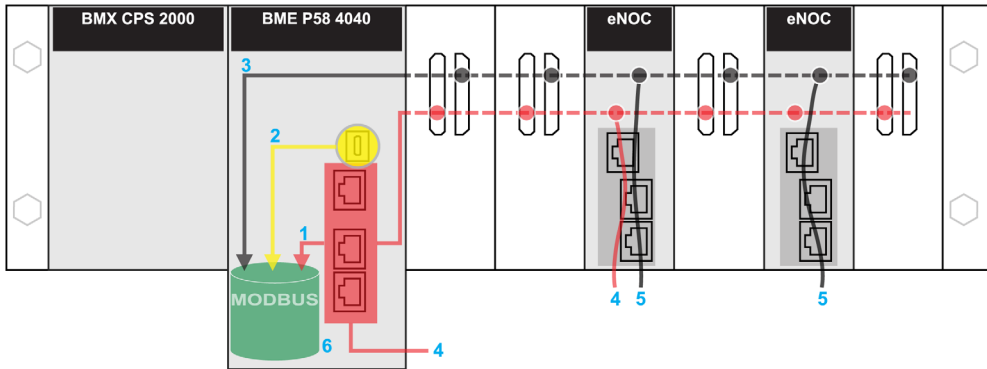
The table below shows the maximum number of Modbus TCP, EtherNet/IP, or UMAS requests that can be served by the CPU's Modbus TCP server at each MAST scan.

When the incoming requests exceed these maximums, they are queued in a first-in/first out (FIFO) buffer. The size of the FIFO buffer is according to the selected CPU:

CPU	Overall Maximum		From USB	Maximum requests sent to IP address of the CPU	Maximum requests sent to IP address of comm. modules
	Requests per Scan ⁽¹⁾	Request FIFO Size			
BMXP581020	8 (16)	32	4	8	16
BMX•5820•0	16 (24)	32	4	12	16
BMXP5830•0	24 (32)	32	4	16	16
BMX•5840•0	32 (40)	50	4	24	16
BMEP5850•0	40 (48)	50	4	32	16
BME•5860•0	56 (64) ⁽²⁾	50	4	32	16
1. This column shows the default limits for the number of requests served per cycle. The limit can be modified through %SW90, between 2 and the number indicated between brackets. 2. The overall limit for the BME•5860•0 CPU is higher than the sum of the limits for the USB, CPU, and NOC modules. This is a provision for future evolutions.					

The MAST task cycle time may increase by up to 0.5 ms per incoming request. When the communications load is high, you can limit the potential jitter of the MAST time by limiting the number of requests that are processed per cycle in %SW90.

Example: This example local rack assembly includes a BMEP584040 CPU and two BMENOC0301/11 Ethernet communication modules. Therefore, the maximum values in this example apply to the BMEP584040 CPU (described above):



red: These requests are sent to the IP address of the CPU.

yellow: These requests are from the USB port of the CPU.

gray: These requests are sent to the IP address of a communications module (NOC).

- 1 The maximum number of requests to the IP address of the BMEP584040 CPU (24).
- 2 The maximum number of requests from the USB port of the CPU (4). (For example, a PC that runs Unity Pro may be connected to the USB port.)
- 3 The maximum number of requests from all communications modules on the local rack (16).
- 4 These requests are sent to the IP address of the BMEP584040 CPU from devices that are connected to an Ethernet port on either the CPU or a BMENOC0301/11 module.
- 5 These requests are sent to the IP address of the BMENOC0301/11 from devices that are connected on the Ethernet port of either the BMENOC0301/11 or the CPU. (In this case, enable the Ethernet backplane port of the BMENOC0301/11.)
- 6 The Modbus server can manage in each request the maximum number of requests from the BMEP584040 CPU (32). It also holds a maximum of 50 requests in a FIFO buffer.

Number of Connections: This table shows the maximum number of simultaneous Modbus TCP, EtherNet/IP, and UMAS connections for the embedded Ethernet port on these CPUs:

CPU	Connections
BMXP581020	32
BMX•5820•0	32
BMXP5830•0	48
BMX•5840•0	64
BMEP5850•0	64
BME•5860•0	80

When an incoming connection request is accepted, the open connection that has been idle for the longest time is closed.

Modbus TCP and EtherNet/IP Client: This table shows the maximum number (per cycle) of communication EFs that support Modbus TCP and EtherNet/IP clients according to the selected CPU:

CPU	EFs per Cycle
BMEP581020	16
BME•5820•0	32
BMEP5830•0	48
BME•5840•0	80
BMEP5850•0	80
BME•5860•0	96

Application Code Execution Performance

This table shows the performance of the application code for each M580 standalone (BMEP58 ...) and Hot Standby (BMEH58...) CPU:

	Reference BMEH58 .../BMEH58 ...								
	1020(H)	2020(H)	2040(H)	3020	3040	4020(C)	4040(C)	5040(C)	6040(C)
boolean application execution (Kinst/ms ⁽¹⁾)	10	10	10	20	20	40	40	50	50
typical execution (Kinst/ms ^(1.))	7.5	7.5	7.5	15	15	30	30	40	40
1. <ul style="list-style-type: none"> ● Kinst/ms: 1,024 instructions per millisecond ● A typical execution holds 65% boolean instructions + 35% fixed arithmetic. 									

States for M580 CPUs

Introduction

This topic describes the operating states for M580 standalone and Hot Standby CPUs.

Operating States for Standalone CPUs

All standalone M580 CPUs have these operating states:

Operating State	Description
AUTOTEST	The CPU is executing its internal self-tests. NOTE: If extended racks are connected to the main local rack and line terminators are not plugged into the unused connectors on the rack extender module, the CPU remains in AUTOTEST after the self-tests have completed.
NOCONF	The application program is not valid.
STOP	The CPU has a valid application, but it is stopped. The CPU sets itself to predefined STOP state parameters, and can be restarted when you are ready.
HALT	The CPU has an application, but it has stopped operating because it encountered an unexpected blocking condition, which puts the CPU in a HALT state, resulting in a recoverable (see page 94) or nonrecoverable condition (see page 92).
RUN	The CPU is executing the application program.
WAIT	The CPU is in a transitory state while it backs up data when a power down condition is detected. The CPU starts again only when power is restored and the supply reserve is replenished. As it is a transitory state, it may not be viewed. The CPU performs a warm restart (see page 369) to exit the WAIT state.
ERROR	The CPU is stopped because a hardware or system error is detected. When the system is ready to be restarted, the CPU performs a cold start (see page 367) to exit the ERROR state.
OS DOWNLOAD	A CPU firmware download is in progress.

Monitoring the CPU Operating State

The LEDs on the CPU front panel provide indications of its operating state ([see page 47](#)).

Hot Standby System States

PAC State Versus Hot Standby System State

The state of the Hot Standby system depends on the operating state of the PAC. These Hot Standby states are supported:

PAC operating state	Hot Standby system state
INIT	INIT
STOP	Stop
RUN	Primary with standby counterpart
	Primary without standby counterpart
	Standby
	Wait

This list describes the Hot Standby states:

- **Primary:** The PAC controls all system processes and devices:
 - It executes program logic.
 - It receives input from, and controls output to, distributed equipment and RIO drops.
 - If connected to a PAC in standby state, the primary PAC checks the status of, and exchanges data with, the standby PAC.

In a Hot Standby network, both PACs can be primary if both the Hot Standby and Ethernet RIO links are not functioning. When either of these two links is restored, the PAC does one of the following:

 - Remains in the primary state.
 - Transitions to the standby state.
 - Transitions to the wait state.
- **Standby:** The standby PAC maintains a state of readiness. It can take control of system processes and devices if the primary PAC cannot continue to perform these functions:
 - It reads the data and the I/O states from the primary PAC.
 - It does not scan distributed equipment, but receives this information from the primary PAC.
 - It executes program logic. You can configure the standby PAC to execute:
 - The first section of program logic (the default setting); or
 - Specified sections of program logic, including all MAST and FAST task sections.

NOTE: You can specify if a section is to be executed in the **Condition** tab the **Properties** dialog for each section.

 - On each scan, it checks the status of the primary PAC.

- **Wait:** The PAC is in RUN mode, but cannot act as either primary or standby. The PAC transitions from the wait state to either the primary or standby state, when all preconditions for that state exist, including:
 - The state of the Hot Standby link.
 - The state of the Ethernet RIO link.
 - The presence of at least one connection with an Ethernet RIO drop.
 - The position of the A/B rotary selection switch on the rear of the CPU.
 - The state of the configuration. For example:
 - If a firmware mismatch exists, the `FW_MISMATCH_ALLOWED` flag is set.
 - If a logic mismatch exists, the `LOGIC_MISMATCH_ALLOWED` flag is set.

In the wait state, the PAC continues to communicate with other modules on the local rack, and can execute program logic, if configured to do so. You can configure a PAC in wait state to execute:

- Specific sections of program logic, specified in the **Condition** tab the **Properties** dialog for each section.
- The first section of program logic.
- No program logic.
- **INIT:** Both the PAC and the Hot Standby system are initializing.
- **Stop:** The PAC is in STOP mode. On the STOP to RUN transition, the PAC may move to the wait, standby, or primary state. This transition depends on the state of the Ethernet RIO and Hot Standby links, and on the position of the A/B rotary selection switch on the rear of the CPU.

NOTE: In addition to the PAC operating states listed here, other operating states that are not related to the Hot Standby system ([see page 31](#)) exist.

PAC Functions by Hot Standby System State

A PAC performs these functions, depending on its Hot Standby state:

PAC functions	Hot Standby system states		
	Primary	Standby	Wait
RIO drops	YES	NO	NO
Distributed equipment	YES	NO	NO
Execution of program logic in MAST and FAST tasks	YES	Depending on configuration, standby PAC can execute: <ul style="list-style-type: none"> • First section (default) • Specified sections (which can include all MAST and FAST task sections) 	Depending on configuration, wait PAC can execute: <ul style="list-style-type: none"> • First section (default) • Specified sections (which can include all MAST and FAST task sections)
1. Data exchange is controlled by the Exchange on STBY attribute.			

PAC functions	Hot Standby system states		
	Primary	Standby	Wait
Application data exchange ¹ between primary and standby CPU	YES	YES	NO
Status data exchange between primary and standby CPU	YES	YES	YES
1. Data exchange is controlled by the Exchange on STBY attribute.			

Electrical Characteristics

Introduction

The power supply module provides current to the modules installed on the local rack, including the CPU. The CPU current consumption contributes to the total rack consumption.

CPU Power Consumption

Typical CPU consumption with a 24 Vdc power supply:

CPU Module	Typical Consumption
BMEP581020(H)	270 mA
BMEP5820•0(H)	270 mA
BMEP5830•0	295 mA
BMEP5840•0(C)	295 mA
BMEP585040(C)	300 mA
BMEP586040(C)	300 mA
BMEH582040(H)	335 mA (with a copper SFP)
BMEH584040(C)	360 mA (with a copper SFP)
BMEH586040(C)	365 mA (with a copper SFP)

Mean Time Between Failures (MTBF)

For all CPU modules, the MTBF (measured at 30 °C continuous) is 600,000 hours.

Real-Time Clock

Introduction

Your CPU has a real-time clock that:

- provides the current date and time
- displays the date and time of the last application shut-down

Clock Accuracy

The resolution of the real-time clock is 1 ms. The clock accuracy is affected by the operating temperature of the application:

Operating Temperature	Maximum Daily Drift (Seconds/Day)	Maximum Yearly Drift (Minutes/Year)
25 °C (77 °F) stabilized	+/- 2.6	+/- 17.4
0...60 °C (32...140 °F)	+/- 5.2	+/-33.1

Clock Back-Up

The accuracy of the real-time clock is maintained for 4 weeks when the CPU power is turned off if the temperature is below 45 °C (113 °F). If the temperature is higher, the back-up time is shorter. The real-time clock back-up does not need any maintenance.

If the back-up power is too low, system bit %S51 is set to 1. This value indicates a loss of time when the power supply was OFF.

Current Date and Time

The CPU updates the current date and time in the system words %SW49–%SW53 and %SW70. This data is in BCD.

NOTE: For M580 PACs, the current time is in universal coordinated time (UTC). If local time is needed, use the `RRTC_DT` function.

Accessing the Date and Time

You can access the date and time:

- on the CPU debug screen
- in the program
- from the DTM diagnostics screen

To read the current date and time, read system words %SW49 through %SW53. This operation sets system bit %S50 to 0.

To write the current date and time, write system words %SW50 through %SW53. This operation sets system bit %S50 to 1.

When system bit %S59 is set to 1, you can increment or decrement the current date and time values with system word %SW59.

The function performed by each bit in word %SW59 is:

Bit	Function
0	increments the day of the week
1	increments the seconds
2	increments the minutes
3	increments the hours
4	increments the days
5	increments the months
6	increments the years
7	increments the centuries
8	decrements the day of the week
9	decrements the seconds
10	decrements the minutes
11	decrements the hours
12	decrements the days
13	decrements the months
14	decrements the years
15	decrements the centuries

NOTE: The preceeding functions are performed when system bit %S59 is set to 1.

Determining the Date and Time of the Last Application Shutdown

The local date and time of the last application shutdown are displayed in system words %SW54 through %SW58. They are displayed in BCD.

System Word	Most Significant Byte	Least Significant Byte
%SW54	seconds (0 to 59)	00
%SW55	hours (0 to 23)	minutes (0 to 59)
%SW56	month (1 to 12)	day in the month (1 to 31)
%SW57	century (0 to 99)	year (0 to 99)
%SW58	day of the week (1 to 7)	reason for the last application shutdown

The reason for the last application shutdown can be displayed by reading the least significant byte of system word %SW58, which can have these values (in BCD):

Word%SW58 Value	Definition
1	application switched to STOP mode
2	application stopped by watchdog
4	power loss
5	stop on detected hardware error
6	stop when errors such as these are detected: <ul style="list-style-type: none">● software error (HALT instruction)● SFC error● application CRC checksum error● undefined system function call Details on the software detected fault type are stored in %SW125.

Addressing Field Buses

Addressing Field Buses

The following field buses can be addressed by either configuring the appropriate protocol or using dedicated modules and devices.

Field Bus	Addressing Method
AS-i	AS-Interface bus is addressed with a Modicon X80 BMXEIA0100 module.
HART	HART communication protocol can be addressed using either the eX80 HART modules: <ul style="list-style-type: none"> ● BMEAHI0812 HART analog input module ● BMEAHO0412 HART analog output module or <ul style="list-style-type: none"> ● a Modicon STB island with an STBNIP2311 EtherNet/IP network interface module and an STBAHI8321 HART interface module.
Modbus TCP, EtherNet/IP	Modbus TCP devices are connected to the Ethernet DIO network.
Modbus Plus	Modbus Plus is supported using a gateway module like TCSEGDB23F24FA or TCSEGDB23F24FK.
PROFIBUS-DP	A PROFIBUS remote master is connected to the Ethernet DIO network. The process variables are exchanged via the DIO scanner service in the CPU. PROFIBUS gateway modules: TCSEGA23F14F or TCSEGA23F14FK
PROFIBUS-PA	A PROFIBUS remote master and a DP/PA interface are connected to an Ethernet DIO network. The process variables are exchanged via the DIO scanner service in the CPU. PROFIBUS gateway modules: TCSEGA23F14F or TCSEGA23F14FK

Section 1.2

BMEP58xxxx CPU Physical Characteristics

Introduction

This section describes the physical elements that are displayed on the front panel of the M580 CPUs. The various communication ports, LED diagnostic information, and several options available for industrial hardening and memory back-up are detailed.

What Is in This Section?

This section contains the following topics:

Topic	Page
Physical Description of M580 Standalone CPUs	41
Physical Description of M580 Hot Standby CPUs	43
LED Diagnostics for M580 Standalone CPUs	47
LED Diagnostics for M580 Hot Standby CPUs	50
USB Port	53
Ethernet Ports	55
SD Memory Card	59
Memory Card Access LED	60
Data Storage Elementary Functions	62
Firmware Update	64
Hardened Equipment	65

Physical Description of M580 Standalone CPUs

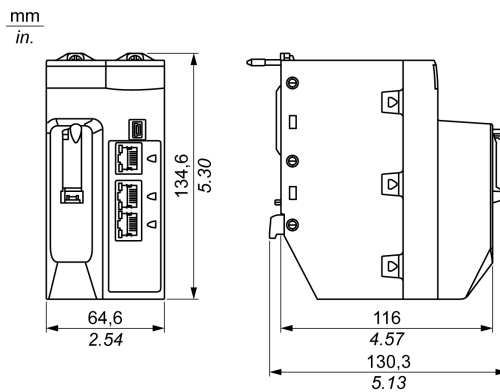
Position on the Local Rack

Every M580 standalone system requires one CPU module. The CPU is installed in the two-module slot position directly to the right of the power supply in the main local rack. The CPU cannot be put in any other slot location or any other rack. If there are extended racks in the local rack configuration, assign address 00 to the rack with the CPU.

NOTE: Refer to the list of M580 standalone CPU modules ([see page 22](#)).

Dimensions

This graphic shows the front and side dimensions of the M580 standalone CPUs:



NOTE:

Consider the height of the CPU when you are planning the installation of the local rack. The CPU extends below the lower edge of the rack by:

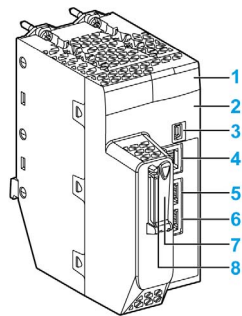
- 29.49 mm (1.161 in.) for an Ethernet rack
- 30.9 mm (1.217 in.) for an X Bus rack

Front Panel


M580 standalone CPUs have similar front panels. Depending on the standalone CPU you choose, these differences apply:

- BMEP58•020: The embedded Ethernet I/O scanner service supports DIO only.
- BMEP58•040: The embedded Ethernet I/O scanner service supports both RIO and DIO.

Physical features:



Legend:

Item	Marking	Description
1	–	LED display (see page 47) for CPU status and diagnostics
2	Eth MAC Address xx.xx.xx.xx.xx	media access control (MAC) address assigned to the CPU, which is a string of six 2-digit hexadecimal numbers separated by dots
	IP ADDRESS: ...	blank space for you to write the IP address assigned to the CPU NOTE: The default IP address starts with 10.10 and uses the last 2 bytes of the MAC address.
3		mini-B USB connector (see page 53) to which you can attach a Unity Pro program, a loader terminal, or an HMI
4	Service	RJ45 Ethernet connector (see page 55) for the service port
5	Device Network	<ul style="list-style-type: none"> • BMEP58•020: dual RJ45 Ethernet connectors (see page 55) that support distributed equipment only • BMEP58•040: dual RJ45 Ethernet connectors (see page 55) that support distributed equipment and RIO drops
6		
7	—	SD memory card (see page 59) slot
8	—	This green LED indicates the status of the memory card: <ul style="list-style-type: none"> • ON: The CPU can access the SD memory card. • blinking: The CPU does not recognize the SD memory card. • flashing: The CPU attempts to access the SD memory card.

Physical Description of M580 Hot Standby CPUs

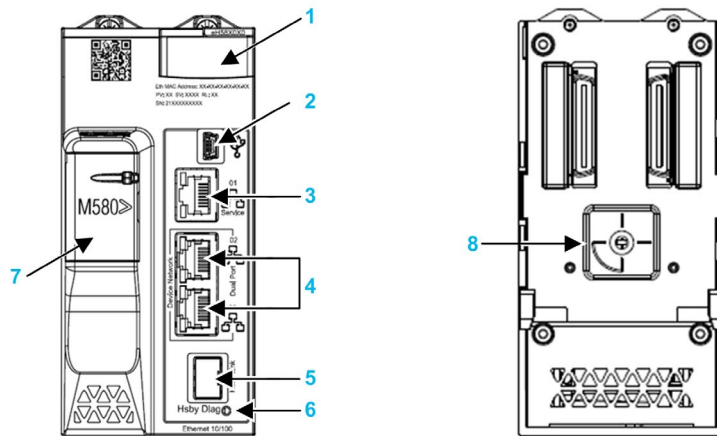
PAC Hot Standby CPU Modules

These M580 CPU modules support M580 Hot Standby systems:

- BMEH582040
- BMEH584040 and BMEH584040C
- BMEH586040 and BMEH586040C

CPU Module Front and Back Views

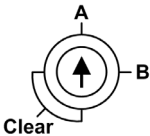
The three Hot Standby CPU modules have the same external hardware features. The front of the module is on the left. The back of the module is on the right:



- 1 LED diagnostic display panel
- 2 Mini-B USB port for module configuration via PC running Unity Pro
- 3 RJ45 Ethernet service port connector
- 4 RJ45 connectors that together serve as a dual port to the Ethernet network
- 5 SFP socket for copper or fiber-optic Hot Standby link connection
- 6 Hot Standby status link LED
- 7 SD memory card slot
- 8 A/B/Clear rotary selector switch, used to designate the PAC as either PAC A or PAC B, or to clear the existing Unity Pro application

Rotary Selector Switch

Use the rotary switch on the back of each M580 Hot Standby CPU to designate the role that the CPU plays in the M580 Hot Standby configuration:



Use the screwdriver provided with the CPU to set the rotary switch according to its role in a Hot Standby system:

Position	Result
A	<ul style="list-style-type: none">Designates the PAC as PAC A (see <i>Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures</i>), as referenced in Unity Pro and the T_M_ECPU_HSBY (see page 220) DDDT.Assigns the PAC IP address A on Ethernet RIO network.
B	<ul style="list-style-type: none">Designates the PAC as PAC B (see <i>Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures</i>), as referenced in Unity Pro and the T_M_ECPU_HSBY DDDT.Assigns the PAC IP address B on Ethernet RIO network.
Clear	<ul style="list-style-type: none">Clears the application in the PAC, and places the PAC into the NO_CONF operational state.If an SD memory card is inserted in the PAC, the application in the card is also cleared, <p>NOTE: Setting the switch for each Hot Standby PAC to the same A/B position can cause a conflict of PAC roles (see <i>Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures</i>).</p>

Clearing CPU Memory

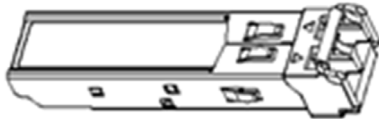
To clear a CPU memory, follow these steps:

Step	Action
1	Set the rotary switch to [Clear] .
2	Power up the PAC.
3	Power down the PAC.
4	Set the rotary switch to [A] or [B] .

When you next power up the PAC, if the remote PAC is primary, the primary PAC transfers the application to the local PAC.

SFP Socket

Each CPU module includes one SFP socket, to which you can connect either a fiber optic or a copper transceiver:



To insert a transceiver:

Step	Action
1	Check that the CPU is powered off.
2	Position the transceiver so that its label is oriented to the left.
3	Press the SFP transceiver firmly into the socket until you feel it snap into place. NOTE: If the SFP transceiver resists, check the orientation of the transceiver and repeat these steps.

To remove a transceiver:

Step	Action
1	Check that the CPU is powered off.
2	Pull out the latch to unlock the transceiver.
3	Pull on the transceiver to remove it.

NOTICE

POTENTIAL EQUIPMENT DAMAGE

Do not Hot Swap the SFP transceiver. Insert or remove the transceiver only when there is no power to the CPU.

Failure to follow these instructions can result in equipment damage.

NOTE: For part numbers and other information regarding the available transceivers, refer to the description of CPU Hot Standby link transceivers (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*).

Each module comes with a stopper. When the SFP socket is not connected to a transceiver, cover the unused socket with the cover to keep out dust



Grounding Considerations

DANGER

ELECTRICAL SHOCK HAZARD

- Switch off the power supply at both ends of the PAC connection, and lock out and tag out both power sources before you insert or remove an Ethernet cable.
- In case lock out and tag out are not available, ensure that the power sources cannot be inadvertently switched on.
- Use suitable insulation equipment when you insert or remove an Ethernet cable.

Failure to follow these instructions will result in death or serious injury.

Do not apply power to a Modicon X80 rack until connections are made at both ends of the Ethernet cable. For example, make these connections before you turn on the power:

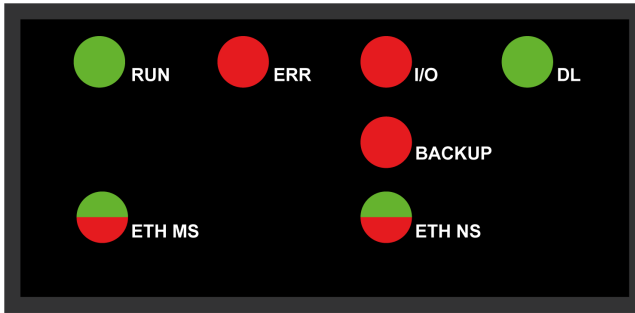
- Connect the Ethernet cable to the BMECRA31210 performance EIO adapter module and another device (adapter module) or dual-ring switch (DRS). (Refer to the *Modicon M580 System Planning Guide for Complex Topologies* (see *Modicon M580, System Planning Guide for, Complex Topologies*) for details about DRSs.)
- Connect the copper Ethernet cable to both SFP transceivers when you use 490NAC0100 copper transceivers.

Use fiber-optic cable to establish a communications link when it is not possible to master the potential between the two grounds.

LED Diagnostics for M580 Standalone CPUs

LED Display







A 7-LED display is located on the front panel of the CPU:



LED Descriptions

LED Indicator	Description
RUN	ON: The CPU is in RUN state.
ERR	ON: The CPU or system has detected an error.
I/O	ON: The CPU or system has detected an error in one or more I/O modules.
DL (download)	<ul style="list-style-type: none"> ● Blinking: Firmware update in progress. ● OFF: No firmware update in progress.
BACKUP	<p>ON:</p> <ul style="list-style-type: none"> ● The memory card or CPU flash memory is missing or inoperable. ● The memory card is not usable (bad format, unrecognized type). ● The memory card or CPU flash memory content is inconsistent with the current application. ● The memory card has been removed and reinserted. ● A PLC → Project Backup... → Backup Clear command has been performed when no memory card is present. The BACKUP LED remains ON until the project is successfully backed up. <p>OFF: The memory card or CPU flash memory content is valid, and the application in the execution memory is identical.</p>
ETH MS	<p>MOD STATUS (green/red): Pattern indicates the Ethernet port configuration status.</p> <p>NOTE: With the detection of a recoverable error, the ETH MS LED can be green or red and on or off.</p>
ETH NS	NET STATUS (green/red): Pattern indicates the Ethernet connection status.

This table describes the LED indicator patterns:

Symbol	Description	Symbol	Description
	off		steady red
	steady green		blinking red
	blinking green		blinking red/green

LED Diagnostic Indications

NOTE: In a Hot Standby system, specific IP addresses (Main IP Address, Main IP Address + 1, IP Address A, IP Address B) are assigned (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*) and these addresses must not be used by other devices in the system.

NOTICE

UNINTENDED EQUIPMENT BEHAVIOR

Confirm that each module has a unique IP address. Duplicate IP addresses can cause unpredictable module/network behavior.

Do not assign an IP address equal to the Main IP Address, the Main IP Address + 1, IP Address A, or IP Address B to any Ethernet device that potentially communicates with the Hot Standby system. A duplicate IP address condition, causing unintended equipment operation, can occur.

Failure to follow these instructions can result in equipment damage.

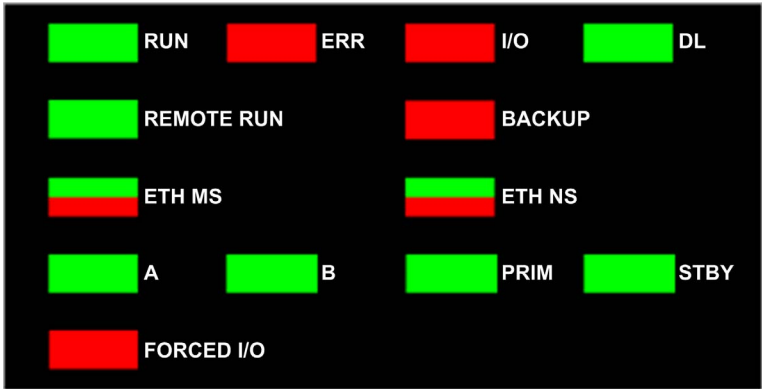
The LEDs provide detailed diagnostic information when you observe their pattern in combination:

Condition	CPU State	RUN	ERR	I/O	ETH MS	ETH NS
power on	Autotest					
not configured (before getting a valid IP address or configuration is invalid)	NOCONF					–
configured	Stop			<ul style="list-style-type: none"> • off: no error detected • steady red: error detected in a module or a channel 		<ul style="list-style-type: none"> • off: invalid IP address • blinking green: valid IP address but no EtherNet/IP connection • steady green: EtherNet/IP connection established
	RUN					
recoverable detected error	HALT			–		<ul style="list-style-type: none"> • blinking red: At least one exclusive owner CIP connection (for which the BMENOC0301/11 is the originator) is timed out. The LED blinks until the connection is reestablished or the module is reset.
duplicate IP address	–	–	–	–		
unrecoverable detected error	–					–
power off	–					
–: any pattern						

LED Diagnostics for M580 Hot Standby CPUs

LED Panel

The front face of a BMEH58•040 Hot Standby CPU presents the following LED panel, which you can use to diagnose the state of the M580 Hot Standby system:



Hot Standby Panel LEDs

The BMEH58•040 Hot Standby CPU LEDs present the following Hot Standby system diagnostics:

LED	Description
A	<ul style="list-style-type: none">● ON (green) indicates:<ul style="list-style-type: none">○ The local CPU A/B/Clear rotary switch (<i>see page 44</i>) is set to "A" , and○ The remote CPU A/B/Clear rotary switch is set to "B".● BLINKING (green) indicates:<ul style="list-style-type: none">○ If LED B is OFF:<ul style="list-style-type: none">- The local CPU A/B/Clear rotary switch is set to "A" , and- The remote CPU A/B/Clear rotary switch is also set to "A".○ If LED B is also BLINKING green:<ul style="list-style-type: none">- The local CPU A/B/Clear rotary switch is set to "Clear".● OFF: Indicates local CPU A/B/Clear rotary switch is not set to "A" or to "Clear".

LED	Description
B	<ul style="list-style-type: none"> ● ON (green) indicates: <ul style="list-style-type: none"> ○ The local CPU A/B/Clear rotary switch is set to “B” , and ○ The remote CPU A/B/Clear rotary switch is set to “A”. ● BLINKING (green) indicates: <ul style="list-style-type: none"> ○ If LED A is OFF: <ul style="list-style-type: none"> - The local CPU A/B/Clear rotary switch is set to “B” , and - The remote CPU A/B/Clear rotary switch is also set to “B”. ○ If LED A is also BLINKING green: <ul style="list-style-type: none"> - The local CPU A/B/Clear rotary switch is set to “Clear.” ● OFF: Indicates local CPU A/B/Clear rotary switch is not set to “B” or “Clear”.
REMOTE RUN	<p>Indicates the RUN status of the remote PAC:</p> <ul style="list-style-type: none"> ● ON: (green): The remote PAC is in RUN state. ● BLINKING: (green): The remote PAC is in STOP state. ● OFF: The local PAC cannot read the state of the remote PAC. Both the Hot Standby link and the Ethernet RIO link are lost.
PRIM	<p>Indicates the primary status of the PAC:</p> <ul style="list-style-type: none"> ● ON (green) The local PAC is primary, but the remote PAC is not in standby state. ● BLINKING: The local PAC is in wait state; the STBY LED is also BLINKING. ● OFF: The local PAC is not primary. <p>NOTE:</p> <ul style="list-style-type: none"> ● If CPU is in RUN mode and both PRIM and STBY LEDs are OFF, CPU is in wait state. ● If both CPUs are in RUN mode, and one CPU is primary and the other CPU is in wait state: <ul style="list-style-type: none"> - On Primary: PRIM LED is ON, STBY LED is BLINK. - On Wait: PRIM LED is OFF, STBY LED is BLINK
STBY	<p>Indicates the standby status of the PAC:</p> <ul style="list-style-type: none"> ● ON (green): Indicates the PAC is in standby state. ● BLINKING (green) indicates either: <ul style="list-style-type: none"> ○ The local PAC is primary, but the remote PAC is not in standby state. ○ The local PAC is in wait state; the PRIM LED is also BLINKING. ● OFF: Indicates local PAC is not in standby state. <p>NOTE:</p> <ul style="list-style-type: none"> ● If CPU is in RUN mode and both PRIM and STBY LEDs are BLINKING, the CPU is in wait state. ● If one CPU is primary and the other CPU is in wait state: <ul style="list-style-type: none"> - On Primary: PRIM LED is ON, STBY LED is BLINKING. - On Wait: PRIM LED is OFF, STBY LED is BLINKING.

Hot Standby Link LED

A Hot Standby link LED is located on the front of the CPU, just below and to the right of the SFP socket for the Hot Standby link connector. Use this LED to diagnose the state of the Hot Standby link:

Status	Color	Description
ON	green	The port is communicating with the remote PAC.
BLINKING	green	The port is configured and operational, but a Hot Standby link is not made.
OFF	—	The Hot Standby link is not configured or is not operational.

Non-Hot Standby Panel LEDs

Refer to the topic LED Indications ([see page 47](#)) to use the other LEDs (non-Hot Standby) to diagnose the CPU.

USB Port

Introduction

The USB port is a high-speed, mini-B USB connector, version 2.0 (480 Mbps) that can be used for a Unity Pro program or human-machine interface (HMI) panel. The USB port can connect to another USB port, version 1.1 or later.

NOTE: Install M580 USB drivers before connecting the USB cable between the CPU and the PC.

Transparency

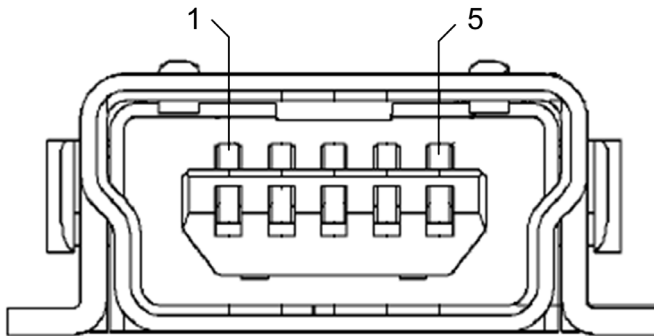
If your system requires transparency between the device connected to the USB port and the M580 device network, add a persistent static route in the device's routing table.

Example of a command to address a device network with IP address $x.x.0.0$ (for a Windows PC): `route add x.x.0.0 mask 255.255.0.0 90.0.0.1 -p`

(In this case, $x.x.0.0$ is the network address used by the M580 device network, and 255.255.0.0 is the corresponding subnet mask.)

Pin Assignments

The USB port has the following pin positions and pinouts:



Legend:

Pin	Description
1	VBus
2	D-
3	D+
4	not connected
5	ground
shell	chassis ground

Cables

Use a BMX XCA USB 018 cable (1.8 m/5.91 ft) to connect the panel to the CPU. (This cable has a type A connector on one side and the mini-B USB on the other side.)

In a fixed assembly with an XBT-type console connected to the CPU, connect the USB cable to a protection bar. Use the exposed part of the shield or the metal lug on the BMX XCA cable to make the connection.

Ethernet Ports

Introduction

There are three RJ45 Ethernet ports on the front of the CPU: one service port, and two device network ports. The ports share the characteristics described below.

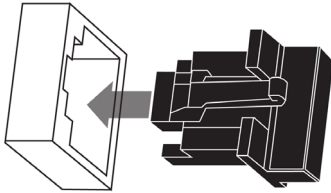
Common Characteristics

All three ports have the same RJ45 connector and all use the same type of Ethernet cables.

NOTE: The three Ethernet ports are connected to chassis ground, and the system requires an equipotential ground.

Dust Cover

To keep dust from entering the unused Ethernet ports, cover the unused ports with the stopper:

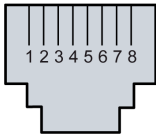


Ethernet Ports

Each RJ45 connector has a pair of LED indicators:



The pin positions, pinouts, and cable connections are the same on all three RJ45 Ethernet ports:

Pin	Description	<div>Pinout:</div> 
1	TD+	
2	TD-	
3	RD+	
4	not connected	
5	not connected	
6	RD-	
7	not connected	
8	not connected	
—	shell/chassis ground	

NOTE: The TD pins (pins 1 and 2) and the RD pins (pins 3 and 6) can be reversed to allow the exclusive use of straight-through cables.

The ports have an auto MDIX capability that automatically detects the direction of the transmission.

Choose from these Ethernet cables to connect to the Ethernet ports:

- TCSECN3M3M••••: Cat 5E Ethernet straight-through shielded cable, rated for industrial use, CE- or UL-compliant
- TCSECE3M3M••••: Cat 5E Ethernet straight-through shielded cable, rated for industrial use, CE-compliant
- TCSECU3M3M••••: Cat 5E Ethernet straight-through shielded cable, rated for industrial use, UL-compliant

The maximum length for a copper cable is 100 m. For distances greater than 100 m, use fiber optic cable. The CPU does not have any fiber ports on it. You may use dual ring switches (DRSs) or BMX NRP •••• fiber converter modules (*see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*) to handle the copper-fiber conversion.

Ethernet Ports on Standalone CPUs

On standalone CPUs, the **ACTIVE** LED is green. The **LNK** LED is either green or yellow, depending on the status:

LED	LED Status	Description
ACTIVE	OFF	No activity is indicated on the Ethernet connection.
	ON / blinking	Data is being transmitted and received on the Ethernet connection.
LNK	OFF	No link is established at this connection.
	ON green	A 100 Mbps link* is established at this connection.
	ON yellow	A 10 Mbps link* is established at this connection.
* The 10/100 Mbps links support both half-duplex and full-duplex data transfer and autonegotiation.		

Hot Standby Link LED

A Hot Standby link LED is located on the front of the CPU, just below and to the right of the SFP socket for the Hot Standby link connector. Use this LED to diagnose the state of the Hot Standby link:

Status	Color	Description
ON	green	The port is communicating with the remote PAC.
BLINKING	green	The port is configured and operational, but a Hot Standby link is not made.
OFF	—	The Hot Standby link is not configured or is not operational.

Service Port

The service port is the uppermost of the three Ethernet ports on the front panel of the CPU. This port can be used:

- to provide an access point that other devices or systems can use to monitor or communicate with the M580 CPU
- as a standalone DIO port that can support a star, daisy chain, or mesh topology of distributed equipment
- to mirror the CPU ports for Ethernet diagnostics. The service tool that views activity on the mirrored port may be a PC or an HMI device.

NOTE: Do not connect the service port to the device network, either directly or through a switch/hub. Doing so may affect system performance.

NOTE: The service port may not provide full performance and features that the **Device Network** ports on the CPU provide.

CAUTION

RISK OF UNINTENDED OPERATION

Do not connect together the service ports of the Hot Standby CPUs. Connecting together the service ports of the primary and standby CPUs can cause unintended system operation.

Failure to follow these instructions can result in injury or equipment damage.

Device Network Dual Ports

When a CPU does not support RIO scanning, the two ports below the service port marked **Device Network** are DIO ports.

These CPUs do not support RIO scanning:

- BMEP581020 and BMEP581020 H
- BMEP582020 and BMEP582020 H
- BMEP583020
- BMEP584020

You may use a **Device Network** port to support a star, daisy chain, or mesh topology of distributed equipment. You may use both **Device Network** ports to support a ring topology.

For details about distributed equipment architectures, refer to the *Modicon M580 Standalone System Planning Guide for Frequently Used Architectures*.

When a CPU supports RIO scanning, the two ports below the service port marked **Device Network** are RIO ports. These CPUs support RIO scanning:

- BMEP582040 and BMEP582040H
- BMEP583040
- BMEP584040
- BMEP585040
- BMEP586040
- BMEH582040
- BMEH584040
- BMEH586040

When used as RIO ports, both ports connect the CPU to the main ring in an Ethernet daisy-chain loop or ring.

For more information about RIO architectures, refer to the *Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures* (see *Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).

SD Memory Card

BMXRMS004GPF SD Memory Card

The SD memory card is an option that can be used for application and data storage. The SD memory card slot in the M580 CPU housing is behind a door.

Use a BMXRMS004GPF memory card in your CPU. It is a 4 GB, Class A card rated for industrial use. Other memory cards, including those used in M340 CPUs, are not compatible with M580 CPUs.

NOTE:

If you insert an incompatible SD memory card in the CPU:

- The CPU remains in NOCONF state ([see page 31](#)).
- The CPU **BACKUP** LED turns ON.
- The memory card access LED remains blinking.

NOTE: The BMXRMS004GPF memory card is formatted specifically for the M580 CPUs. If you use this card with another CPU or tool, the card may not be recognized.

Memory Card Characteristics

These memory card characteristics apply to M580 CPUs:

Characteristic	Value
global memory size	4 GB
application backup size	200 MB
data storage size	3.8 GB
write/erase cycles (typical)	100,000
operating temperature range	–40...+85 °C (–40...+185 °F)
file retention time	10 years
memory zone for FTP access	data storage directory only

NOTE: Due to formatting, wearout, and other internal mechanisms, the actual available capacity of the memory card is slightly lower than its global size.

Formatting the Memory Card

The formatting procedure is described in *Formatting the Memory Card* topic in the *Unity Pro System Block Library* (see *Unity Pro, System, Block Library*).

Memory Card Access LED

Introduction

The green memory card access LED underneath the SD memory card door indicates the CPU access to the memory card when a card is inserted. This LED can be seen when the door is open.

Dedicated LED States









By itself, the **memory card access** LEDs indicate these states:









LED Status	Description
ON	The memory card is recognized, but the CPU is not accessing it.
flashing	The CPU is accessing the memory card.
blinking	The memory card is not recognized.
OFF	The memory card can be removed from the CPU slot or the CPU does not recognize the memory card.

NOTE: Confirm that the LED is OFF before you remove the card from the slot.





Combined LED Meanings

The access card LED operates together with the **BACKUP** LED (*see page 47*). Their combined patterns indicate the following diagnostic information:

Memory Card Status	Conditions	CPU State	Memory Card Access LED	BACKUP LED
no memory card in the slot	—	no configuration		
memory card not OK	—	no configuration		
memory card without project	—	no configuration		
memory card with a non-compatible project	—	no configuration		
— no specific circumstances or CPU state				

Memory Card Status	Conditions	CPU State	Memory Card Access LED	BACKUP LED
memory card with a compatible project	An error is detected when the project is restored from the memory card to the CPU RAM.	no configuration	during transfer: 	during transfer: 
			end of transfer: 	end of transfer: 
	No error is detected when the project is restored from the memory card to the CPU RAM.	—	during transfer: 	during transfer: 
			end of transfer: 	end of transfer: 
— no specific circumstances or CPU state				

This legend shows the different LED patterns:

Symbol	Meaning	Symbol	Meaning
	off		steady red
	steady green		blinking green

Data Storage Elementary Functions

Data Storage Elementary Functions

These DataStorage_EF elementary functions are supported in Unity Pro for the M580 CPUs:

EF	CPU		Description
	BMEP58-0-0	BMEH58-040	
CLOSE_FILE	X	X	The CLOSE_FILE function closes the file identified by the file descriptor attribute. If another user is working on the same file via a different descriptor, the file remains open.
CREATE_FILE (see Unity Pro, System, Block Library)	X	—	The CREATE_FILE function creates a new file, assigns it the specified file name, and indicates the purposes for which the file is opened: read-only, write-only, read-write.
DELETE_FILE (see Unity Pro, System, Block Library)	X	—	The DELETE_FILE function deletes the specified file.
GET_FILE_INFO (see Unity Pro, System, Block Library)	X	X	The GET_FILE_INFO function retrieves information about a specified target file. Execute the OPEN_FILE function for the target file before executing the GET_FILE_INFO function, because the identity of the target file comes from the output parameter of the OPEN_FILE block.
GET_FREESIZE (see Unity Pro, System, Block Library)	X	X	The GET_FREESIZE function displays the amount of available space on the SD memory card.
OPEN_FILE (see Unity Pro, System, Block Library)	X	X (read only)	The OPEN_FILE function opens a specified existing file.
RD_FILE_TO_DATA (see Unity Pro, System, Block Library)	X	X	The RD_FILE_TO_DATA function enables reading data from a file, at the current position in the file, and copies the data to a direct address variable, a located variable, or an unlocated variable.
SEEK_FILE (see Unity Pro, System, Block Library)	X	X	The SEEK_FILE function sets the current byte offset in the file to a new specified offset position, which can be: the offset, the current position plus the offset, the file size plus the offset.
X (supported) — (not supported)			

EF	CPU		Description
	BMEP58•0•0	BMEH58•040	
SET_FILE_ATTRIBUTES (see <i>Unity Pro, System, Block Library</i>)	X	—	The SET_FILE_ATTRIBUTES function sets the read-only status of a file attribute. Read-only status can be set or cleared. This function can be applies only to a file that is already open via the CREATE_FILE or OPEN_FILE function.
WR_DATA_TO_FILE (see <i>Unity Pro, System, Block Library</i>)	X	—	The WR_DATA_TO_FILE function enables the writing of the value of a direct address variable, a located variable, or an unlocated variable to a file. The value is written to the current position in the file. After the write, the current position in the file is updated.
X (supported) — (not supported)			

For additional information on each function, refer to the chapter *Implementing File Management* (see *Unity Pro, System, Block Library*) in the *Unity Pro System Block Library*.

Firmware Update

Introduction

You can update the CPU firmware by downloading a new firmware version with Unity Loader.

Download the firmware through a connection to one of these:

- CPU mini-B USB connector (*see page 53*)
- CPU **Service** port (*see page 57*)
- Ethernet network

NOTE:

- For a description of the download procedure, refer to the *Unity Loader, a SoCollaborative software User Manual*.
- When using an M580 Hot Standby configuration, refer to the *Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures* (*see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).

Enabling CPU Firmware Update

To enable the firmware update, check the CPU security settings (*see page 115*).

Firmware File

Select the firmware file (**.dx*) that is compatible with your CPU.

Update Procedure

Update the CPU and BMEXBP••00 rack firmware:

Step	Action
1	Install Unity Loader software.
2	Connect the PC that is running Unity Loader to the CPU.
3	Launch Unity Loader.
4	Click Firmware tab.
5	In the PC list box, select the <i>.dx</i> file that contains the firmware file.
6	When connected with Ethernet, check that the MAC address indicated in the PLC box corresponds to the MAC address marked on the CPU.
7	Check that transfer sign is green to allow transfer from PC to CPU.
8	Click Transfer .
9	Click Close .

Hardened Equipment

Introduction

Hardened equipment is the ruggedized version of standard equipment that can operate in extended temperature ranges and in dirty or corrosive environments. There are hardened versions of several M580 CPU's.

Extended Temperature Considerations


This table shows the temperature ranges for different equipment standards:

Classification	Temperature Range
standard	0 ... 60 °C (32...140 °F)
coated	-25 ... 60 °C (-13...140 °F)
hardened	-25 ... 70 °C (-13...158 °F)

When used in the standard temperature range, hardened equipment has the same performance characteristics as the standard equipment. However, at the higher and lower ends of the extended temperature range (lower than 0 °C or higher than 60 °C), the hardened power supplies can have reduced power ratings that affect power calculations.

If hardened equipment is operated above or below the extended temperature limits (lower than - 25 °C (-13 °F) or higher than 70 °C (158 °F)), the equipment can operate abnormally.

NOTE: Refer to the list of hardened and coated M580 CPU modules ([see page 22](#)).

 **WARNING**

UNINTENDED EQUIPMENT OPERATION
Do not operate M580 equipment outside of its specified temperature range.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Operating in Harsh Environments

Hardened equipment has a conformal coating applied to its electronic boards. When associated with appropriate installation and maintenance, this treatment allows it to be more robust in harsh chemical environments.

Conformal coating increases the isolation capability of the circuit boards and their resistance to:

- condensation
- dusty atmospheres (conducting foreign particles)
- chemical corrosion, in sulphurous atmospheres (for example, in oil refineries or purification plants) or in atmospheres that contain halogens such as chlorine.

Chapter 2

Standards, Certifications, and Conformity Tests

Introduction

This chapter describes the operational standards for modules in an M580 PAC system. Agency certifications, environmental conditions, and mechanical characteristics of the modules are detailed.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Standards and Certifications	68
Service Conditions and Recommendations Relating to Environment	70
Conformity Tests	72

Standards and Certifications

Introduction

M580 PACs have been designed to comply with the relevant standards and rules for electrical equipment in an industrial automation environment.

NOTE: The M580 PAC standard and certifications are consistent with the Modicon X80 and M340 ranges.

Industrial Standards

Requirements specific to the PAC functional characteristics, immunity, robustness, and safety:

- IEC/EN 61131-2 completed by IEC/EN 61010-2-201
- CSA C22.2 No.142 completed by CSA C22.2 No. 61010-2-201
- UL 508 completed by UL 61010-2-201

M580 PACs are intended for use in industrial environments, as well as the following:

- pollution degree 2, over-voltage category II (IEC 60664-1)
- low-voltage installations, in which the main power branch is protected on both wires by devices as fuses or breakers that limit the current to 15A for North America and 16A for the rest of the world

Merchant Navy Certification

The products are designed to comply with major merchant navy agencies requirements (IACS).

More details on merchant navy certifications are available on Schneider Electric website: www.schneider-electric.com.

European Directives for EC Marking

- low voltage: 2006/95/EG and 2014/35/UE from April 2016
- electromagnetic compatibility: 2004/108/EC and 2014/30/UE from April 2016

Installation in Classified Ex Area

- For USA and Canada: Hazardous locations class I, division 2, groups A, B, C, and D according to CSA 22.2 No.213, or ISA12.12.01, or FM3611
- For other countries: EC ATEX (directive 94/9/EC and 2014/34/UE from April 2014), or IECEx in defined atmosphere zone 2 (gas) and/or zone 22 (dust) according to IEC/EN 60079-0, IEC/EN 60079-15, and IEC/EN 60079-31

More details on certifications and Ex installation guides are available on Schneider Electric website: www.schneider-electric.com.

Specific Countries

- For Australia and New Zealand: ACMA requirements for RCM marking
- For Russia and Eurasian Customs Union: EAC

Environmentally Friendly Design

- Hazardous substances: This product is compliant with:
 - WEEE, Directive 2012/19/EU
 - RoHS, Directive 2011/65/EU
 - RoHS China, Standard SJ/T 11363-2006
 - REACH regulation EC 1907/2006

NOTE: Documentation about sustainable development is available on Schneider Electric website (Product Environmental Profile and End of Life Instructions, RoHS and REACH certificates).

- End of life (WEEE): This product contains electronic boards. Therefore, dispose of this product in specific treatment channels.

Service Conditions and Recommendations Relating to Environment

Operating Temperature/Hygrometry/Altitude

Condition		Standard M580 Components	Hardened M580 Components
Temperature	Operation	0...+60 °C (+32...+140 °F)	–25...+70 °C (–13...+158 °F)
	Storage	–40...+85 °C (–40...+185 °F)	–40...+85 °C (–40...+185 °F)
Relative humidity (without condensation)	Cyclical humidity	5...95% up to +55 °C (+131 °F)	5...95% up to +55 °C (+131 °F)
	Continuous humidity	5...93% up to +55 °C (+131 °F)	5...93% up to +60 °C (+140 °F)
Altitude	Operation	<ul style="list-style-type: none"> 0...2000 m (0...6562 ft): full specification for temperature and isolation 2000...4000 m (6562...13123 ft): <ul style="list-style-type: none"> Temperature derating: +1 °C/400 m (+1.8 °F/1312 ft) Isolation loss: 150 Vdc/1000 m (150 Vdc/3280 ft) 	

Supply Voltage

Operating conditions relative to the supply voltage:

Power Supply		BMXCPS References					
		2000	2010	3020 (H)	3500 (H)	3540 T	4002 (H)
Voltage	Rated	100...240 Vac	24 Vdc	24...48 Vdc	100...240 Vac	125 Vdc	100...240 Vac
	Limit	85...264 Vac	18...31.2 Vdc	18...62.4 Vdc	85...264 Vac	100...150 Vdc	85...264 Vac
Input Power	Maximum	70 VA	–	–	120 VA	45 W	130 VA
Current Consumption	Nominal	0.61 A at 115 Vac 0.31 A at 240 Vac	1 A at 24 Vdc	1.65 A at 24 Vdc 0.83 A at 48 Vdc	1.04 A at 115 Vac 0.52 A at 240 Vac	0.36 A at 125 Vdc	1.1 A at 115 Vac 0.55 A at 230 Vac
Frequency	Rated	50...60 Hz	–	–	50...60 Hz	–	50...60 Hz
	Limit	47...63 Hz	–	–	47...63 Hz	–	47...63 Hz
Micro-power outages	Duration	≤ 1/2 period	≤ 10 ms ⁽¹⁾	≤ 10 ms ⁽¹⁾⁽²⁾	≤ 1/2 period	≤ 50 ms at 125 Vdc	≤ 1/2 period
	Repetition	≥ 1 s	≥ 1 s	≥ 1 s	≥ 1 s	≥ 1 s	≥ 1 s
1. Limited to 1 ms at maximum load with minimum supply (18 Vdc).							
2. ≤10 ms at maximum load 18 W with supply greater than 20.4 V							

Power Supply	BMXCPS References					
	2000	2010	3020 (H)	3500 (H)	3540 T	4002 (H)
Harmonic rate	10 %	–	–	10 %	–	10 %
Residual ripple included (0 to peak)	–	5 %	5 %	–	5 %	–
1. Limited to 1 ms at maximum load with minimum supply (18 Vdc).						
2. ≤ 10 ms at maximum load 18 W with supply greater than 20.4 V						

Conformity Tests

Installation Wiring and Maintenance

Install, wire, and maintain devices in compliance with the instructions provided in the Grounding and Electromagnetic Compatibility of PLC Systems, Basic Principles and Measures, User Manual (*see page 14*) and Control Panel Technical Guide, How to protect a machine from malfunctions due to electromagnetic disturbance (*see page 14*).

Equipment and Personnel Safety (EC)

Name of Test	Standards	Level
dielectric strength and insulation resistance	IEC/EN 61131-2 IEC 61010-2-201 UL CSA	<ul style="list-style-type: none"> dielectric: $2 U_n + 1000 \text{ V}$; $t = 1 \text{ min}$ PELV: 3000 V insulation: <ul style="list-style-type: none"> $U_n \leq 50 \text{ V}$: $10 \text{ M}\Omega$ $50 \text{ V} \leq U_n \leq 250 \text{ V}$: $100 \text{ M}\Omega$
continuity of earth	IEC/EN 61131-2 IEC 61010-2-201 UL CSA	30 A , $R \leq 0.1 \Omega$, $t = 2 \text{ min}$
leakage current	UL CSA	$\leq 3.5 \text{ mA}$
protection offered by enclosure	IEC/EN 61131-2 IEC 61010-2-201	IP 20 and protection against standardized pins
impact resistance	IEC/EN 61131-2 IEC 61010-2-201 UL CSA	500 g ball dropped from a height 1.3 m (energy 6.8 J minimum)
overload	IEC/EN 61131-2 IEC 61010-2-201 UL CSA	50 cycles, U_n , $1.5 I_n$ $t = 1 \text{ s ON} + 9 \text{ s OFF}$
endurance	IEC/EN 61131-2 IEC 61010-2-201 UL CSA	I_n , U_n 6000 cycles: $t = 1 \text{ s ON} + 9 \text{ s OFF}$
temperature rise	IEC/EN 61131-2 UL CSA ATEX - IECEX	ambient temperature: $+60 \text{ }^\circ\text{C}$ (for ruggedized range (<i>see page 70</i>): $+70 \text{ }^\circ\text{C}$)
U_n nominal voltage I_n nominal current		

NOTE: (EC): tests required by European directives EC and based on IEC/EN 61131-2 standards.

Immunity Tests — L.F. Interference (EC)

Name of Test	Standards	Level
voltage and frequency variations	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-11	0.85 Un , 1.10 Un 0.94 F_n , 1.04 F_n 4 steps t = 30 min
	IACS E10 IEC 61000-4-11	0.80 Un , 1.20 Un 0.90 F_n , 1.10 F_n t = 1.5 s/5 s
direct voltage variations	IEC/EN 61131-2 IEC 61000-4-29 IACS E10 (PAC not connected to charging battery)	0.85 Un + ripple: 5% peak 1.2 Un + ripple: 5% peak 2 steps t = 30 min
third harmonic	IEC/EN 61131-2	H3 (10% Un) 0° / 180° 2 steps t = 5 min
immunity to conducted low frequency (only IACS)	IACS E10	for ac: H2...H15 (10% Un), H15...H100 (10...1% Un), H100...H200 (1% Un) for dc: H2...H200 (10% Un)
voltage interruptions	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-11 IEC 61000-4-29 IACS E10	power supply immunity: 1 ms for dc PS1 / 10 ms for ac or dc PS2 Check operating mode for longer interruptions. for IACS: 30 s for ac or dc
	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-11	for ac PS2 : <ul style="list-style-type: none"> ● 20% Un, t0: 1/2 period ● 40% Un, cycle 10/12 ● 70% Un, cycle 25/30 ● 0% Un, cycle 250/300
voltage shut-down and start-up	IEC/EN 61131-2	Un ...0... Un ; t = Un / 60 s U_{min} ...0... U_{min} ; t = U_{min} / 5 s U_{min} ...0.9 U_{dl} ... U_{min} ; t = U_{min} / 60 s
U_{min} minimum voltage U_{dl} detection level when powered Un nominal voltage F_n nominal frequency PS1 applies to PAC supplied by battery PS2 applies to PAC energized from ac or dc supplies		

Name of Test	Standards	Level
magnetic field	IEC/EN 61131-2 IEC/TS 61000-6-5 IEC 61000-4-8 (for MV power stations: IEC 61850-3)	power frequency: 50/60 Hz 100 A/m continuous 1000 A/m, t = 3 s 3 axes
	IEC 61000-4-10 (for MV power stations: IEC 61850-3)	oscillatory: 100 kHz–1 MHz, 100 A/m t=9 s 3 axes
conducted common mode disturbances range 0...150 kHz	IEC 61000-4-16 (for MV power stations: IEC 61850-3)	for remote systems: <ul style="list-style-type: none"> ● 50/60 Hz and dc, 300 V, t = 1 s ● 50/60 Hz and dc, 30 V, t = 1 min ● 5 Hz...150 kHz, sweep 3...30 V
U_{min} minimum voltage U_{dl} detection level when powered U_n nominal voltage F_n nominal frequency PS1 applies to PAC supplied by battery PS2 applies to PAC energized from ac or dc supplies		

NOTE: (EC): tests required by European directives EC and based on IEC/EN 61131-2 standards.

Immunity Tests — H.F. Interference (EC)

Name of Test	Standards	Level
electrostatic discharges	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-2 IACS E10	6 kV contact 8 kV air 6 kV indirect contact
radiated radio frequency electromagnetic field	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-3 IACS E10	15 V/m, 80 MHz...3 GHz sinusoidal modulation amplitude 80%, 1 kHz + internal clock frequencies
electrical fast transient burst	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-4 IACS E10	for ac and dc main supplies: 2 kV in common mode / 2 kV in wire mode for ac and dc auxiliary supplies, ac unshielded I/Os: 2 kV in common mode for analog, dc unshielded I/Os, communication, and all shielded lines: 1 kV in common mode
surge	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-5 IACS E10	for ac and dc main and auxiliary supplies, ac unshielded I/Os: 2 kV in common mode / 1 kV in differential mode for analog, dc unshielded I/Os: 0.5 kV in common mode / 0.5 kV in differential mode for communication and all shielded lines: 1 kV in common mode

Name of Test	Standards	Level
conducted disturbances induced by radiated electromagnetic fields	IEC/EN 61131-2 IEC/EN 61000-6-2 IEC 61000-4-6 IACS E10	10 V, 0.15...80 MHz sinus amplitude modulated 80%, 1 kHz + spot frequencies
damped oscillatory wave	IEC/EN 61131-2 IEC/EN 61000-4-18 IACS E10	for ac and dc main supplies and ac auxiliary supplies, ac unshielded I/Os: 2.5 kV in common mode / 1 kV in differential mode for dc auxiliary supplies, analog, dc unshielded I/Os: 1 kV in common mode / 0.5 kV in differential mode for communication and all shielded lines: 0.5 kV in common mode

NOTE: These tests are performed without a cabinet, with devices fixed on a metal grid and wired as per the recommendations in the *Grounding and Electromagnetic Compatibility of PLC Systems, Basic Principles and Measures, User Manual* (see page 14).

NOTE: (EC): tests required by European directives EC and based on IEC/EN 61131-2 standards.

Electromagnetic Emissions (EC)

Name of Test	Standards	Level
conducted emission	IEC/EN 61131-2 FCC part 15 IEC/EN 61000-6-4 CISPR 11&22, Class A, Group 1 IACS E10	150...500 kHz: quasi-peak 79 dB (μV/m); average 66 dB (μV/m) 500 kHz...30 MHz: quasi-peak 73 dB (μV/m); average 60 dB (μV/m) ac and dc power (general power distribution zone): <ul style="list-style-type: none"> 10...150 kHz: quasi-peak 120...69 dB (μV/m) 150 kHz...0.5 MHz: quasi-peak 79 dB (μV/m) 0.5...30 MHz: quasi-peak 73 dB (μV/m) ac and dc power (bridge and deck zone for evaluation): <ul style="list-style-type: none"> 10...150 kHz: quasi-peak 96...50 dB (μV/m) 150 kHz...0.35 MHz: quasi-peak 60...50 dB (μV/m) 0.35...30 MHz: quasi-peak 50 dB (μV/m)
radiated emission	IEC/EN 61131-2 FCC part 15 IEC/EN 61000-6-2 CISPR 11&22, Class A, Group 1 IACS E10	30...230 MHz: quasi-peak 40 dB (μV/m) (at 10 m); 50 dB (μV/m) (at 3 m) 230 MHz...1 GHz: quasi-peak 47 dB (μV/m) (at 10 m); 57 dB (μV/m) (at 3 m) for general power distribution zone: <ul style="list-style-type: none"> 0.15...30 MHz: quasi-peak 80...50 dB (μV/m) (at 3 m) 30...100 MHz: quasi-peak 60...54 dB (μV/m) (at 3 m) 100 MHz...2 GHz: quasi-peak 54 dB (μV/m) (at 3 m) 156...165 MHz: quasi-peak 24 dB (μV/m) (at 3 m)

NOTE: (EC): tests required by European directives EC and based on IEC/EN 61131-2 standards.

Withstand Tests — Climatic Variations (Power On)

Name of Test	Standards	Level
dry heat	IEC 60068-2-2 (Bb & Bd)	+60 °C, t = 16 h (for ruggedized range (<i>see page 70</i>): +70 °C, t = 16 h)
	IACS E10	+60 °C, t = 16 h and +70 °C, t = 2 h (for ruggedized range: +70 °C, t = 16 h)
cold	IEC 60068-2-1 (Ab & Ad) IACS E10	0 °C...–25 °C, t = 16 h + power on at 0 °C (for ruggedized range: power on at –25 °C)
damp heat, steady state (continuous humidity)	IEC 60068-2-78 (Cab) IACS E10	+55 °C, 93% relative humidity, t = 96 h (for ruggedized range: +60 °C)
damp heat, cyclic (cyclical humidity)	IEC 60068-2-30 (Db) IACS E10	+55...+25 °C, 93...95% relative humidity, 2 cycles t = 12 h + 12 h
change of temperature	IEC 60068-2-14 (Nb)	0...+60 °C, 5 cycles t = 6 h + 6 h (for ruggedized range: –25...+70 °C)

Immunity Tests — Climatic Variations (Power Off)

Name of Test	Standards	Level
dry heat	IEC/EN 61131-2 IEC 60068-2-2 (Bb & Bd) IEC/EN 60945	+85 °C, t = 96 h
cold	IEC/EN 61131-2 IEC 60068-2-1 (Ab & Ad) IACS E10	–40 °C, t = 96 h
damp heat, cyclic (cyclical humidity)	IEC/EN 61131-2 IEC 60068-2-30 (Db)	+55...+25 °C, 93...95 % relative humidity, 2 cycles t = 12 h + 12 h
change of temperature (thermal shocks)	IEC/EN 61131-2 IEC 60068-2-14 (Na)	–40...+85 °C, 5 cycles t = 3 h + 3 h

Withstand Tests — Mechanical Constraints (Power On)

Name of Test	Standards	Level
sinusoidal vibrations	IEC/EN 61131-2 IEC 60068-2-6 (Fc)	<ul style="list-style-type: none"> • basic IEC/EN 61131-2: 5...150 Hz, +/- 3.5 mm amplitude (5...8.4 Hz), 1 g (8.4...150 Hz) • specific profile: 5...150 Hz, +/- 10.4 mm amplitude (5...8.4 Hz), 3 g (8.4...150 Hz) • for basic and specific, endurance: 10 sweep cycles for each axis
	IACS E10	3...100 Hz, 1 mm amplitude (3...13.2 Hz), 0.7 g (13.2...100 Hz) endurance at each resonance frequency: 90 min for each axis, amplification coefficient < 10
	IEC 60068-2-6	seismic analysis: 3...35 Hz, 22.5 mm amplitude (3...8.1 Hz), 6 g (8.1...35 Hz)
shocks	IEC/EN 61131-2 IEC 60068-2-27 (Ea)	30 g, 11 ms; 3 shocks/direction/axis
		<p>NOTE: In case of using fast actuators (response time ≤ 15 ms) driven by relay outputs: 15 g, 11 ms; 3 shocks/direction/axis.</p> <p>25 g, 6 ms; 100 bumps/direction/axis (bumps)</p> <p>NOTE: In case of using fast actuators (response time ≤ 15 ms) driven by relay outputs: 15 g, 6 ms; 100 bumps/direction/axis.</p>
free fall during operation	IEC/EN 61131-2 IEC 60068-2-32 (Ed Method 1)	1 m, 2 falls

Immunity Tests — Mechanical Constraints (Power Off)

Name of Test	Standards	Level
random free fall with packaging	IEC/EN 61131-2 IEC 60068-2-32 (Method 1)	1 m, 5 falls
flat free fall	IEC/EN 61131-2 IEC 60068-2-32 (Ed Method 1)	10 cm, 2 falls
controlled free fall	IEC/EN 61131-2 IEC 60068-2-31 (Ec)	30° or 10 cm, 2 falls
plugging / unplugging	IEC/EN 61131-2	<p>for modules and connectors:</p> <ul style="list-style-type: none"> • operations: 50 for permanent connections, 500 for non-permanent connections

Specific Environment

Name of Test	Standards	Level
corrosion areas - gas, salt, dust	ISA S71.4	mixed flowing gases: class G3, 25 °C, 75 % relative humidity, t = 14 days
	IEC 60721-3-3	mixed flowing gases: class 3C3, 25 °C, 75 % relative humidity, t = 14 days
	IEC 60068-2-52	salt spray: test Kb, severity 2
	IEC 60721-3-3	sand / dust: class 3S3

Protective Enclosure

M580 PACs are open equipment designed to an IP20 level of ingress protection. For installation in industrial manufacturing workshops or in heat and humidity processing environments, install the M580 PAC in an IP54 enclosure.

NOTE: For IP20 compliance, use a BMX XEM 010 protective cover on empty rack slots.

A system may be installed outside an enclosure if it is operating in a restricted-access room not exceeding pollution level 2 (for example, a control room with no machines or dust-producing activities).

Part II

Installing and Diagnosing Modules on the Local Rack

Introduction

This part provides instructions for installing and assembling M580 CPUs.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	Installing Modules in an M580 Rack	81
4	M580 Diagnostics	91

Chapter 3

Installing Modules in an M580 Rack

Overview

This chapter explains how to install a CPU module in an M580 rack.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Module Guidelines	82
Installing the CPU	83
Installing an SD Memory Card in a CPU	88

Module Guidelines

Guidelines

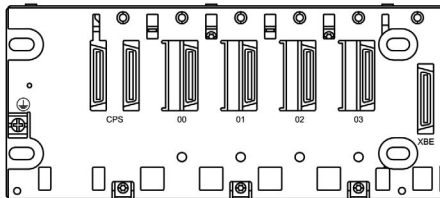
Rack Position	Rack Type	Slots Marking			
		00	01	02	...n (1)
local	main rack	CPU		module	module
	X80 extended rack	module	module	module	module
	Premium extended rack	module	module	module	module
remote drop	main rack	(e)X80 EIO adapter module	module	module	module
	extended rack	module	module	module	module
1 slots from number 03 to last numbered slot of the rack					

NOTE: When your installation has more than one rack in the local rack or at a remote drop, the BMX XBE 1000 rack extender module goes in the slot marked **XBE** of the X80 racks.

Check that the CPU is installed in the two slots marked **00** and **01** on the local rack before powering up the system. If the CPU is not installed in these two slots, the CPU starts in NOCONF state (*see page 31*) and uses the configured IP address (not the default IP address, which starts with 10.10 and uses the last two bytes of the MAC address).

Rack Markings

Example of BMXXBP.... (PV:02 or later) rack with slot markings:



Installing the CPU

Introduction

You can install any standard CPU (BMEP58•0•0) or any Hot Standby CPU (BMEH58•0•0) in these racks:

- BMXXBP•••• (PV:02 or later) X Bus rack
- BMEXBP••00 or BMEXBP••02 Ethernet rack

Exception: You can install the BMXCPS4002 only on these dual-bus (Ethernet and X Bus) racks:

- BMEXBP0602
- BMEXBP1002

Installation Precautions

An M580 CPU is powered by the rack bus. Confirm that the rack power supply is turned off before installing the CPU.

DANGER

HAZARD OF ELECTRIC SHOCK

Remove all power sources before installing the CPU.

Failure to follow these instructions will result in death or serious injury.

Remove the protective cover from the rack slot connectors before plugging the module in the rack.

WARNING

UNEXPECTED EQUIPMENT OPERATION


Check that the CPU does not contain an unsupported SD memory card before powering up the CPU.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: Check that the memory card slot door is closed after a memory card is inserted in the CPU.

NOTE: Refer to %SW97 to check the status of the SD card.

Grounding Considerations

 **DANGER**

ELECTRICAL SHOCK HAZARD

- Switch off the power supply at both ends of the PAC connection, and lock out and tag out both the power sources.
- In case lock out and tag out are not available, ensure that the power sources cannot be inadvertently switched on.
- Use suitable insulation equipment when inserting or removing all or part of this equipment.

Failure to follow these instructions will result in death or serious injury.

Use fiber-optic cable to establish a communications link when it is not possible to equalize the potential between the two grounds.

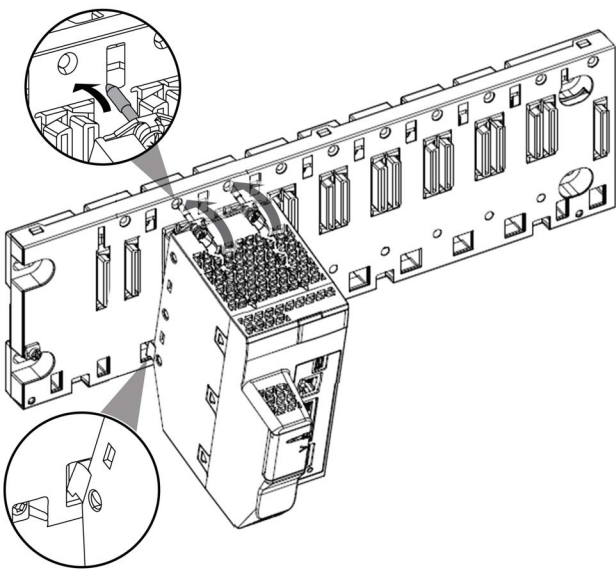
NOTE: Refer to the ground protection information provided in the *Grounding and Electromagnetic Compatibility of PLC Systems, Basic Principles and Measures, User Manual (see page 14)* and *Control Panel Technical Guide, How to protect a machine from malfunctions due to electromagnetic disturbance (see page 14)*.

Installing the CPU and other Modules in the Rack

Install the CPU in the rack slots marked **00** and **01**. If you do not install the CPU in these two slots, it starts in NOCONF state (see page 31) state and uses the default IP address, which starts with 10.10 and uses the last two bytes of the MAC address.

Follow these steps to install a CPU in a rack:

Step	Action	Illustration
1	Verify that the power supply is turned off.	–
2	<p>If you are installing a Hot Standby CPU, on the back of the CPU, set the A/B/Clear selector switch (see page 44) to the appropriate selection, “A” or “B”.</p> <p>NOTE: When you later install the companion Hot Standby CPU, set its rotary switch to the other A/B position.</p>	–

Step	Action	Illustration
3	Verify that: <ul style="list-style-type: none"> • if an SD memory card is used, it is supported by the CPU • the connectors' protective covers are removed • the CPU is placed on the slots marked 00 and 01 	
4	Position the locating pins situated at the rear of the module (on the bottom part) in the corresponding slots in the rack.	
5	Swivel the module towards the top of the rack so that the module sits flush with the back of the rack. The module is now set in position.	
6	Tighten the 2 screws on top of the CPU to maintain the module in place on the rack. tightening torque: 1.5 N.m (1.106 lbf ft) max.	—
7	For each additional module, repeat steps 4, 5 and 6 until all modules are installed on the rack.	—

Installing Modules in the Second Local Rack

If you are installing a Hot Standby system, you need to install the same collection of modules, with the same versions of firmware, that were installed on the first rack. Install each module in the same slot that its counterpart occupies on the first rack. Follow the same procedure described above, except set the A/B/Clear selector switch (*see page 44*) on the back of the standby CPU to other A/B position.

Connecting the Hot Standby Local Racks

If you are installing a Hot Standby system, you need to connect the communication link to CPU A and CPU B before applying power to either local rack. If you start up the CPUs before they are connected via the Hot Standby link, both CPUs attempt to assume the role of primary CPU in your Hot Standby system.

DANGER

HAZARD OF ELECTRIC SHOCK

Ground the power supplies by connecting the protective earth ground terminal on each power supply module to the protective earth ground of the installation. Connect them in either of the following ways:

- Connect the protective earth ground terminal of the power supply to the protective earth ground of the installation with a separate cable, independent of the rack ground cable.
- Connect the protective earth ground terminal of the power supply to the ground screw of the rack (where the rack itself is grounded).

Do not connect anything else to the power supply ground.

Failure to follow these instructions will result in death or serious injury.

DANGER

HAZARD OF ELECTRIC SHOCK

- Use only cables with ring or spade lugs and check that there is a good ground connection.
- Make sure that grounding hardware is tightened properly.

Failure to follow these instructions will result in death or serious injury.

Before you connect the two Hot Standby local racks, verify that an equipotential grounding system is in place that includes the two racks (plus any other equipment you intend to connect to the two Hot Standby local racks).

NOTICE

UNINTENDED EQUIPMENT OPERATION

When installing modules with fiber optic transceivers, do the following to help prevent dust and pollution from disrupting light production into the fiber optic cable.

- Keep caps on jumpers and transceivers when not in use.
- Insert the optical cable into the transceivers carefully, respecting the longitudinal axis of the transceiver.
- Do not use force when inserting the cable into the optical transceivers.

Failure to follow these instructions can result in equipment damage.

Each Hot Standby CPU includes on its front face an SFP socket ([see page 43](#)). This socket can accept an SFP transceiver module (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*) for either copper or single mode fiber optic cabling of the Hot Standby link. Your choice of SFP transceiver and cabling is determined by the distance between the two Hot Standby local racks (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*).

Installing an SD Memory Card in a CPU

Introduction

The BME•58•••• CPUs support the use of the BMXRMS004GPF 4GB SD memory card.

Memory Card Maintenance

To keep the memory card in normal working order:

- Avoid removing the memory card from its slot when the CPU accesses the card (memory card access green LED ON or blinking).
- Avoid touching the memory card connectors.
- Keep the memory card away from electrostatic and electromagnetic sources as well as heat, sunlight, water, and moisture.
- Avoid impact on the memory card.
- Before sending a memory card by post (mail), check the postal service security policy. In some countries, the postal service exposes mail to high levels of radiation as a security measure. These high levels of radiation may erase the contents of the memory card and render it unusable.
- If a card is extracted without generating a rising edge of the bit %S65 and without checking that the memory card access green LED is OFF, the data (files, application, and so on) may be lost or become unreliable.

Memory Card Insertion Procedure

Procedure for inserting a memory card into a BME•58•••• CPU:

Step	Description
1	Open the SD memory card protective door.
2	Insert the card in its slot.
3	Push the memory card until you hear a click. Result: The card should now be clipped into its slot. Note: Insertion of the memory card does not force an application restore.
4	Close the memory card protective door.

Memory Card Removal Procedure

NOTE: Before removing a memory card, a rising edge on bit %S65 needs to be generated. If a card is extracted without generating a rising edge of the bit %S65 and without checking that the memory card access green LED is OFF, the data may be lost.

Procedure for removing a memory card from a BME•58•••• CPU:

Step	Description
1	Generate a rising edge on bit %S65.
2	Check that the memory card access green LED is OFF.
3	Open the SD memory card protective door.
4	Push the memory card until you hear a click, then release the pressure on the card. Result: The card should unclip from its slot.
5	Remove the card from its slot. Note: The memory card access green LED is ON when the memory card is removed from the CPU.
6	Close the memory card protective door.

Chapter 4

M580 Diagnostics

Introduction

This chapter provides information on diagnostics that can be performed via hardware indications (based on LED status) and system bits or words when necessary. The entire M580 system diagnostics is explained in the *Modicon M580 System Planning Guide*.

The CPU manages different types of detected error:

- detected errors that can be recovered and do not change the PAC behavior unless specific options are used
- detected errors that cannot be recovered and lead the CPU to the halt state
- CPU or system detected errors that lead the CPU to an error state

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Blocking Conditions	92
Non-blocking Conditions	94
CPU or System Errors	95
CPU Application Compatibility	96

Blocking Conditions

Introduction

Blocking conditions caused during the execution of the application program do not cause system errors, but they stop the CPU. The CPU goes into the HALT state ([see page 31](#)).

NOTE:

- When a BMEH58-040 CPU is in the HALT state, the RIO and DIO outputs behave the same way as they do when the CPU is in STOP state ([see page 349](#)).
- For information about Hot Standby diagnostics, refer to the diagnostics chapter (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*) in the M580 Hot Standby installation guide.

Diagnostics

Visual indications of a blocking condition are the **ERR** LED on the CPU front panel ([see page 47](#)).

A description of the error is provided in system word %SW125.

The address of the instruction that was executing when the blocking condition occurred is provided by system words %SW126 through %SW127.

%SW125 system word values and corresponding blocking condition description:

%SW125 Value (hex)	Blocking Condition Description
0...	execution of an unknown function
0002	SD card signature feature (used with SIG_CHECK and SIG_WRITE functions)
2258	execution of the HALT instruction
2259	execution flow different than the reference flow
23..	execution of a CALL function towards an undefined subroutine
81F4	SFC node incorrect
82F4	SFC code inaccessible
83F4	SFC work space inaccessible
84F4	too many initial SFC steps
85F4	too many active SFC steps
86F4	SFC sequence code incorrect
87F4	SFC code description incorrect
88F4	SFC reference table incorrect
89F4	SFC internal index calculation detected error
8AF4	SFC step status not available
8BF4	SFC memory too small after a change due to a download

%SW125 Value (hex)	Blocking Condition Description
8CF4	transition/action section inaccessible
8DF4	SFC work space too small
8EF4	version of the SFC code older than the interpreter
8FF4	version of the SFC code more recent than the interpreter
90F4	poor description of an SFC object: NULL pointer
91F4	action identifier not authorized
92F4	poor definition of the time for an action identifier
93F4	macro step cannot be found in the list of active steps for deactivation
94F4	overflow in the action table
95F4	overflow in the step activation/deactivation table
9690	error detected in the application CRC check (checksum)
DE87	calculation detected error on numbers with decimal points
DEB0	watchdog overrun
DEF0	division by 0
DEF1	character string transfer detected error
DEF2	capacity exceeded
DEF3	index overrun
DEF7	SFC execution detected error
DEFE	SFC steps undefined

Restarting the Application

After a blocking condition has occurred, the halted CPU needs to be initialized. The CPU can also be initialized by setting the %S0 bit to 1.

When initialized, the application behaves as follows:

- the data resume their initial value
- tasks are stopped at end of cycle
- the input image is refreshed
- outputs are controlled in fallback position

The RUN command then allows the application to be restarted.

Non-blocking Conditions

Introduction

The system enters a non-blocking condition when it detects an input/output error on the backplane bus (X Bus or Ethernet) or through execution of an instruction, which can be processed by the user program and does not modify the CPU status.

Conditions Linked to I/O Diagnostics

A non-blocking condition linked to the I/O is diagnosed with the following indications:

- CPU I/O LED pattern: steady ON
- module I/O LED pattern: steady ON
- system bits (type of error):
 - %S10 set to 0: I/O error detected on one of the modules on the rack (channel power supply detected error, or broken channel, or module not compliant with the configuration, or inoperative module, or module power supply detected error)
 - %S16 set to 0: I/O error detected in the task in progress
 - %S40–%S47 set to 0: I/O error detected on rack address 0 to 7
- system bits and words combined with the channel having an error detected (I/O channel number and type of detected error) or I/O module Device DDT information (for modules configured in Device DDT addressing mode):
 - bit %Ir.m.c.ERR set to 1: channel error detected (implicit exchanges)
 - word %MWIr.m.c.2: the word value indicates the type of error detected on the specified channel and depends on the I/O module (implicit exchanges)

Conditions Linked to Execution of the Program Diagnostics

A non-blocking condition linked to execution of the program is diagnosed with the following system bits and words:

- system bits (type of error detected):
 - %S15 set to 1: character string manipulation error detected
 - %S18 set to 1: capacity overrun, error detected on a floating point, or division by 0
 - %S20 set to 1: index overrun
- system word (nature of the error detected):
 - %SW125 (*see page 92*) (always updated)

NOTE: The CPU can be forced to the HALT state (*see page 31*) on program execution recoverable condition.

There are 2 ways to force a CPU to stop when non-blocking errors linked to the execution of the program are detected:

- Use the diagnostic program function accessible through Unity Pro programming software.
- set the system bit %S78 (HALTIFERROR) to 1.

CPU or System Errors

Introduction

CPU or system errors are related either to the CPU (equipment or software) or to the rack internal bus wiring. The system can no longer operate correctly when these errors occur.

A CPU or system error causes the CPU to stop in ERROR mode and requires a cold restart. Before applying a cold restart, set the CPU to STOP mode to keep the PAC from returning to ERROR mode.

Diagnostics

A CPU or system error is diagnosed with the following indications:

- CPU I/O LED pattern: steady on
- system word %SW124 value defines the detected error source:
 - 80 hex: system watchdog error or rack internal bus wiring error
 - 81 hex: rack internal bus wiring error
 - 90 hex: interruption not foreseen, or system task pile overrun

CPU Application Compatibility

Application Compatibility

These tables show the standalone (BMEP58•0•0) and Hot Standby (BMEH58•0•0) CPUs that can download and execute applications that are built on a different CPU.

These applications are built on standalone CPUs and transferred to standalone CPUs:

Standalone CPUs	Download and execute the application here (BMEP58...								
Build the application here (↓).	1020	2020	2040	3020	3040	4020	4040	5040	6040
BMEP581020	X	X	–	X	–	X	–	–	–
BMEP582020	–	X	–	X	–	X	–	–	–
BMEP582040	–	–	X	–	X	–	X	X	X
BMEP583020	–	–	–	X	–	X	–	–	–
BMEP583040	–	–	–	–	X	–	X	X	X
BMEP584020	–	–	–	–	–	X	–	–	–
BMEP584040	–	–	–	–	–	–	X	X	X
BMEP585040	–	–	–	–	–	–	–	X	X
BMEP586040	–	–	–	–	–	–	–	–	X
X yes – no									

These applications are built on Hot Standby CPUs and transferred to Hot Standby CPUs:

Hot Standby CPUs	Download and execute the application here (BMEH58...		
Build the application here (↓).	2040	4040	6040
BMEH582040	X	X	X
BMEP584040	–	X	X
BMEP586040	–	–	X
X yes – no			

Example: An application built on a BMEP583020 CPU can only be downloaded or executed on a BMEP583020 or a BMEP584020 CPU.

NOTE: For all M580 CPUs, versions 1.10 and 2.00 are not compatible. You cannot configure a CPU V2.00, and download the application to a CPU V1.10.

Part III

Configuring the CPU in Unity Pro

Introduction

This part describes how to configure a M580 system with Unity Pro.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
5	M580 CPU Configuration	99
6	M580 CPU Programming and Operating Modes	355

Chapter 5

M580 CPU Configuration

Introduction

The chapter describes the configuration of the M580 CPU.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	Unity Pro Projects	100
5.2	Configuring the CPU with Unity Pro	112
5.3	Configuring the M580 CPU with DTMs in Unity Pro	132
5.4	Diagnostics through the Unity Pro DTM Browser	139
5.5	Online Action	155
5.6	Diagnostics Available through Modbus/TCP	162
5.7	Diagnostics Available through EtherNet/IP CIP Objects	165
5.8	DTM Device Lists	202
5.9	Explicit Messaging	226
5.10	Explicit Messaging Using the MBP_MSTR Block in Quantum RIO Drops	256
5.11	Implicit Messaging	279
5.12	Configuring the M580 CPU as an EtherNet/IP Adapter	304
5.13	Hardware Catalog	316
5.14	M580 CPU Embedded Web Pages	324
5.15	M580 Hot Standby CPU Web Pages	343

Section 5.1

Unity Pro Projects

Overview

Use this section to add an M580 CPU to your Unity Pro application.

What Is in This Section?

This section contains the following topics:

Topic	Page
Creating a Project in Unity Pro	101
Helping Secure a Project in Unity Pro	103
Configuring the Size and Location of Inputs and Outputs	105
Project Management	108
DIO Scanner Functionality	110

Creating a Project in Unity Pro

Introduction

If you have not created a project in Unity Pro and installed a power supply and an M580 CPU, use the following steps to create a new Unity Pro project containing these components:

- M580 CPU (*see page 19*)
- power supply

Creating and Saving a Unity Pro Project

Follow these steps to create a Unity Pro project:

Step	Action
1	Open Unity Pro.
2	Click File → New... to open the New Project window.
3	In the PLC window, expand the Modicon M580 node, and select a CPU. NOTE: Refer to the CPU Scanner Service (<i>see page 23</i>) topic to select the appropriate CPU, depending upon your DIO and RIO needs. In the Rack window, expand the Modicon M580 local drop node, and select a rack.
4	Click OK . Result: The Project Browser dialog opens.
5	Click File → Save to open the Save As dialog.
6	Enter a File name for your Unity Pro project and click Save . Result: Unity Pro saves your project to the specified path location.

Changing the Default Storage Location (Optional)

You can change the default location that Unity Pro uses to store project files before you click **Save**:

Step	Action
1	Click Tools → Options to open the Options Management window.
2	In the left pane, navigate to Options → General → Paths .
3	In the right pane, type in a new path location for the Project path . You can also edit these items: <ul style="list-style-type: none"> • Import/Export file path • XVM path • Project settings templates path
4	Click OK to close the window and save your changes.

Selecting a Power Supply

A default power supply is automatically added to the rack in a new Unity Pro project. To use a different power supply, follow these steps:

Step	Action
1	In the Project Browser , double-click PLC Bus to display a graphical representation of the hardware rack: <ul style="list-style-type: none">• The selected M580 CPU is in the second position.• A default power supply appears in the first position.• Unity Pro automatically opens the Hardware Catalog that corresponds to the PLC bus tab.
2	Select the power supply automatically added to the PLC bus .
3	Press the Delete key to remove the power supply.
4	Double-click the first slot of the PLC bus to open the New Device list.
5	Double-click the preferred power supply to make it appear in the PLC bus .
6	File → Save Click to save your project.

Helping Secure a Project in Unity Pro

Creating an Application Password

In Unity Pro, create a password to help protect your application from unwanted modifications. The password is encrypted and stored in the PAC. Any time the application is modified, the password is required.

Step	Action
1	In the Project Browser window, right-click Project → Properties .
2	In the Properties of Project window, click the Protection tab.
3	In the Application field, click Change password .
4	In the Modify Password window, enter a password in the Entry and Confirmation fields.
5	Click OK .
6	In the Application field, select the Auto-lock check box if you want to require the password to resume the application display. You may also click the up/down arrows to set the number of minutes at which time the application would auto-lock.
7	To save the changes: <ul style="list-style-type: none"> ● Click Apply to leave the Properties of Project window open. – or – ● Click OK to close the window.
8	Click File → Save to save your application.
9	If you wish to change the password at a later time, follow the preceding steps.

More information about application password is given in Application Protection (*see Unity Pro, Operating Modes*) page.

NOTE: when Exporting a project to a .XEF or a .ZEF file, the application password is cleared.

Using Memory Protect

In Unity Pro, select the **Memory Protect** option to help protect your application from unwanted modifications.

Step	Action
1	In the Project Browser window, expand the Configuration folder to display the CPU.
2	To open the CPU configuration window: <ul style="list-style-type: none">● Double-click the CPU. – or –● Right-click BME P58 •0•0 → Open.
3	In the CPU window, click the Configuration tab.
4	Select the Memory protect check box, and enter an input address of your choice.
5	Click File → Save to save your application.

Configuring the Size and Location of Inputs and Outputs


Introduction

In the Unity Pro **Project Browser**, double-click **PLC Bus** to display the main rack. Then click on the CPU (but not on the Ethernet connectors) to open the CPU configuration window.

Setting Global Addresses and Operating Mode Parameters

Click on the **Configuration** tab to edit the size and starting positions of inputs and outputs:

Step	Action	
1	Double-click the image of the M580 CPU in the PLC Bus to view its properties.	
2	Select the Configuration tab.	
3	In the Operating mode area, select the boxes to enable the following parameters in your application:	
4	Run/Stop input	Use these two parameters to place the PAC into Run or Stop mode. For more information regarding the effect of these parameters, refer to the topic Managing Run/Stop Input (<i>see page 363</i>). (default = de-selected)
	Run/Stop by input only	
	Memory protect	This function is activated by an input bit. It prohibits the transfer of a project into the PAC and modifications in online mode, regardless of the communication channel. The Run and Stop commands are authorized. (default = de-selected)
	Automatic start in Run	The enabling of this option automatically places the PAC into RUN mode in the event of a cold start. (default = de-selected)
	Initialize %MWi on cold start	<p>On a cold start (<i>see page 366</i>) or on download if you select the box (default state):</p> <ul style="list-style-type: none"> • The %MWi are handled like other global variables (initialized to 0 or initial value, according to current application) in all cold start cases. <p>On cold start or on download if you de-select the box:</p> <ul style="list-style-type: none"> • If %MW were previously saved in internal flash memory (using the %SW96 word) they are restored from internal flash memory, • If not: <ul style="list-style-type: none"> ○ If cold start is linked to a power-off or of a push on the reset button, the %MW are initialized. ○ If not, the current values of %MW are maintained. <p>NOTE: if the new (or restored) application has more %MW than the previous one, the added %MW are set to 0 (non-zero initial values are not applied)</p>
	Cold Start Only	<p>If selected, this option forces the cold start (<i>see page 367</i>) of the application, instead of the normal warm start. By default, the Cold Start Only option is unchecked. An application using this function is not:</p> <ul style="list-style-type: none"> • Downloadable to a PAC with a previous version. • Executable on a PAC with a previous version.

Step	Action										
5	<p>Configure the size of the memory locations in the Size of global address fields.</p> <p>NOTE: High end standalone and Hot Standby CPUs (BMEP584040, BMEP585040, BMEP586040, BMEH584040 and BMEH586040) include State RAM memory management. The State RAM feature supports LL984 logic sections for converted LL984 applications, plus Quantum Ethernet RIO drops.</p> <p>The following memory management options are presented in the Configuration tab:</p> <table border="1"> <tr> <td>Mem usage</td><td>The percentage of CPU memory usage, based on the cumulative values input into the %M, %MW, %I, and %IW fields, below. (Supported only by high end standalone and Hot Standby CPUs that support State RAM.)</td></tr> <tr> <td>%M-0x</td><td rowspan="5"> <p>Enter the appropriate value for each address field type. (%I and %IW are supported only by high end standalone and Hot Standby CPUs that support State RAM.)</p> <p>NOTE: The values for %IW and %MW, have to be divisible by 8 for version before 2.30 and divisible by 128 for other versions. The value for %KW have to be divisible by 8 for all versions.</p> </td></tr> <tr> <td>%MW-4x</td></tr> <tr> <td>%I-1x</td></tr> <tr> <td>%IW-3x</td></tr> <tr> <td>%KW</td></tr> <tr> <td>Viewer</td><td>Opens the State RAM Viewer, which displays the allocation of used memory.</td></tr> </table> <p>NOTE: To input:</p> <ul style="list-style-type: none"> Maximum values: Click the Maximum values button, select the appropriate boxes in the Max column, then click OK. Default values: Click the Default values button, select the appropriate boxes in the Default column, then click OK. <p>NOTE: M580 / S908 applications: In M580 CPUs that are compatible with Quantum S908 network adapter (<i>see Modicon Quantum 140CRA31908, Adapter Module, Installation and Configuration Guide</i>) and an OS version ≥ 02.30: (number of %I + number of %M) ≤ 65535. The maximum number of %I is 65280. The maximum number of %M is 65280.</p>	Mem usage	The percentage of CPU memory usage, based on the cumulative values input into the %M, %MW, %I, and %IW fields, below. (Supported only by high end standalone and Hot Standby CPUs that support State RAM.)	%M-0x	<p>Enter the appropriate value for each address field type. (%I and %IW are supported only by high end standalone and Hot Standby CPUs that support State RAM.)</p> <p>NOTE: The values for %IW and %MW, have to be divisible by 8 for version before 2.30 and divisible by 128 for other versions. The value for %KW have to be divisible by 8 for all versions.</p>	%MW-4x	%I-1x	%IW-3x	%KW	Viewer	Opens the State RAM Viewer , which displays the allocation of used memory.
Mem usage	The percentage of CPU memory usage, based on the cumulative values input into the %M, %MW, %I, and %IW fields, below. (Supported only by high end standalone and Hot Standby CPUs that support State RAM.)										
%M-0x	<p>Enter the appropriate value for each address field type. (%I and %IW are supported only by high end standalone and Hot Standby CPUs that support State RAM.)</p> <p>NOTE: The values for %IW and %MW, have to be divisible by 8 for version before 2.30 and divisible by 128 for other versions. The value for %KW have to be divisible by 8 for all versions.</p>										
%MW-4x											
%I-1x											
%IW-3x											
%KW											
Viewer	Opens the State RAM Viewer , which displays the allocation of used memory.										
6	Select the Online modification in RUN or STOP check box (in the Configuration Online Modification field) to use the change configuration on the fly (CCOTF) feature.										
7	Select Edit → Validate (or click the  toolbar button) to save the configuration.										

NOTE:

- After you validate module settings for the first time, you cannot edit the module name. If you subsequently decide to change the module name, delete the existing module from the configuration, then add and rename a replacement module.
- In addition to the Configuration tab, described above, the CPU configuration window presents an **I/O Objects** tab, and an **Animation** tab with three sub-tabs: Task, Real-time Clock, and Information.

Completing the Ethernet Network Configuration

After you configure these settings, configure the CPU settings beginning with its Channel Properties. Then configure the Ethernet network devices.

Project Management

Downloading the Application to the CPU

Download the Unity Pro application to the CPU through one of its ports or through a connection to an Ethernet communication module:

Method	Connection
USB port	If the CPU and the PC that are running Unity Pro both have USB ports, you can download the application to the CPU directly through the USB ports (<i>see page 53</i>) (version 1.1 or later).
Ethernet port	If the CPU and the PC that are running Unity Pro both have Ethernet ports, you can download the application to the CPU directly through the Ethernet ports.
communication module	You can download the application to the CPU by connecting Unity Pro to the IP address of a communication module.

NOTE: For details, refer to *Downloading CPU Applications (see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures)* in the *Modicon M580 Hot Standby System Planning Guide for Frequently Used Architectures*.

Converting Legacy Applications to M580

For details on this conversion process, contact your Schneider Electric customer support.

Restoring and Backing Up Projects

The CPU application RAM (*see page 361*) and the CPU flash memory automatically and manually perform the following:

- Restore a project in the CPU from the flash memory (and the memory card if inserted):
 - Automatically after a power cycle
 - Automatically on a warm restart
 - Automatically on a cold start
 - Manually with a Unity Pro command: **PLC → Project Backup → Backup Restore**

NOTE: If a memory card is inserted with a different application than the application in the CPU, the application is transferred from the memory card to the CPU application RAM when the restore function is carried out.

- Save the CPU project in the flash memory (and the memory card if inserted):
 - Automatically after an online modification is performed in the application RAM
 - Automatically after a download
 - Automatically on detection of %S66 system bit rising edge
 - Manually with a Unity Pro command: **PLC → Project Backup → Backup Save**

NOTE: Backup begins after the completion of the current MAST cycle and before the start of the next MAST cycle.

If MAST is configured as periodic, set the MAST period to a value larger than the actual MAST execution time. This lets the processor complete an entire backup without interruption.

If the MAST period is set to a value less than the actual MAST execution time, backup processing is fragmented and requires a longer time to finish.

- Compare the CPU project and the flash memory project:
 - Manually with a Unity Pro command: **PLC → Project Backup → Backup Compare**

NOTE: When a valid memory card is inserted (*see page 59*) with a valid application, the application backup and restore operations are performed as follows:

- The application backup is performed on the memory card first and then on the flash memory.
- The application restore is performed from the memory card to the CPU application RAM first and then copied from the application RAM to the flash memory.

DIO Scanner Functionality

Introduction

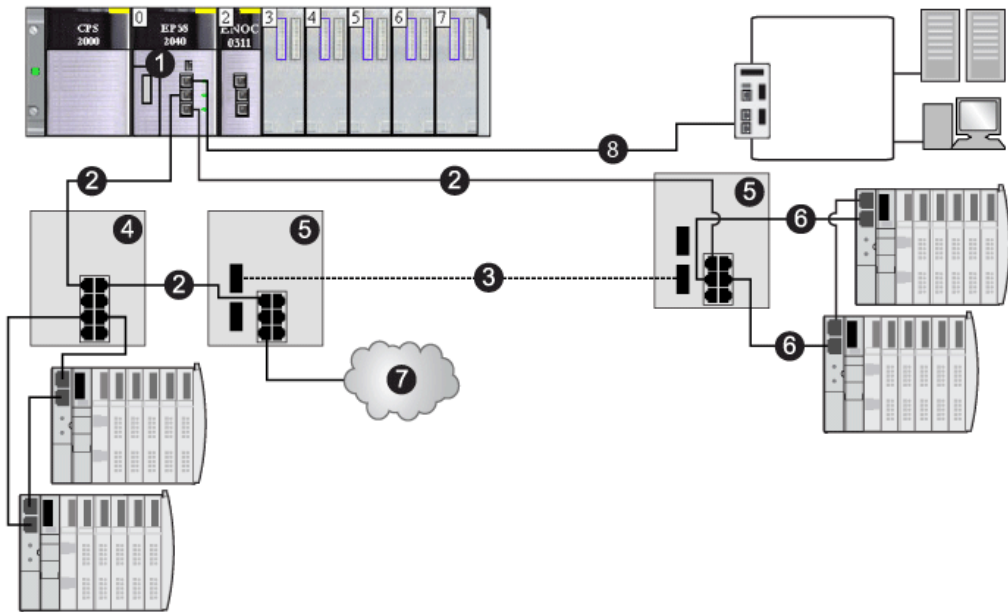
An embedded DIO scanner service in a standalone (BMEP58•0•0) or Hot Standby (BMEH58•0•0) M580 CPU can manage distributed equipment. Through this service, Ethernet gateway devices (like Profibus and CANopen masters) can operate as distributed equipment.

All DIO scanning communications occur over the Ethernet backplane or through an Ethernet port.

NOTE: The BMEP58•040 CPUs also manage RIO modules through the RIO scanner service, but this discussion applies to the DIO scanner service.

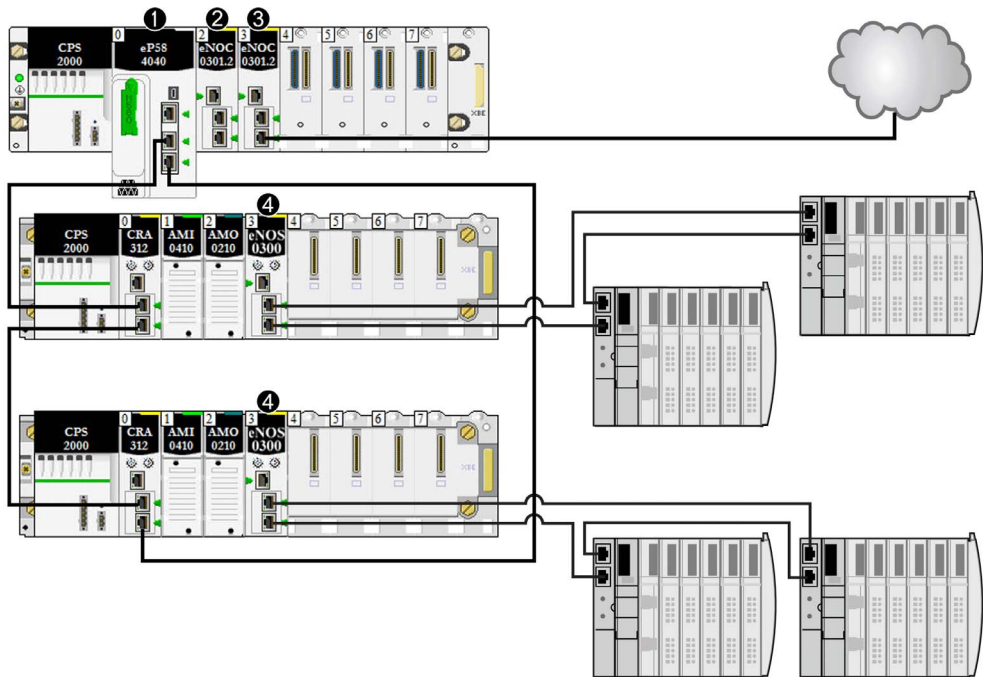
DIO Scanner Service Overview

In this network example, the CPU is connected to the DIO network (2) and the control network (8).



- 1 a CPU with an embedded DIO scanner service
- 2 copper portion of the main ring
- 3 fiber portion of the main ring
- 4 DRS connecting a DIO sub-ring to the main ring
- 5 DRS configured for copper-to-fiber and fiber-to-copper transition connecting a DIO sub-ring to the main ring
- 6 DIO sub-ring
- 7 DIO cloud
- 8 CPU connecting the control network to the M580 system

This illustration shows direct connections to distributed equipment:



- 1 A CPU on the main rack runs the Ethernet I/O communication server service.
- 2 A BMENOC0301/11 Ethernet communication module (Ethernet backplane connection disabled) manages distributed equipment on the device network.
- 3 A BMENOC0301/11 Ethernet communication module (Ethernet backplane connection enabled) is connected to a DIO cloud.
- 4 A BMENOS0300 network option switch module is connected to a DIO sub-ring.

Section 5.2

Configuring the CPU with Unity Pro

Introduction

Use the instructions in this section to configure the M580 CPU in Unity Pro.

NOTE: Some configuration features for the M580 CPU are accessed through the Unity Pro **DTM Browser**. Those configuration instructions appear elsewhere in this document (*see page 132*).

What Is in This Section?

This section contains the following topics:

Topic	Page
Unity Pro Configuration Tabs	113
About Unity Pro Configuration	114
Security Tab	115
IPConfig Tab	119
RSTP Tab	121
SNMP Tab	123
NTP Tab	125
Switch Tab	128
QoS Tab	129
Service Port Tab	130
Advanced Settings Tab	131

Unity Pro Configuration Tabs

Accessing the Unity Pro Configuration Tabs

Access the CPU configuration parameters for RIO and distributed equipment:

Step	Action
1	Open a project that includes an M580 CPU that supports RIO and DIO networks.
2	In the Project Browser , double-click Project → Configuration → PLC bus .
3	In the PLC bus dialog box, double-click the drawing with 3 Ethernet ports in the middle of the CPU.
4	In the Security tab, check to see that the services that you require are enabled (<i>see page 116</i>). (See the Note below.)
5	In the IPConfig tab, you may change the IP address of the CPU or you may configure the default address, which starts with 10.10 and uses the last 2 bytes of the MAC address.

NOTE: For improved security, some of the communication services (FTP, TFTP, and HTTP) are disabled by default. You may wish to perform some actions (such as a firmware update, web access, or remote I/O) that require the availability of one or more of these services. Before configuring Ethernet parameters, set the security levels (*see page 115*) to meet your requirements. When these services are not needed, you should disable them.

Unity Pro Configuration Tabs

This table indicates the Unity Pro configuration tabs that are available (X) and unavailable (—) for M580 CPUs:

Unity Pro Tab	Services	
	CPUs with Embedded RIO Scanning (BME•58•040)	CPUs without Embedded RIO Scanning (BME•58•020)
Security	X	X
IPConfig	X	X
RSTP	X	X
SNMP	X	X
NTP	X	X
Switch	—	X
QoS	—	X
Service Port	X	X
Advanced Settings	—	X

NOTE: To maintain RIO performance, you cannot access these tabs for BME•58•040 CPUs.

About Unity Pro Configuration

Accessing Configuration Settings

Access the configuration settings for the M580 CPU in Unity Pro:

Step	Action
1	Open Unity Pro.
2	Open a Unity Pro project that includes an M580 CPU in the configuration.
3	Open the Project Browser (Tools → Project Browser) .
4	Double-click PLC bus in the Project Browser .
5	<p>In the virtual rack, double-click the Ethernet ports of the M580 CPU to see these configuration tabs:</p> <ul style="list-style-type: none">● Security● IPConfig● RSTP● SNMP● NTP● Switch (See the note.)● QoS (See the note.)● Service Port● Advanced Settings (See the note.) <p>These configuration tabs are described in detail in the pages that follow.</p> <p>NOTE: This tab is not available for CPUs that provide the RIO Ethernet scanning services.</p>

Security Tab

Introduction

Unity Pro provides security services for the CPU. Enable and disable these services on the **Security** tab in Unity Pro.

Accessing the Security Tab

View the **Security** configuration options:

Step	Action
1	Open your Unity Pro project.
2	Double-click the Ethernet ports on the CPU in the local rack (or right-click the Ethernet ports and select Open Submodule).
3	Select the Security tab in the RIO DIO Communicator Head window to enable/disable Ethernet services.

Available Ethernet Services

You can enable/disable these Ethernet services:

Field	Comment
Enforce Security and Unlock Security	Refer to the description, below, for details. <i>(see page 116)</i>
FTP	Enable or disable (default) firmware upgrade, SD memory card data remote access, data storage remote access, and device configuration management using the FDR service. NOTE: Local data storage remains operational, but remote access to data storage is disabled.
TFTP	Enable or disable (default) the ability to read RIO drop configuration and device configuration management using the FDR service. NOTE: Enable this service to use eX80 Ethernet adapter modules.
HTTP	Enable or disable (default) the web access service.
DHCP / BOOTP	Enable or disable (default) the automatic assignment of IP addressing settings. For DHCP, also enable/disable automatic assignment of subnet mask, gateway IP address, and DNS server names.
SNMP	Enable or disable (default) the protocol used to monitor the device.
EIP	Enable or disable (default) access to the EtherNet/IP server.
Access Control	Enable (default) or disable Ethernet access to the multiple servers in the CPU from unauthorized network devices.
1 Set Access Control to Enabled to modify this field.	

Field		Comment
Authorized addresses ⁽¹⁾	Subnet	Yes/No
	IP Address	0.0.0.0 ... 223.255.255.255
	Subnet mask	224.0.0.0 ... 255.255.255.252
	FTP	Select this to grant access to the FTP server in the CPU.
	TFTP	Select this to grant access to the TFTP server in the CPU.
	HTTP	Select this to grant access to the HTTP server in the CPU.
	Port 502	Select this to grant access to port 502 (typically used for Modbus messaging) of the CPU.
	EIP	Select this to grant access to the EtherNet/IP server in the CPU.
	SNMP	Select this to grant access to the SNMP agent resident in the CPU.
¹ Set Access Control to Enabled to modify this field.		

NOTE: Refer to the ETH_PORT_CTRL topic (*see page 373*) for information regarding using this function block to control the FTP, TFTP, HTTP, and DHCP/BOOTP protocols.

Enable/Disable Ethernet Services

You can enable/disable Ethernet services on the **Security** tab as follows:

- Enable/disable FTP, TFTP, HTTP, EIP, SNMP, and DHCP/BOOTP for all IP addresses. (You can use this feature offline only. The configuration screen is grayed in online mode.)
– or –
- Enable/disable FTP, TFTP, HTTP, Port 502, EIP, and SNMP for each authorized IP address. (You can use this feature online.)

Set the **Security** tab parameters before you download the application to the CPU. The default settings (maximum security level) reduce the communication capacities and port access.

NOTE: Schneider Electric recommends disabling services that are not being used.

Enforce Security and Unlock Security Fields

- When you click **Enforce Security** (the **Security** tab default setting):
FTP, TFTP, HTTP, EIP, SNMP, and DHCP/BOOTP are disabled and **Access Control** is enabled.
- When you click **Unlock Security**:
FTP, TFTP, HTTP, EIP, SNMP, and DHCP/BOOTP are enabled, and **Access Control** is disabled.

NOTE: You can set each field individually once the global setting is applied.

Using Access Control for Authorized Addresses

Use the **Access Control** area to restrict device access to the CPU in its role as a server. After you enable access control in the **Security** dialog, you can add the IP addresses of the devices that you want to communicate with the CPU to the list of **Authorized Addresses**:

- By default, the IP address of the CPU's embedded Ethernet I/O scanner service with **Subnet** set to **Yes** allows any device in the subnet to communicate with the CPU through EtherNet/IP or Modbus TCP.
- Add the IP address of any client device that may send a request to the CPU's Ethernet I/O scanner service, which, in this case, acts as a Modbus TCP or EtherNet/IP server.
- Add the IP address of your maintenance PC to communicate with the PAC through the CPU's Ethernet I/O scanner service via Unity Pro to configure and diagnose your application.

NOTE: The subnet in the **IP Address** column can be the subnet itself or any IP address inside the subnet. If you select **Yes** for a subnet that does not have a subnet mask, a pop-up window states that the screen cannot be validated because of a detected error.

You can enter a maximum of 127 authorized IP addresses or subnets.

Adding Devices to the Authorized Addresses List

To add devices to the **Authorized Addresses** list:

Step	Action
1	Set Access Control to Enabled .
2	In the IP Address column of the Authorized Addresses list, enter an IP address.
3	<p>Enter the address of the device to access the CPU's Ethernet I/O scanner service with either of these methods:</p> <ul style="list-style-type: none"> • <i>Add a single IP address:</i> Enter the IP address of the device and select No in the Subnet column. • <i>Add a subnet:</i> Enter a subnet address in the IP Address column. Select Yes in the Subnet column. Enter a subnet mask in the Subnet Mask column. <p>NOTE:</p> <ul style="list-style-type: none"> • The subnet in the IP Address column can be the subnet itself or any IP address in the subnet. If you enter a subnet without a subnet mask, an on-screen message states that the screen cannot be validated. • A red exclamation point (!) indicates a detected error in the entry. You can save the configuration only after the detected error is addressed.
4	Select one or more of the following methods of access you are granting the device or subnet: FTP, TFTP, HTTP, Port 502, EIP, SNMP .
5	<p>Repeat steps 2 - 4 for each additional device or subnet to which you want to grant access to the CPU's Ethernet I/O scanner service.</p> <p>NOTE: You can enter up to 127 authorized IP addresses or subnets.</p>
6	Click Apply .

Removing Devices from the Authorized Addresses List

To remove devices from the **Authorized Addresses** list:

Step	Action
1	In the Authorized Addresses list, select the IP address of the device to delete.
2	Press the Delete button.
3	Click Apply .

IPConfig Tab

IPConfig Parameters

IP address configuration field on the IPConfig tab:

Parameter	Default Value	Description
Main IP address	192.168.10.1	The IP address of the CPU and DIO scanner. This address can be used: <ul style="list-style-type: none"> • By Unity Pro, an HMI, or SCADA to communicate with the CPU. • To access the CPU web pages. • By the CPU to perform I/O scanning of DIO devices.
IP address A	192.168.11.1	This address applies to the RIO scanner service in the CPU designated as A . (See the note below.)
IP address B	–	For M580 Hot Standby CPUs only, this address applies to the RIO scanner service in the CPU designated as B . (See the note below.)
Subnetwork mask	255.255.0.0	This bit mask identifies or determines the IP address bits that correspond to the network address and the subnetwork portion of the address. (The value can be changed to any valid value in the subnetwork.)
Gateway address	192.168.10.1	This is the IP address of the default gateway to which messages for other networks are transmitted.
NOTE: <ul style="list-style-type: none"> • If you change IP address A, the system may recalculate all IP addresses (including those of the drops) to keep all devices in the same subnetwork. • In M580 Hot Standby systems, both CPU A and CPU B maintain a redundant owner connection with each RIO device (BM•CRA312•0 adapter). For this reason, when a Hot Standby switchover occurs, the state of RIO outputs is not affected – the Hot Standby switchover transition is bumpless. 		

Viewing and Editing the IP Address and Device Name of Network Devices

The **CRA IP address configuration** area on the **IPConfig** tab is provided for CPUs with Ethernet I/O scanner service (CPUs with commercial references that end *40*). Use this area to display a list of RIO/DIO scanners and BM•CRA312•0 adapters, and view or edit the device IP address and device Identifier:

Step	Action
1	Click the CRA IP address configuration link to open the Ethernet Network window.
2	<p>In the Subtype header, filter the device list by selecting:</p> <ul style="list-style-type: none">● Scanner RIO/DIO● CRA● ... (select both) <p>This list applies the selected filter, and displays all detected network devices of the selected type(s.)</p>
3	<p>The IP Address field displays the address that was automatically assigned when the device was added to the network.</p> <p>NOTE: Although the IP address is editable, Schneider Electric recommends that you accept the automatically assigned IP address.</p>
4	<p>The Identifier field displays the identifier for the module, which is also the Device Name. To edit the Identifier setting:</p> <ol style="list-style-type: none">1. Double-click on the Identifier value. The value becomes editable.2. Type in a new value.3. Click the Unity Pro Validate button. <p>The new Identifier setting is applied.</p>

NOTE: All other fields in the **Ethernet Network** window are read-only.

Advanced Configuration

To configure DHCP and FDR services in the DTM browser, click the **Services configuration** link in the **Advanced configuration** section of the window.

RSTP Tab

Introduction

The Ethernet DEVICE NETWORK ports on the front of the M580 CPU support *rapid spanning tree protocol* (RSTP). RSTP is an OSI layer 2 protocol defined by IEEE 802.1D 2004. RSTP performs these services:

- RSTP creates a loop-free logical network path for Ethernet devices that are part of a topology that includes redundant physical paths. When either DEVICE NETWORK port (ETH 2 or ETH 3) on the CPU is disconnected, the RSTP service directs traffic to the other port.
- RSTP automatically restores network communication by activating redundant links when a network event causes a loss of service.

NOTE: When an RSTP link is disconnected, the RSTP service acts on an event and forwards traffic through the correct port. During this re-connect time (50ms max), some packets may be lost.

The RSTP service creates a loop-free logical network path for Ethernet devices that are part of a topology that includes redundant physical paths. When the network experiences a loss of service, the RSTP-enabled module automatically restores network communication by activating redundant links.

NOTE: RSTP can be implemented only when all network switches are configured to support RSTP.

Changing these parameters can affect sub-ring diagnostics, RIO determinism, and network recovery times.

Assign the Bridge Priority for RIO/DIO Scanner Service

A **bridge priority** value is used to establish the relative position of a switch in the RSTP hierarchy. Bridge priority is a 2-byte value for the switch. The valid range is 0 ... 65535, with a default of 32768 (the midpoint).

Assign the **Bridge Priority** on the **RSTP** page:

Step	Action
1	Select RSTP to see the RSTP Operational State .
2	Select a Bridge Priority from the drop-down list in the RSTP Operational State area: <ul style="list-style-type: none"> • Root (0) (default) • Backup Root (4096) • Participant (32768)
3	Finish the configuration: <ul style="list-style-type: none"> • OK: Assign the Bridge Priority, and close the window. • Apply: Assign the Bridge Priority, and keep the window open.

RSTP Parameters for CPUs with RIO and DIO Scanner Service

RSTP tab:

Field	Parameter	Value	Comment
RSTP Operational State	Bridge Priority	Root (0)	default
		Backup Root (4096)	–
		Participant (32768)	–

RSTP Parameters for CPUs without RIO Scanner Service (DIO Scanner Service Only)

RSTP tab:

Field	Parameter	Value	Comment
RSTP Operational State	Bridge Priority	Root(0)	–
		Backup Root(4096)	–
		Participant(32768)	default
Bridge parameters	Force version	2	You cannot edit this value.
	Forward delay (ms)	21000	
	Maximum Age Time (ms)	40000	
	Transmit Hold Count	40	
	Hello Time (ms)	2000	
Port 2 Parameters	–	–	You cannot edit these field parameters.
Port 3 Parameters	–	–	You cannot edit these field parameters.

SNMP Tab

Introduction

Use the **SNMP** tab in Unity Pro to configure individual SNMP parameters for these modules:

- M580 CPU modules
- (e)X80 EIO adapter modules on RIO drops
- 140CRA3120• RIO adapter modules in Quantum EIO systems

An SNMP v1 agent is a software component of the SNMP service that runs on these modules to allow access to diagnostic and management information for the modules. You can use SNMP browsers, network management software, and other tools to access this data. In addition, the SNMP agent can be configured with the IP addresses of one or two devices (typically PCs that run network management software) to be the targets of event-driven trap messages. Such messages inform the management device of events like cold starts and the inability of the software to authenticate a device.

Use the **SNMP** tab to configure the SNMP agents for communication modules in the local rack and RIO drops. The SNMP agent can connect to and communicate with one or two SNMP managers as part of an SNMP service. The SNMP service includes:

- authentication checking by the Ethernet communication module, of any SNMP manager that sends SNMP requests
- management of events or traps

SNMP Parameters

View and edit these properties on the **SNMP** page:

Property		Description
IP Address Managers:	IP Address Manager 1	The IP address of the first SNMP manager to which the SNMP agent sends notices of traps.
	IP Address Manager 2	The IP address of the second SNMP manager to which the SNMP agent sends notices of traps.
Agent:	Location	The device location (32 characters maximum)
	Contact	Information describing the person to contact for device maintenance (32 characters maximum)
	SNMP Manager	Select one: <ul style="list-style-type: none"> • Disabled: You can edit the Location and Contact settings on this page. • Enabled: You cannot edit the Location and Contact settings on this page. (Those settings are managed by the SNMP Manager.)

Property		Description
Community Names:	Get	Password required by the SNMP agent before executing read commands from an SNMP manager (default = public).
	Set	Password required by the SNMP agent before executing write commands from an SNMP manager (default = private).
	Trap	Password an SNMP manager requires from the SNMP agent before the manager accepts trap notices from the agent (default = alert).
Security:	Enable Authentication Failure Trap	TRUE causes the SNMP agent to send a trap notice to the SNMP manager if an unauthorized manager sends a Get or Set command to the agent (default = Disabled).

Apply the configuration by clicking a button:

- **Apply**: Save changes.
- **OK**: Save changes and close the window.

NTP Tab

Introduction

You can configure an M580 CPU as an NTP server or an NTP client in the Unity Pro NTP tab. The NTP service has these features:

- A periodic time correction is obtained from the reference-standard time server.
- There is an automatic switchover to a backup (secondary) time server if an error is detected with the normal time server system.
- Controller projects use a function block to read the accurate clock, allowing project events or variables to be time stamped. (Refer to the *System Time Stamping User Guide (see System Time Stamping, User Guide)* for detailed information about timestamping performance.)

NOTE:

When the M580 CPU is configured as either an NTP server or as an NTP client, the BM•CRA312•0 (e)X80 EIO adapter modules are NTP clients of the CPU:

- When only BM•CRA31200 modules are configured as NTP clients, the accuracy of this server allows time discrimination of 20 ms.
- All BM•CRA31200 modules in the network have the same client configuration.

To begin, open the CPU configuration tabs in Unity Pro ([see page 113](#)).

NTP Client Mode

When the PAC is configured as an NTP client, the network time service (SNTP) synchronizes the clock in the M580 CPU to that of the time server. The synchronized value is used to update the clock in the CPU. Typical time service configurations utilize redundant servers and diverse network paths to achieve high accuracy and reliability.

To establish the accurate Ethernet system network time, the system performs the following at power up:

- requires the CPU to boot
- uses the CPU to obtain the time from the NTP server
- requires a predefined interval until time is accurate; your configuration determines how long before time is accurate
- may require several updates to achieve peak accuracy

Once an accurate time is received, the service sets the status in the associated time service register.

The time service clock value starts at 0 until fully updated from the CPU.

Model	Starting Date
Modicon M580 with Unity Pro	January 1st 1980 00:00:00.00

Stop or run PAC:

- Stop and run have no effect on the accuracy of the clock.
- Stop and run have no effect on the update of the clock.
- A transition from one mode to the other has no effect on the accuracy of the Ethernet system network time.

Download application:

- The status clock value associated with the time service register in the M580 CPU is reinitialized after an application is downloaded or after an NTP server swap. The time is accurate after two polling periods.

NOTE: For NTP diagnostics, refer to the NTP web page.

NTP Server Mode

When the PAC is configured as an NTP server, it can synchronize client clocks (such as a BM•CRA31200 (e)X80 EIO adapter module). The CPU's internal clock is then used as reference clock for NTP services.

NTP Parameters for a CPU

Use the pull-down menu in the **NTP** field to configure the CPU as an **NTP Client** or an **NTP Server**:

Value	Comment
Disabled	default: Both the NTP server and the NTP client services of the PAC are disabled.
NTP Client	<p>The PAC functions as the NTP client. In this case, configure the NTP Server Configuration parameters.</p> <p>NOTE: Enable the NTP client here to automatically enable the NTP client service on all BM•CRA312•0 adapter modules.</p>
NTP Server	<p>The Ethernet I/O scanner PAC acts as an NTP server.</p> <p>NOTE: Enable the NTP client here to automatically enable the NTP client service on all BM•CRA312•0 adapter modules and to configure the BM•CRA312•0 to use the PAC as the NTP server.</p>

Assign values to these parameters in the **NTP Server Configuration** field:

Parameter	Comment
Primary NTP Server IP address	the IP address of the NTP server, from which the PAC first requests a time value
Secondary NTP Server IP address	the IP address of the backup NTP server, from which the PAC requests a time value after not receiving a response from the primary NTP server
Polling Period	The time (in seconds) between updates from the NTP server. Smaller values typically result in better accuracy.

Switch Tab

Description

The **Switch** tab is only available for CPUs without RIO scanner service. It contains these fields:

Field	Parameter	Value	Comment
ETH1	–	–	You cannot edit these field parameters here. Configuration can be modified in the Service Port tab (<i>see page 130</i>).
ETH2	Enabled	Yes	default
		No	–
	Baud Rate	Auto 10/100 Mbits/sec	default
		100 Mbits/sec Half duplex	–
		100 Mbits/sec Full duplex	–
		10 Mbits/sec Half duplex	–
		10 Mbits/sec Full duplex	–
ETH3	Enabled	Yes	default
		No	–
	Baud Rate	Auto 10/100 Mbits/sec	default
		100 Mbits/sec Half duplex	–
		100 Mbits/sec Full duplex	–
		10 Mbits/sec Half duplex	–
		10 Mbits/sec Full duplex	–
Backplane	–	–	You cannot edit these field parameters.

NOTE: ETH1 port is a dedicated service port and the Ethernet backplane network is dedicated to the communication between modules on the rack. The switch parameters for those two ports cannot be configured in the **Switch** tab.

QoS Tab

Description

The M580 CPU can be configured to perform Ethernet packet tagging. The CPU supports the OSI layer 3 quality of service (QoS) standard defined in RFC-2475. When you enable QoS, the CPU adds a *differentiated services code point* (DSCP) tag to each Ethernet packet that it transmits to indicate the priority of that packet.

QoS Tab

The **QoS** tab is available only on CPUs that do not support the RIO scanner service (only on CPUs with commercial references that end with 20).

Field	Parameter	Value	Comment
DSCP Tagging	–	Enabled	default
		Disabled	–
PTP	DSCP PTP Event Priority	59	–
	DSCP PTP General Priority	47	–
EtherNet/IP Traffic	DSCP Value For I/O Data Schedule Priority Messages	47	–
	DSCP Value For Explicit Message	27	–
	DSCP Value For I/O Data Urgent Priority Messages	55	–
	DSCP Value For I/O Data High Priority Messages	43	–
	DSCP Value For I/O Data Low Priority Messages	31	–
Modbus TCP Traffic	DSCP Value For I/O Messages	43	–
	DSCP Value For Explicit Message	27	–
Network Time Protocol Traffic	DSCP Value For Network Time Protocol Messages	59	–

DSCP tagging lets you prioritize the Ethernet packet streams based on the type of traffic in that stream.

To implement QoS settings in your Ethernet network:

- Use network switches that support QoS.
- Consistently apply DSCP values to network devices and switches that support DSCP.
- Confirm that switches apply a consistent set of rules for sorting DSCP tags, when transmitting and receiving Ethernet packets.

Service Port Tab

Service Port Parameters

These parameters are on the Unity Pro **Service Port** tab:

Field	Parameter	Value	Comment
Service Port	–	Enabled (default)	Enable the port and edit port parameters.
	–	Disabled	Disable the port (no access to parameters).
Service Port Mode	–	Access (default)	This mode supports communications to Ethernet devices.
	–	Mirroring	In port mirroring mode, data traffic from one or more of the other ports is copied to this port. Connect a packet sniffing tool to this port to monitor and analyze port traffic. NOTE: In this mode, the Service port acts like a read-only port. That is, you cannot access devices (ping, connection to Unity Pro, and so on) through the Service port.
Access Port Configuration	Service Port Number	ETH1	You cannot edit the value in the Service Port Number field.
Port Mirroring Configuration	Source Port(s)	Internal Port	Ethernet traffic to and from the internal processor sent to the Service Port
		ETH2	Ethernet traffic to and from ETH2 sent to the Service Port
		ETH3	Ethernet traffic to and from ETH3 sent to the Service Port
		Backplane Port	Ethernet traffic to and from the backplane sent to the Service Port

On-line Behavior

The **Service Port** parameters are stored in the application, but you can reconfigure (change) the parameters in connected mode. Values that you reconfigure in connected mode are sent to the PAC through explicit messaging.

The changed values are not stored, so a mismatch can exist between the parameters that are being used and those that are in the stored application.

Advanced Settings Tab

Introduction

The **Advanced Settings** tab is only available for CPUs that do not support RIO scanning (DIO scanner service only). The **Advanced Settings** contains these fields:

- **EtherNet/IP Timeout Settings**
- **EtherNet/IP Scanner Behavior**

Timeout Settings

These parameters are in the **EtherNet/IP Timeout Settings** field:

Parameter	Value	Comment
FW_Open I/O Connection Timeout (msec)	4960	Specifies the amount of time the scanner waits for FW_Open response of an I/O connection.
FW_Open EM Connection Timeout (msec)	3000	Specifies the amount of time the scanner waits for FW_Open response of an EM connection.
EM Connection RPI (msec)	10000	Sets T->O and O->T RPI for all EM connections.
EM Request Timeout (sec)	10	Specifies the amount of time the scanner will wait between the request and the response of an explicit message.

Scanner Behavior

These parameters are in the **EtherNet/IP Scanner Behavior** field:

Parameter	Value	Comment
Allow RESET via explicit message	Disabled	(Default.) The scanner ignores the Identity object reset service request.
	Enabled	The scanner will reset if an Identity object reset service request is received.
Behavior when CPU state is STOP	Idle	(Default.) The EtherNet/IP I/O connection stays open, but the Run/Idle flag is set to Idle.
	STOP	The EtherNet/IP IO connection is closed.

Section 5.3

Configuring the M580 CPU with DTMs in Unity Pro

Introduction

Some configuration features for the M580 CPU are accessed through its corresponding M580 DTM in the Unity Pro **DTM Browser**.

Use the instructions in this section to configure the M580 CPU through the DTM.

What Is in This Section?

This section contains the following topics:

Topic	Page
About DTM Configuration in Unity Pro	133
Accessing Channel Properties	134
Configuring DHCP and FDR Address Servers	136

About DTM Configuration in Unity Pro

Introduction

The configuration of the M580 CPU through standard Unity Pro features is described elsewhere in this guide (*see page 112*).

Some configuration that is specific to a particular device (like the M580 CPU) is done through a corresponding device type manager (DTM) in Unity Pro. This section describes that configuration.

Accessing Configuration Settings

Follow these steps to access the configuration settings in the DTM for the M580 CPU in Unity Pro:

Step	Action
1	Open Unity Pro.
2	Open a Unity Pro project that includes a M580 CPU in the configuration.
3	Open the DTM Browser (Tools → DTM Browser).
4	Double-click the DTM that corresponds to the M580 CPU in the DTM Browser to open the device editor of the DTM.
5	These headings appear in the configuration tree of the M580 DTM: <ul style="list-style-type: none">● Channel Properties● Services● EtherNet/IP Local Slaves● Device List● Logging

Accessing Channel Properties

Introduction

On the Unity Pro **Channel Properties** page, you can select a **Source IP Address** (PC) from a pull-down menu.

The **Source IP Address** (PC) menu is a list of IP addresses that are configured for a PC that has the Unity Pro DTM installed.

To make the connection, choose a **Source IP Address** (PC) that is in the same network as the CPU and the device network.

You can execute these tasks through this connection:

- Perform fieldbus discovery.
- Execute Online Actions.
- Send an explicit message to an EtherNet/IP device.
- Send an explicit message to a Modbus TCP device.
- Diagnose modules.

Open the Page

View the **Channel Properties** for the CPU:

Step	Action
1	Open a Unity Pro project that includes a M580 CPU.
2	Open the DTM Browser (Tools → DTM Browser).
3	In the DTM Browser , find the name that you assigned to the CPU.
4	Double-click (or right-click Open) the name of the CPU to open the configuration window.
5	Select Channel Properties in the navigation pane.

Property Descriptions

This table describes the parameters for the **Channel Properties**:

Field	Parameter	Description
Source Address	Source IP Address (PC)	A list of IP addresses assigned to network interface cards installed on your PC. NOTE: If the configured main IP address of the CPU is not in the subnet of any of the IP configured on the interface cards of the PC, then the first interface card IP is suggested by default.
	Sub-Network Mask (read-only)	The subnet mask that is associated with the selected source IP address (PC).

Field	Parameter	Description
EtherNet/IP Network Detection	Begin detection range address	The first IP address in the address range for automatic field bus discovery of EtherNet/IP devices.
	End detection range address	The last IP address in the address range for automatic field bus discovery of EtherNet/IP devices.
Modbus Network Detection	Begin detection range address	The first IP address in the address range for automatic field bus discovery of Modbus TCP devices.
	End detection range address	The last IP address in the address range for automatic field bus discovery of Modbus TCP devices.

Make the Connection

Connect to the **Source IP Address (PC)**:

Step	Action
1	Select an IP address from the Source IP Address (PC) pull-down menu.
2	Press the Apply button.
3	In the DTM Browser , find the name that you assigned to the CPU.
4	Right-click on the name of the CPU and scroll to Connect .

TCP/IP Monitoring

Expand (+) the **Channel Properties** heading in the configuration tree and select the **TCP/IP** item at level 1.

The read-only information on this page monitors the IP parameters that were configured in Unity Pro.

Managing Source IP Addresses for Multiple PCs

When you connect a PC to a DTM-based Unity Pro application, Unity requires that you define the IP address of the PC connected to the PLC, which is referred to as the *source IP address (PC)*. Rather than having to perform a **Build** in Unity each time you connect a PC to the PLC, the source IP address (PC) is selected automatically when you import the Unity application. During application import, the DTM retrieves all available configured NIC addresses of a connected PC and matches the subnet mask of the master with the available NIC list.

- If a match between the subnet mask of the master and the NIC list exists, Unity automatically selects the matched IP address as the *source IP address (PC)* in the **Channel Properties** page.
- If multiple matches exist, Unity automatically selects the IP address nearest to the subnet mask.
- If no match exists, Unity automatically selects the IP address to the nearest available subnet mask.

Configuring DHCP and FDR Address Servers

DHCP and FDR Address Servers

The M580 CPU includes both a dynamic host communication protocol (DHCP) and a fast device replacement (FDR) server. The DHCP server provides IP address settings to networked up to devices. The FDR server provides operating parameter settings to replacement Ethernet devices that are equipped with FDR client functionality.

Accessing the Address Server

Access the address server for the M580 CPU in Unity Pro:

Step	Action
1	Open Unity Pro.
2	Open a Unity Pro project that includes a M580 CPU in the configuration.
3	Open the DTM Browser (Tools → DTM Browser).
4	Double-click the DTM that corresponds to the M580 CPU in the DTM Browser to open the device editor of the DTM.
5	Expand (+) the Services heading in the configuration tree.
6	Select the Address Server item in the configuration tree to see the address server configuration.

Configuration

Configure the address server to perform these tasks:

- Enable and disable the CPU FDR service.
- View an automatically generated list of all devices included in the CPU configuration, displaying for each device:
 - IP addressing parameters
 - whether the device IP addressing parameters are provided by the CPU embedded DHCP server

Manually add remote devices that are not part of the CPU configuration to the CPU DHCP client list.

NOTE: Remote devices added in this way are equipped with DHCP client software and are configured to subscribe to the CPU IP addressing service.

Enabling the FDR Service

To enable the FDR service, set the **FDR Server** field to **Enabled**. To disable the service, toggle the same field to **Disabled**.

You can disable the FDR service for CPUs that do not support RIO scanning (commercial references that end in 20). The FDR service is always enabled for CPUs that support RIO scanning (commercial references that end in 40).

Any networked Ethernet device equipped with FDR client functionality can subscribe to the CPU FDR service.

The maximum size of the FDR client operating parameter files depends on the CPU reference. When this capacity is reached, the CPU cannot store additional client FDR files

CPU Reference	PRM File Size	Concurrent Connections
BMEP581020	8 MB	64
BMEP582020	16 MB	128
BMEP582040	17 MB	136
BMEP583020	16 MB	128
BMEP583040	25 MB	208
BMEP584020	16 MB	128
BMEP584040	25 MB	208
BMEP585040	25 MB	208
BMEP586040	25 MB	208
BMEH582040	25 MB	208
BMEH584040	25 MB	208
BMEH586040	25 MB	208

NOTE: The FDR usage percentage is monitored by the FDR_USAGE variable in the DDDT (*see page 212*).

Viewing the Auto-Generated DHCP Client List

The list of **Automatically Added Devices** includes a row for each remote device that is:

- part of the CPU configuration
- configured to subscribe to the CPU DHCP addressing service

NOTE: You cannot add devices to this list in this page. Instead, use the configuration pages for the remote device to subscribe to this service.

This table describes the available properties:

Property	Description
Device No	The number assigned to the device in the Unity Pro configuration.
IP Address	The client device IP address.
DHCP	TRUE indicates that the device subscribes to the DHCP service.
Identifier Type	Indicates the mechanism used by the server to recognize the client (MAC address or DHCP device name).
Identifier	The actual MAC address or DHCP device name.

Property	Description
Netmask	The client device subnet mask.
Gateway	A DHCP client device uses the gateway IP address to access other devices that are not located on the local subnet. A value of 0.0.0.0 constrains the DHCP client device by allowing it to communicate only with devices on the local subnet.

Manually Adding Remote Modules to the DHCP Service

Remote modules that are part of the CPU configuration – and which have subscribed to the CPU IP addressing service – automatically appear in the **Automatically Added Devices** list.

Other remote modules that are not part of the CPU configuration can be manually added to the CPU DHCP IP addressing service.

Manually add networked Ethernet modules that are not part of the CPU configuration to the CPU IP addressing service:

Step	Description										
1	In the Address Server page, click the Add button in the Manually Added Devices field to instruct Unity Pro to add an empty row to the list.										
2	<div>In the new row, configure these parameters for the client device:</div> <table> <tr> <td>IP Address</td><td>Type in the IP address of the client device.</td></tr> <tr> <td>Identifier Type</td><td>Select the type of value the client device uses to identify itself to the FDR server: <ul style="list-style-type: none"> ● MAC address ● device Name </td></tr> <tr> <td>Identifier</td><td>Depending upon the identifier type, type in the client device setting for the MAC address or name.</td></tr> <tr> <td>Netmask</td><td>Type in the client device subnet mask.</td></tr> <tr> <td>Gateway</td><td>Type in the gateway address that remote devices can use to communicate with devices located on other networks. Use 0.0.0.0 if remote devices do not communicate with devices located on other networks.</td></tr> </table>	IP Address	Type in the IP address of the client device.	Identifier Type	Select the type of value the client device uses to identify itself to the FDR server: <ul style="list-style-type: none"> ● MAC address ● device Name 	Identifier	Depending upon the identifier type, type in the client device setting for the MAC address or name.	Netmask	Type in the client device subnet mask.	Gateway	Type in the gateway address that remote devices can use to communicate with devices located on other networks. Use 0.0.0.0 if remote devices do not communicate with devices located on other networks.
IP Address	Type in the IP address of the client device.										
Identifier Type	Select the type of value the client device uses to identify itself to the FDR server: <ul style="list-style-type: none"> ● MAC address ● device Name 										
Identifier	Depending upon the identifier type, type in the client device setting for the MAC address or name.										
Netmask	Type in the client device subnet mask.										
Gateway	Type in the gateway address that remote devices can use to communicate with devices located on other networks. Use 0.0.0.0 if remote devices do not communicate with devices located on other networks.										
3	Refer to the topic Configuring Properties in the Device Editor (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>) for instructions on how to apply edited properties to networked devices.										

Section 5.4

Diagnostics through the Unity Pro DTM Browser

What Is in This Section?

This section contains the following topics:

Topic	Page
Introducing Diagnostics in the Unity Pro DTM	140
Bandwidth Diagnostics	142
RSTP Diagnostics	144
Network Time Service Diagnostics	146
Local Slave / Connection Diagnostics	148
Local Slave or Connection I/O Value Diagnostics	152
Logging DTM Events to a Unity Pro Logging Screen	153
Logging DTM and Module Events to the SYSLOG Server	154

Introducing Diagnostics in the Unity Pro DTM

Introduction

The Unity Pro DTM provides diagnostics information that is collected at configured polling intervals. Use this information to diagnose the operation of the embedded Ethernet scanner service in the CPU.

Connect the DTM

Before you can open the diagnostics page, make the connection between the DTM for the CPU's embedded scanner service:

Step	Action
1	Open a Unity Pro project.
2	Open the Unity Pro DTM Browser (Tools → DTM Browser).
3	Right-click the name that is assigned to your CPU in the DTM Browser .
4	Select Connect .

Open the Page

Access the **Diagnosis** information:

Step	Action
1	Right-click the name that is assigned to your CPU in the DTM Browser .
2	Select Device Menu → Diagnosis to view the available diagnostics pages.



Diagnostics Information

The diagnostics window has two distinct areas:

- left pane: LED icons indicate the operating status of modules, devices, and connections.
- right pane: These pages show diagnostics data for these items:
 - CPU's embedded scanner service
 - local slave nodes that are activated for the CPU's embedded scanner service
 - EtherNet/IP connections between the CPU's embedded scanner service and a remote EtherNet/IP device

When the appropriate DTM is connected to the CPU, Unity Pro sends an explicit message request once per second to detect the state of the CPU's embedded scanner service and of all the remote devices and EtherNet/IP connections linked to the CPU.

Unity Pro places one of these status icons over the module, device, or connection in the left pane of the **Diagnostic** window to indicate its current status:

Icon	Communication module	Connection to a remote device
	Run state is indicated.	The health bit for every EtherNet/IP connection and Modbus TCP request (to a remote device, sub-device, or module) is set to active (1).
	One of these states is indicated: <ul style="list-style-type: none">● unknown● stopped● not connected	The health bit for at least one EtherNet/IP connection or Modbus TCP request (to a remote device, sub-device, or module) is set to inactive (0).

Bandwidth Diagnostics

Introduction

Use the **Bandwidth** page to view the dynamic and static data for the bandwidth use by the embedded Ethernet scanner service in the CPU.

NOTE: Before you can open the diagnostics page, make the connection between the DTM for the CPU's embedded scanner service and the physical module.

Open the Page

Access the **Bandwidth** information:

Step	Action
1	In the DTM Browser , right-click the name that is assigned to your CPU.
2	Select Device menu → Diagnosis .
3	In the left pane of the Diagnosis window, select the CPU node.
4	Select the Bandwidth tab to open that page.

Data Display

Use the **Refresh Every 500ms** checkbox to display the static or dynamic data:

Checkbox	Description
Selected	<ul style="list-style-type: none">• Display data that is dynamically updated every 500 ms.• Increment the number at the top of the table each time data is refreshed.
De-selected	<ul style="list-style-type: none">• Display static data.• Do not increment the number at the top of the table. That number now represents a constant value.

Bandwidth Diagnostic Parameters

The **Bandwidth** page displays the following parameters for the communication module:

Parameter	Description
I/O - Scanner:	
EtherNet/IP Sent	The number of EtherNet/IP packets the module has sent in packets/second.
EtherNet/IP Received	The number of EtherNet/IP packets the module has received in packets/second.
Modbus TCP Received	The number of Modbus TCP requests the module has sent in packets/second.
Modbus TCP Responses	The number of Modbus TCP responses that the CPU's embedded scanner service has received in packets/second.
I/O - Adapter:	
EtherNet/IP Sent	The number of EtherNet/IP packets (per second) that the CPU's embedded scanner service has sent in the role of a local slave.
EtherNet/IP Received	The number of EtherNet/IP packets (per second) that the CPU's embedded scanner service has received in the role of a local slave.
I/O - Module	
Module Capacity	The maximum number of packets (per second) that the CPU's embedded scanner service can process.
Module Utilization	The percentage of the CPU's embedded scanner service capacity being used by the application.
Messaging - Client:	
EtherNet/IP Activity	The number of explicit messages (packets per second) sent by the CPU's embedded scanner service using the EtherNet/IP protocol.
Modbus TCP Activity	The number of explicit messages (packets per second) sent by the CPU's embedded scanner service using the Modbus TCP protocol.
Messaging - Server:	
EtherNet/IP Activity	The number of server messages (packets per second) received by the CPU's embedded scanner service using the EtherNet/IP protocol.
Modbus TCP Activity	The number of server messages (packets per second) received by the CPU's embedded scanner service using the Modbus TCP protocol.
Module:	
Processor Utilization	The percentage of the CPU's embedded scanner service processing capacity used by the present level of communication activity.

RSTP Diagnostics

Introduction

Use the **RSTP Diagnostic** page to view the status of the RSTP service of the embedded Ethernet scanner service in the CPU. The page displays dynamically generated and static data for the module.

NOTE: Before you can open the diagnostics page, make the connection between the DTM for the CPU's embedded scanner service and the physical module.

Open the Page

Access the **RSTP Diagnosis** information:

Step	Action
1	In the DTM Browser , right-click the name that is assigned to your CPU.
2	Select Device menu → Diagnosis .
3	In the left pane of the Diagnosis window, select the CPU node.
4	Select RSTP Diagnostic tab to open that page.

Data Display

Select the **Refresh Every 500ms** check box to display the static or dynamic data:

Checkbox	Description
Selected	<ul style="list-style-type: none">● Display data that is dynamically updated every 500 ms.● Increment the number at the top of the table each time data is refreshed.
De-selected	<ul style="list-style-type: none">● Display static data.● Do not increment the number at the top of the table. That number now represents a constant value.

RSTP Diagnostic Parameters

The **RSTP Diagnostic** page displays the following parameters for each CPU port:

Parameter	Description
Bridge RSTP Diagnostic:	
Bridge Priority	This 8-byte field contains the two-byte value that is assigned to the CPU's embedded Ethernet switch.
MAC Address	The Ethernet address of the CPU, found on the front of the CPU.
Designated Root ID	The Bridge ID of the root device.
Root Path Cost	The aggregate cost of port costs from this switch back to the root device.
Default Hello Time	The interval at which Configuration BPDU messages are transmitted during a network convergence. For RSTP this is a fixed value of 2 seconds.
Learned Hello Time	The current Hello Time value learned from the root switch.
Configured Max Age	The value (6 ... 40) that other switches use for MaxAge when this switch is acting as the root.
Learned Max Age	The maximum age learned from the root switch. This is the actual value currently used by this switch.
Total Topology Changes	The total number of topology changes detected by this switch since the management entity was last reset or initialized.
Ports ETH 2 and ETH 3 RSTP Statistics:	
Status	The port's current state as defined by RSTP protocol. This state controls the action the port takes when it receives a frame. Possible values are: disabled, discarding, learning, forwarding.
Role:	The port's current role per RSTP protocol. Possible values are: root port, designated port, alternate port, backup port, disabled port.
Cost	The logical cost of this port as a path to the root switch. If this port is configured for AUTO then the cost is determined based on the connection speed of the port.
STP Packets	<p>A value in this field indicates that a device on the network has the STP protocol enabled.</p> <p>NOTE:</p> <ul style="list-style-type: none"> Other devices that are enabled for STP can severely affect the network convergence times. Schneider Electric recommends that you disable the STP protocol (but not the RSTP protocol) on every network device that supports STP. The CPU does not support the STP protocol. The CPU's embedded switch ignores STP packets.

Network Time Service Diagnostics

Introduction

Use the **Network Time Service Diagnostic** page to display dynamically generated data describing the operation of the simple network time protocol (SNTP) service that you configured in the network time server page (*see page 125*) in Unity Pro.

NOTE: Before you can open the diagnostics page, make the connection between the DTM for the target communication module and the CPU.

Refer to the *System Time Stamping User Guide (see System Time Stamping, User Guide)* for detailed diagnostic information.

Open the Page

Access the **NTP Diagnostic** information:

Step	Action
1	In the DTM Browser , find the name that is assigned to the CPU.
2	Right-click the CPU DTM, and select Device menu → Diagnosis .
3	In the left pane of the Diagnosis window, select the CPU node.
4	Select the NTP Diagnostic tab to open that page.

Click the **Reset Counter** button to reset the counting statistics on this page to 0.

Network Time Service Diagnostic Parameters

This table describes the time synchronization service parameters:

Parameter	Description
Refresh Every 500ms	Check this box to dynamically update the page every 500ms. The number of times this page has been refreshed appears immediately to the right.
Network Time Service	Monitor the operational status of the service in the module: <ul style="list-style-type: none">● <i>green</i>: operational● <i>orange</i>: disabled
Network Time Server Status	Monitor the communication status of the NTP server: <ul style="list-style-type: none">● <i>green</i>: The NTP server is reachable.● <i>red</i>: The NTP server is not reachable.
Last Update	Elapsed time, in seconds, since the most recent NTP server update.
Current Date	System date
Current Time	The system time is presented in the <i>hh:mm:ss</i> format.

Parameter	Description
DST Status	Set the status of the automatic daylight savings service: <ul style="list-style-type: none"> ● <i>ON</i>: The automatic adjustment of daylight savings is enabled. The current date and time reflect the daylight savings time adjustment. ● <i>OFF</i>: The automatic adjustment of daylight savings is disabled. (The current date and time may not reflect the daylight savings time adjustment.)
Quality	This correction (in seconds) applies to the local counter at every NTP server update. Numbers greater than 0 indicate increasingly excessive traffic condition or an NTP server overload.
Requests	This value represents the total number of client requests sent to the NTP server.
Responses	This value represents the total number of server responses sent from the NTP server.
Errors	This value represents the total number of unanswered NTP requests.
Last Error	This value indicates the last detected error code received from the NTP client: <ul style="list-style-type: none"> ● 0: good NTP configuration ● 1: late NTP server response (can be caused by excessive network traffic or server overload) ● 2: NTP not configured ● 3: invalid NTP parameter setting ● 4: NTP component disabled ● 5: NTP server is not synchronized (NTP server needs to be synchronized so that the NTP accesses behave as defined in the client NTP settings) ● 7: unrecoverable NTP transmission ● 9: invalid NTP server IP address ● 15: invalid syntax in the custom time zone rules file
Primary / Secondary NTP Server IP	The IP addresses correspond to the primary and secondary NTP servers. NOTE: A green LED to the right of the primary or secondary NTP server IP address indicates the active server.
Auto Adjust Clock for Daylight Savings	Configure the daylight savings adjustment service: <ul style="list-style-type: none"> ● enabled ● disabled
DST Start / DST End	Specify the day on which daylight savings time begins and ends:
	Month Set the month in which daylight savings time starts or ends.
	Day of Week Set the day of the week on which daylight savings time starts or ends.
	Week# Set the occurrence of the specified day within the specified month.
Time Zone	Select the time zone plus or minus Universal Time, Coordinated (UTC)
Offset	Configure the time (in minutes) to be combined with the time zone selection (above) to produce the system time.
Polling Period	Set the frequency with which the NTP client requests an updated time from the NTP server

Local Slave / Connection Diagnostics

Introduction

Use the **Local Slave Diagnostic** page and the **Connection Diagnostic** page to display the I/O status and production/consumption information for a selected local slave or connection.

NOTE:

- Before you can open the diagnostics page, make the connection between the DTM for the target communication module and the CPU.
- To get data from the primary CPU, make the connection to the Main IP address of the CPU
(see *Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*).

Open the Page

Access the diagnostics information:

Step	Action
1	In the DTM Browser , find the name that is assigned to the CPU.
2	Right-click the CPU DTM, and select Device menu → Diagnosis .
3	In the left pane of the Diagnosis window, select the CPU node.
4	Select the Local Slave Diagnostic tab or the Connection Diagnostic tab to open that page.

Data Display

Use the **Refresh Every 500ms** checkbox to display the static or dynamic data:

Checkbox	Description
Selected	<ul style="list-style-type: none">• Display data that is dynamically updated every 500 ms.• Increment the number at the top of the table each time data is refreshed.
De-selected	<ul style="list-style-type: none">• Display static data.• Do not increment the number at the top of the table. That number now represents a constant value.

Local Slave / Connection Diagnostic Parameters

The following tables display the diagnostic parameters for the selected local slave or scanner connection.

This table shows the **Status** diagnostic parameters for the selected connection:

Parameter	Description
Input	An integer representing input status.
Output	An integer representing output status.
General	An integer representing basic connection status.
Extended	An integer representing extended connection status.

The **Input** and **Output** status diagnostic parameters can present these values:

Input/Output Status (dec)	Description
0	OK
33	Time-out
53	IDLE
54	Connection established
58	Not connected (TCP)
65	Not connected (CIP)
68	Connection establishing
70	Not connected (EPIC)
77	Scanner stopped

This table shows the **Counter** diagnostic parameters for the selected connection:

Parameter	Description
Frame Error	Increments each time a frame is not sent by missing resources or is impossible to send.
Time-Out	Increments each time a connection times out.
Refused	Increments when connection is refused by the remote station.
Production	Increments each time a message is produced.
Consumption	Increments each time a message is consumed.
Production Byte	Total of produced messages, in bytes, since the communication module was last reset.
Consumption Byte	Total of consumed messages, in bytes, since the communication module was last reset.
Theoretical Packets per second	Packets per second calculated using current configuration value.
Real Packets per second	Actual number of packets per second generated by this connection.

This table shows the **Diagnostic** parameters for the selected connection:

Parameter	Description
CIP Status	An integer representing CIP status.
Extended Status	An integer representing extended CIP status.
Production Connection ID	The connection ID for the data produced by the local slave.
Consumption Connection ID	The connection ID for the data produced by the local slave.
O -> T API	Actual packet interval (API) of the production connection.
T -> O API	Actual packet interval (API) of the consumption connection.
O -> T RPI	Requested packet interval (RPI) of the production connection.
T -> O RPI	Requested packet interval (RPI) of the consumption connection.

This table shows the **Socket Diagnostics** diagnostic parameters for the selected connection:

Parameter	Description
Socket ID	Internal identification of the socket.
Remote IP Address	IP address of the remote station for this connection.
Remote Port	UDP port number of the remote station for this connection.
Local IP Address	IP address of the communication module for this connection.
Local Port	UDP port number of the communication module for this connection.

This table shows the **Production** diagnostic parameters for the selected connection:

Parameter	Description
Sequence Number	The number of the sequence in the production.
Max Time	Maximum time between two produced messages.
Min Time	Minimum time between two produced messages.
RPI	Current production time.
Overrun	Increments each time a produced message exceeds RPI.
Underrun	Increments each time a produced message is less than RPI.

This table shows the **Consumption** diagnostic parameters for the selected connection:

Parameter	Description
Sequence Number	The number of the sequence in the consumption.
Max Time	Maximum time between two consumption messages.
Min Time	Minimum time between two consumption messages.
RPI	Current consumption time.
Over Run	Increments each time a consumed message exceeds RPI.
Under Run	Increments each time a consumed message is less than RPI.

Local Slave or Connection I/O Value Diagnostics

Introduction

Use the **I/O Values** page to display both the input data image and output data image for the selected local slave or scanner connection.

NOTE: Before you can open the diagnostics page, make the connection (*see page 309*) between the DTM and the target communication module.

Open the Page

Access the **I/O Values** information:

Step	Action
1	In the DTM Browser , find the name that is assigned to the CPU DTM.
2	Right-click the CPU DTM , and select Device menu → Diagnosis .
3	In the left pane of the Diagnosis window, select the CPU.
4	Select the I/O Values tab.

Data Display

Use the **Refresh Every 500ms** checkbox to display the static or dynamic data:

Checkbox	Description
Selected	<ul style="list-style-type: none"> Display data that is dynamically updated every 500 ms. Increment the number at the top of the table each time data is refreshed.
De-selected	<ul style="list-style-type: none"> Display static data. Do not increment the number at the top of the table. That number now represents a constant value.

Local Slave / Scanner Connection I/O Values

This page displays theses parameters for either a local slave or a remote device connection input and output values:

Parameter	Description
Input/Output data display	A display of the local slave or remote device input or output data image.
Length	The number of bytes in the input or output data image.
Status	The Scanner Diagnostic object's status, with respect to the read of the input or output data image.

Logging DTM Events to a Unity Pro Logging Screen

Description

Unity Pro maintains a log of events for:

- the Unity Pro embedded FDT container
- each Ethernet communication module DTM
- each EtherNet/IP remote device DTM

Events relating to the Unity Pro FDT container are displayed in the **FDT log event** page of the **Output Window**.

Events relating to a communication module or remote EtherNet/IP device are displayed:

- in configuration mode: in the **Device Editor**, by selecting the **Logging** node in the left pane
- in diagnostic mode: in the **Diagnostics** window, by selecting the **Logging** node in the left pane

Logging Attributes

The **Logging** window displays the result of an operation or function performed by Unity Pro. Each log entry includes the following attributes:

Attribute	Description
Date/Time	The time the event occurred, displayed in the format: yyyy-mm--dd hh:mm:ss
Log Level	The level of event importance. Values include:
	Information A successfully completed operation.
	Warning An operation that Unity Pro completed, but which may lead to a subsequent error.
	Error An operation that Unity Pro was unable to complete.
Message	A brief description of the core meaning of the event.
Detail Message	A more detailed description of the event, which may include parameter names, location paths, etc.

Accessing the Logging Screen

In Unity Pro:

Step	Action
1	Open a project that includes a BME •58 •0•0 Ethernet CPU.
2	Click Tools → DTM Browser to open the DTM Browser .
3	In the DTM Browser , double-click the CPU (or right-click Open) to open the configuration window.
4	Select Logging in the navigation tree in the left pane of the window.

Logging DTM and Module Events to the SYSLOG Server

Configuring the SYSLOG Server

Configure the SYSLOG server address for logging DTM and module events:

Step	Action
1	In Unity Pro, select Tools → Project Settings .
2	In the left pane of the Project Settings window, select Project Settings → General → PLC diagnostics .
3	<p>In the right pane:</p> <ul style="list-style-type: none"> ● Select the PLC event logging check box. ● In the SYSLOG server address field enter the IP address of the SYSLOG server. ● In the SYSLOG server port number field, enter the port number. <p>NOTE: The SYSLOG server protocol is not configurable, and is set to tcp by default.</p>

NOTE: Refer to the *Modicon Controllers Platform Cyber Security Reference Manual* for information on setting up a SYSLOG server in your system architecture (see *Modicon Controllers Platform, Cyber Security, Reference Manual*).

DTM Events Logged to the SYSLOG Server

These DTM events are logged to the SYSLOG server:

- Configuration parameter change
- Add/Delete device
- Rebuild All
- Build Changes
- Renaming of I/O variables
- Add/Modify tasks

BME•58•0•0 CPU Events Logged to the SYSLOG Server

These BME•58•0•0 CPU events are logged to the SYSLOG server:

- TCP connection error due to Access Control List
- Enable/Disable of communication services outside configuration
- Ethernet port link up/down events
- RSTP topology change
- Program operating mode change of COMs (RUN, STOP, INIT)
- Successful and unsuccessful FTP login

Section 5.5

Online Action

What Is in This Section?

This section contains the following topics:

Topic	Page
Online Action	156
EtherNet/IP Objects Tab	158
Service Port Tab	159
Pinging a Network Device	160

Online Action

Introduction

You can view and configure the settings in the **Online Action** menu when the M580 CPU is connected through the Unity Pro **DTM Browser**.

Accessing Online Action

Follow these directions to access the **Online Action** settings for the M580 CPU:

Step	Action
1	Open the DTM Browser in Unity Pro (Tools → DTM Browser).
2	Select the M580 DTM in the DTM Browser .
3	Connect the DTM to the Unity Pro application (Edit → Connect).
4	Right-click the M580 DTM.
5	Scroll to the Online Action menu (Device menu → Additional functions → Online Action).
6	3 tabs appear: <ul style="list-style-type: none">● Ethernet/IP Objects● Port Configuration● Ping

EtherNet/IP Objects

Displays object parameters value when available.

Click **Refresh** to update the displayed values.

Port Configuration

Configure and read the service port mode:

Field	Description
Service Port Mode	<ul style="list-style-type: none">● Access (default)● Mirroring <p>NOTE: This mode can also be set in the CPU configuration tabs (<i>see page 130</i>).</p>
Access Port Configuration	Displays the access port configuration information (refer to CPU configuration tabs (<i>see page 130</i>)).
Port Mirroring Configuration	Displays the port mirroring configuration (refer to CPU configuration tabs (<i>see page 130</i>)).

Ping

Field	Parameter	Description
Address	IP Address	Type the IP address to ping.
Ping	Ping	Click to ping the address set.
	Ping Result	Displays the ping result.
	Repeat (100ms)	Select this parameter to repeat ping if no reply is received.
	Stop on Error	Select this parameter to stop repeating ping if an error is detected when Repeat (100ms) is selected.
	Clear	Click to clear the Ping Result display.

EtherNet/IP Objects Tab

Introduction

Use the **EtherNet/IP Objects** tab in the **Online Action** window:

- Retrieve and display current data describing the state of CIP objects for the selected CPU or remote EtherNet/IP device.
- Reset the selected CPU or remote EtherNet/IP device.

Access the Page

Open the **EtherNet/IP Objects** tab:

Step	Action
1	Connect the DTM to the module (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
2	Open the Online Action page (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
3	Select the EtherNet/IP Objects tab.

Available CIP Objects

You can retrieve CIP objects according to the Unity Pro operating mode:

Mode	Available CIP Objects
Standard	Identity object (<i>see page 167</i>)
Advanced	Identity object (<i>see page 167</i>)
	Connection Manager object (<i>see page 171</i>)
	TCP/IP Interface object (<i>see page 177</i>)
	Ethernet Link object (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>)
	QoS object (<i>see page 175</i>)

Service Port Tab

Introduction

Use the **Service Port** tab in the **Online Action** window to view and edit communication port properties for a distributed EtherNet/IP device. Use this tab to execute these commands:

- *Refresh*: Use a Get command to retrieve port configuration settings from a distributed EtherNet/IP device.
- *Update*: Use a Set command to write all or selected edited values to the same distributed EtherNet/IP device

The configuration information on the **Service Port** tab is sent in EtherNet/IP explicit messages that employ the address and messaging settings configured for Ethernet/IP explicit messaging (below).

Access the Page

Open the **EtherNet/IP Objects** tab:

Step	Action
1	Connect the DTM to the module (see <i>Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
2	Open the Online Action page (see <i>Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
3	Select the EtherNet/IP Objects tab.
4	Configure the Service port with the instructions from the offline configuration (see <i>Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
5	Click the Update button to apply the new configuration.

Pinging a Network Device

Overview

Use the Unity Pro ping function to send an ICMP echo request to a target Ethernet device to determine:

- if the target device is present, and if so
- the elapsed time to receive an echo response from the target device

The target device is identified by its IP address setting. Enter only valid IP addresses in the **IP Address** field.

The ping function can be performed in the **Ping** page of the **Online Action** window:

The screenshot shows the 'Online Action' window with the 'Ping' tab selected. The window has three tabs: 'Module Information', 'Port Configuration', and 'Ping'. The 'Ping' tab contains the following elements:

- Address** section: A label 'Address' above a text field 'IP Address' containing the value '192.168.1.6'.
- Ping** section: A 'Ping' button, two checkboxes labeled 'Repeat (100ms)' and 'Stop on Error' (both unchecked), and a 'Clear' button.
- Ping Result** section: A large text area for displaying the results of the ping operation.

Pinging a Network Device

Ping a network device:

Step	Action
1	In the DTM Browser , select the CPU upstream of the remote EtherNet/IP device you want to ping.
2	Right-click and select Device Menu → Online Action . Result: The Online Action window opens.
3	In the Online Action window, select the device you want to ping. Result: The window displays pages containing online information for the selected device. NOTE: The specific collection of displayed pages depends on the type of device selected: <ul style="list-style-type: none"> ● the CPU ● a remote EtherNet/IP device ● a remote Modbus TCP device
4	Select the Ping page. To send... <ul style="list-style-type: none"> ● a single ping: Deselect the Repeat checkbox. ● a series of pings (1 every 100 ms): Select the Repeat checkbox.
5	(Optional) Select Stop on Error to stop pinging an unsuccessful communication.
6	Click Ping once to begin pinging.
7	Click Ping a second time to stop repeated pinging, where no error has been detected.
8	The Ping Result box displays the ping outcome. Click Clear to empty the Ping Result box.

Section 5.6

Diagnostics Available through Modbus/TCP

Modbus Diagnostic Codes

Introduction

CPUs and BMENOC0301/11 communication modules in M580 systems support the diagnostic codes in these tables.

Function Code 3

Some module diagnostics (I/O connection, extended health, redundancy status, FDR server, etc.) are available to Modbus clients that read the local Modbus server area. Use Modbus function code 3 with the unit ID set to 100 for register mapping:

Type	Offset Modbus Address	Size (Words)
Basic Networks Diagnostic Data	0	39
Ethernet Port Diagnostics Data (Internal port)	39	103
Ethernet Port Diagnostics Data (ETH 1)	142	103
Ethernet Port Diagnostics Data (ETH 2)	245	103
Ethernet Port Diagnostics Data (ETH 3)	348	103
Ethernet Port Diagnostics Data (backplane)	451	103
Modbus TCP/Port 502 Diagnostic Data	554	114
Modbus TCP/Port 502 Connection Table Data	668	515
SNTP Diagnostics	1218	57
QoS Diagnostics	1275	11
Identify	2001	24

Function Code 8

Modbus function code 08 provides a variety of diagnostic functions::

Operation Code	Diag. Control	Description
0x01	0x0100	network diagnostic data
	0x0200	Read the Ethernet port diagnostic data from the switch manager.
	0x0300	Read the Modbus TCP/port 502 diagnostic data from the Modbus server.
	0x0400	Read the Modbus TCP/port 502 connection table from the Modbus server.
	0x07F0	Read the data structure offset data from the Modbus server.
0x02	0x0100	Clear the basic network diagnostic data. NOTE: Only specific parameters of basic network diagnostic data are used to clear requests.
	0x0200	Clear the Ethernet port diagnostic data. NOTE: Only specific parameters of basic network diagnostic data are used to clear requests.
	0x0300	Clear the Modbus TCP/port 502 diagnostic data. NOTE: Only specific parameters of Modbus port 502 diagnostic data are used to clear requests.
	0x0400	Clear the Modbus TCP/port 502 connection table. NOTE: Only specific parameters of Modbus port 502 connection data are use to clear requests.
0x03	0	Clear all diagnostic data. NOTE: Only specific parameters of each diagnostic data are used to clear requests.

Read Device Identification

Modbus function code 43, subcode 14: A Modbus request associated with function code 43 (Read Device Identification) asks a Modbus server to return the vendor name, product name, version number, and other optional fields:

Category	Object ID	Object Name	Type	Requirement
Basic	0x00	VendorName (vendor name)	ASCII string	mandatory
	0x01	ProductCode (product code)	ASCII string	mandatory
	0x02	MajorMinorRevision (version number)	ASCII string	mandatory
Regular	0x03	VendorUrl (vendor URL)	ASCII string	optional
	0x04	ProductName (product name)	ASCII string	optional
	0x05	ModelName (model name)	ASCII string	optional
	0x06	UserApplicationName (user application name)	ASCII string	optional
	0x07...0x7F	(reserved)	ASCII string	optional
Extended	0x80...0xFF	device-dependent		optional

This table provides sample responses to the Modbus request (function code 43, subcode 14):

Module	0x00 Vendor ID	0x01 Part Number	0x02 Version
BMEP584020 CPU	Schneider Electric	BMEP584020	v02.10
BMENOC0301 module	Schneider Electric	BMENOC0301	V02.04 build 0009
BMENOC0311 module	Schneider Electric	BMENOC0311	V02.04 build 0009
BMENOC0321 module	Schneider Electric	BMENOC0321	V01.01 build 0004

Section 5.7

Diagnostics Available through EtherNet/IP CIP Objects

Introduction

Modicon M580 applications use CIP within a producer/consumer model to provide communication services in an industrial environment. This section describes the available CIP objects for Modicon M580 CPU modules.

What Is in This Section?

This section contains the following topics:

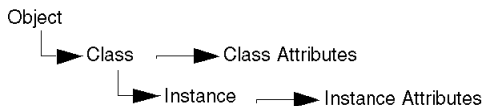
Topic	Page
About CIP Objects	166
Identity Object	167
Assembly Object	169
Connection Manager Object	171
Modbus Object	173
Quality Of Service (QoS) Object	175
TCP/IP Interface Object	177
Ethernet Link Object	179
EtherNet/IP Interface Diagnostics Object	183
EtherNet/IP IO Scanner Diagnostics Object	186
IO Connection Diagnostics Object	188
EtherNet/IP Explicit Connection Diagnostics Object	192
EtherNet/IP Explicit Connection Diagnostics List Object	194
RSTP Diagnostics Object	196
Service Port Control Object	200

About CIP Objects

Overview

The Ethernet communication module can access CIP data and services located in connected devices. The CIP objects and their content depend on the design of each device.

CIP object data and content are exposed—and accessed—hierarchically in the following nested levels:



NOTE:

You can use explicit messaging to access these items:

- Access a collection of instance attributes by including only the class and instance values for the object in the explicit message.
- Access a single attribute by adding a specific attribute value to the explicit message with the class and instance values for the object.

This chapter describes the CIP objects that the Ethernet communication module exposes to remote devices.

Identity Object

Overview

The Identity object presents the instances, attributes and services described below.

Class ID

01

Instance IDs

The Identity object presents two instances:

- 0: class
- 1: instance

Attributes

Identity object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID		Description	Type	GET	SET
hex	dec				
01	01	Vendor ID	UINT	X	—
02	02	Device Type	UINT	X	—
03	03	Product Code	UINT	X	—
04	04	Revision	STRUCT	X	—
		Major	USINT		
		Minor	USINT		
05	05	Status bit 2: 0x01=the module is configured bits 4-7: 0x03=no I/O connections established 0x06=at least 1 I/O connection in run mode 0x07=at least 1 I/O connection established, all in IDLE mode	Word	X	—
06	06	Serial Number	UDINT	X	—
07	07	Product Name	STRING	X	—
18	24	Modbus Identity	STRUCT	X	—
X = supported — = not supported					

Services

The Identity object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns: <ul style="list-style-type: none"> all class attributes (instance = 0) instance attributes 1 to 7 (instance = 1)
0E	14	Get_Attribute_Single	X	X	Returns the value of the specified attribute.
X = supported — = not supported					

Assembly Object

Overview

The assembly object consists of the attributes and services. Assembly instances exist only when you configure local slaves (*see page 304*) for the M580 CPU modules.

You can send an explicit message to the assembly object only when no other connections have been established that read from or write to this object. For example, you can send an explicit message to the assembly object if a local slave instance is enabled, but no other module is scanning that local slave.

Class ID

04

Instance IDs

The assembly object presents these instance identifiers:

- 0: class
- 101, 102, 111, 112, 121, 122: instance

Attributes

The assembly object consists of these attributes:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
03	Number of Instances	X	—
X = supported — = not supported			

Instance attributes:

Instance ID	Attribute ID	Description	Type	GET	SET
101	03	Local slave 1: T->O (output data)	Array of BYTE	X	—
102		Local slave 1: O>T (input data)	Array of BYTE	X	—
111	03	Local slave 2: T->O (output data)	Array of BYTE	X	—
112		Local slave 2: O>T (input data)	Array of BYTE	X	—
X = supported — = not supported					

Services

The CIP assembly object performs these services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
0E	14	Get_Attribute_Single	X	X	Returns the value of the specified attribute
X = supported — = not supported					
1. When valid, the size of the data written to the assembly object using the Set_Attribute_Single service equals the size of the assembly object as configured in the target module.					

Connection Manager Object

Overview

The Connection Manager object presents the instances, attributes and services described below.

Class ID

06

Instance IDs

The Connection Manager object presents two instance values:

- 0: class
- 1: instance

Attributes

Connection Manager object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID		Description	Type	GET	SET	Value
hex	dec					
01	01	Open Requests	UINT	X	X	Number of Forward Open service requests received
02	02	Open Format Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to bad format
03	03	Open Resource Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to lack of resources
X = supported — = not supported						

Attribute ID		Description	Type	GET	SET	Value
hex	dec					
04	04	Open Other Rejects	UINT	X	X	Number of Forward Open service requests that were rejected for reasons other than bad format or lack of resources
05	05	Close Requests	UINT	X	X	Number of Forward Close service requests received
06	06	Close Format Requests	UINT	X	X	Number of Forward Close service requests that were rejected due to bad format
07	07	Close Other Requests	UINT	X	X	Number of Forward Close service requests that were rejected for reasons other than bad format
08	08	Connection Timeouts	UINT	X	X	Total number of connection timeouts that occurred in connections controlled by this connections manager
09	09	Connection Entry List	STRUCT	X	—	0 (Unsupported optional item)
0B	11	CPU_Utilization	UINT	X	—	0 (Unsupported optional item)
0C	12	MaxBufSize	UDINT	X	—	0 (Unsupported optional item)
0D	13	BufSize Remaining	UDINT	X	—	0 (Unsupported optional item)
X = supported — = not supported						

Services

The Connection Manager object performs the following services on the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
0E	14	Get_Attribute_Single	X	X	Returns the value of the specified attribute.
X = supported — = not supported					

Modbus Object

Overview

The Modbus object converts EtherNet/IP service requests to Modbus functions, and Modbus exception codes to CIP General Status codes. It presents the instances, attributes and services described below.

Class ID

44 (hex), 68 (decimal)

Instance IDs

The Modbus object presents two instance values:

- 0: class
- 1: instance

Attributes

The Modbus object consists of the following attributes:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID	Description	Type	GET	SET
—	No instance attributes are supported	—	—	—

Services

The Modbus object performs the following services upon the listed object types:

Service ID		Description	Class	Instance
hex	dec			
0E	14	Get_Attribute_Single	X	X
4B	75	Read_Discrete_Inputs	—	X
4C	76	Read_Coils	—	X
4D	77	Read_Input_Registers	—	X
4E	78	Read_Holding_Registers	—	X
4F	79	Write_Coils	—	X
50	80	Write_Holding_Registers	—	X
51	81	Modbus_Passthrough	—	X
X = supported — = not supported				

Quality Of Service (QoS) Object

Overview

The QoS object implements Differentiated Services Code Point (DSCP or *DiffServe*) values for the purpose of providing a method of prioritizing Ethernet messages. The QoS object presents the instances, attributes and services described below.

Class ID

48 (hex), 72 (decimal)

Instance IDs

The QoS object presents two instance values:

- 0: class
- 1: instance

Attributes

The QoS object consists of the following attributes:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID	Description	Type	GET	SET	Value
04	DSCP Urgent	USINT	X	X	For CIP transport class 0/1 Urgent priority messages.
05	DSCP Scheduled	USINT	X	X	For CIP transport class 0/1 Urgent priority messages.
06	DSCP High	USINT	X	X	For CIP transport class 0/1 Urgent priority messages.
07	DSCP Low	USINT	X	X	For CIP transport class 0/1 Urgent priority messages.
08	DSCP Explicit	USINT	X	X	For CIP explicit messages (transport class 2/3 and UCMM).
X = supported — = not supported					

NOTE: A change in the instance attribute value takes effect on device re-start, for configurations made from flash memory.

Services

The QoS object performs the following services upon the listed object types:

Service ID		Description	Class	Instance
hex	dec			
0E	14	Get_Attribute_Single	X	X
10	16	Set_Attribute_Single	—	X
X = supported — = not supported				

TCP/IP Interface Object

Overview

The TCP/IP interface object presents the instances (per network), attributes and services described below.

Class ID

F5 (hex), 245 (decimal)

Instance IDs

The TCP/IP interface object presents 2 instance values:

- 0: class
- 1: instance

Attributes

TCP/IP interface object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID	Description	Type	GET	SET	Value
01	Status	DWORD	X	—	0x01
02	Configuration Capability	DWORD	X	—	0x01 = from BootP 0x11 = from flash 0x00 = other
03	Configuration Control	DWORD	X	X	0x01 = out-of-box default
04	Physical Link Object	STRUCT	X	—	
	Path Size	UINT			
	Path	Padded EPATH			
05	Interface Configuration	STRUCT	X	X	0x00 = out-of-box default
	IP Address	UDINT			
	Network Mask	UDINT			
	Gateway Address	UDINT			
	Name Server	UDINT			
	Name Server 2	UDINT			
	Domain Name	STRING			
06	Host Name	STRING	X	—	
X = supported — = not supported					

Services

The TCP/IP interface object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
0E	14	Get_Attribute_Single	X	X	Returns the value of the specified attribute.
10	16	Set_Attribute_Single ¹	—	X	Sets the value of the specified attribute.
X = supported — = not supported					
1. The Set_Attribute_Single service can execute only when these preconditions are satisfied:					
<ul style="list-style-type: none"> ● Configure the Ethernet communication module to obtain its IP address from flash memory. ● Confirm that the PLC is in stop mode. 					

Ethernet Link Object

Overview

The Ethernet Link object consists of the instances, attributes, and services described below.

Class ID

F6 (hex), 246 (decimal)

Instance IDs

The Ethernet Link object presents these instance values:

- 101: backplane slot 1
- 102: backplane slot 2
- 103: backplane slot 3
- ...
- 112: backplane slot 12
- 255: internal port

Attributes

The Ethernet Link object presents the following attributes:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
03	Number of Instances	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID		Description	Type	GET	SET	Value
hex	dec					
01	01	Interface Speed	UDINT	X	—	Valid values: 0, 10, 100.
02	02	Interface Flags	DWORD	X	—	Bit 0: link status 0 = Inactive 1 = Active Bit 1: duplex mode 0 = half duplex 1 = full duplex Bits 2...4: negotiation status 3 = successfully negotiated speed and duplex 4 = forced speed and link Bit 5: manual setting requires reset 0 = automatic 1 = device need reset Bit 6: local hardware detected error 0 = no event 1 = event detected
03	03	Physical Address	ARRAY of 6 USINT	X	—	module MAC address
04	04	Interface Counters	STRUCT	X	—	
		In octets	UDINT			octets received on the interface
		In Ucast Packets	UDINT			unicast packets received on the interface
		In NUcast Packets	UDINT			non-unicast packets received on the interface
		In Discards	UDINT			inbound packets received on the interface, but discarded
		In Errors	UDINT			inbound packets with detected errors (does not include in discards)
		In Unknown Protos	UDINT			inbound packets with unknown protocol
		Out Octets	UDINT			octets sent on the interface
		Out Ucast Packets	UDINT			unicast packets sent on the interface
		Out NUcast Packets	UDINT			non-unicast packets sent on the interface
		Out Discards	UDINT			outbound packets discarded
		Out Errors	UDINT			outbound packets with detected errors
X = supported — = not supported						

Attribute ID		Description	Type	GET	SET	Value
hex	dec					
05	05	Media Counters	STRUCT	X	—	
		Alignment Errors	UDINT			frames that are not an integral number of octets in length
		FCS Errors	UDINT			bad CRC — frames received do not pass the FCS check
		Single Collisions	UDINT			successfully transmitted frames that experienced exactly 1 collision
		Multiple Collisions	UDINT			successfully transmitted frames that experienced more than 1 collision
		SQE Test Errors	UDINT			number of times the detected SQE test error is generated
		Deferred Transmissions	UDINT			frames for which first transmission attempt is delayed because the medium is busy
		Late Collisions	UDINT			number of times a collision is detected later than 512 bit times into the transmission of a packet
		Excessive Collisions	UDINT			frames that do not transmit due to excessive collisions
		MAC Transmit Errors	UDINT			frames that do not transmit due to a detected internal MAC sublayer transmit error
		Carrier Sense Errors	UDINT			times that the carrier sense condition was lost or not asserted when attempting to transmit a frame
		Frame Too Long	UDINT			frames received that exceed the maximum permitted frame size
		MAC Receive Errors	UDINT			frames not received on an interface due to a detected internal MAC sublayer receive error
X = supported — = not supported						

Attribute ID		Description	Type	GET	SET	Value
hex	dec					
06	06	Interface Control	STRUCT	X	X	API of the connection
		Control Bits	WORD			Bit 0: Auto-negotiation disabled (0) or enabled (1). NOTE: When auto-negotiation is enabled, 0x0C (object state conflict) is returned when attempting to set either: <ul style="list-style-type: none"> ● forced interface speed ● forced duplex mode
		Forced Interface Speed	UINT			Bit 1: forced duplex mode (if auto-negotiation bit = 0) 0 = half duplex 1 = full duplex
10	16	Interface Label	SHORT_STRING	X	—	Valid values include 10000000 and 100000000. NOTE: Attempting to set any other value returns the detected error 0x09 (invalid attribute value). A fixed textual string identifying the interface, that should include 'internal' for internal interfaces. Maximum number of characters is 64.
X = supported — = not supported						

Services

The Ethernet Link object performs the following services upon the listed object types:

Service ID		Description	Class	Instance
hex	dec			
01	01	Get_Attributes_All	X	X
10	16	Set_Attribute_Single	—	X
0E	14	Get_Attribute_Single	X	X
4C	76	Get_and_Clear	—	X
X = supported — = not supported				

EtherNet/IP Interface Diagnostics Object

Overview

The EtherNet/IP Interface Diagnostics object presents the instances, attributes and services described below.

Class ID

350 (hex), 848 (decimal)

Instance IDs

The EtherNet/IP Interface object presents two instance values:

- 0: class
- 1: instance

Attributes

EtherNet/IP Interface Diagnostics object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID	Description	Type	GET	SET	Value
01	Protocols Supported	UINT	X	—	
02	Connection Diagnostics	STRUCT	X	—	
	Max CIP IO Connections opened	UINT			Number of Class 1 connections opened since the last reset
	Current CIP IO Connections	UINT			Number of Class 1 connections currently opened
	Max CIP Explicit Connections opened	UINT			Number of Class 3 connections opened since the last reset
	Current CIP Explicit Connections	UINT			Number of Class 3 connections currently opened
	CIP Connections Opening Errors	UINT			Increments each time a Forward Open is not successful (Originator and Target)
	CIP Connections Timeout Errors	UINT			Increments when a connection times out (Originator and Target)
	Max EIP TCP Connections opened	UINT			Number of TCP connections (used for EIP, as client or server) opened since the last reset
	Current EIP TCP Connections	UINT			Number of TCP connections (used for EIP, as client or server) currently open
03	IO Messaging Diagnostics	STRUCT	X	X	
	IO Production Counter	UDINT			Increments each time a Class 0/1 message is sent
	IO Consumption Counter	UDINT			Increments each time a Class 0/1 message is received
	IO Production Send Errors Counter	UINT			Increments each time a Class 0/1 message is not sent
	IO Consumption Receive Errors Counter	UINT			Increments each time a consumption is received with a detected error
X = supported — = not supported					

Attribute ID	Description	Type	GET	SET	Value
04	Explicit Messaging Diagnostics	STRUCT	X	X	
	Class 3 Msg Send Counter	UDINT			Increments each time a Class 3 message is sent (client and server)
	Class 3 Msg Receive Counter	UDINT			Increments each time a Class 3 message is received (client and server)
	UCMM Msg Receive Counter	UDINT			Increments each time a UCMM message is sent (client and server)
	UCMM Msg Receive Counter	UDINT			Increments each time a UCMM message is received (client and server)
X = supported — = not supported					

Services

The EtherNet/IP Interface Diagnostics object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
0E	14	Get_Attribute_Single	—	X	Returns the value of the specified attribute.
4C	76	Get_and_Clear	—	X	Returns and clears the values of all instance attributes.
X = supported — = not supported					

EtherNet/IP IO Scanner Diagnostics Object

Overview

The EtherNet/IP IO Scanner Diagnostics object presents the instances, attributes and services described below.

Class ID

351 (hex), 849 (decimal)

Instance IDs

The EtherNet/IP IO Scanner Diagnostics object presents two instances:

- 0: class
- 1: instance

Attributes

EtherNet/IP IO Scanner Diagnostics object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 (instance attributes):

Attribute ID	Description	Type	GET	SET
01	IO Status Table	STRUCT	X	—
	Size	UINT		
	Status	ARRAY of UNINT		
X = supported — = not supported				

Services

The EtherNet/IP IO Scanner Diagnostics object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
0E	14	Get_Attribute_Single	X	X	Returns the value of the specified attribute.
X = supported — = not supported					

IO Connection Diagnostics Object

Overview

The IO Connection Diagnostics object presents the instances, attributes and services described below.

Class ID

352 (hex), 850 (decimal)

Instance IDs

The IO Connection Diagnostics object presents two instance values:

- 0 (class)
- 257 ... 643 (instance): The instance number matches the connection number in the **Connection Settings** configuration (*see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*).

NOTE: The Instance ID number = the Connection ID. For *M580* specifically, you can look up the Connection ID on the DTM Device List screen.

Attributes

IO Connection Diagnostics object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 to 256 (instance attributes):

Attribute ID	Description	Type	GET	SET	Value
01	IO Communication Diagnostics	STRUCT	X	X	
	IO Production Counter	UDINT			Increments at each production
	IO Consumption Counter	UDINT			Increments at each consumption
	IO Production Send Errors Counter	UINT			Increments each time a production is not sent
	IO Consumption Receive Errors Counter	UINT			Increments each time a consumption is received with a detected error
	CIP Connection Timeout Errors	UINT			Increments when a connection times out
	CIP Connection Opening Errors	UINT			Increments each time a connection is unable to open
	CIP Connection State	UINT			State of the Connection Bit
	CIP Last Error General Status	UINT			General status of the last error detected on the connection
	CIP Last Error Extended Status	UINT			Extended status of the last error detected on the connection
	Input Communication Status	UINT			Communication status of the inputs (see table, below)
	Output Communication Status	UINT			Communication status of the outputs (see table, below)

X = supported

— = not supported

Attribute ID	Description	Type	GET	SET	Value
02	Connection Diagnostics	STRUCT	X	X	
	Production Connection ID	UDINT			Connection ID for production
	Consumption Connection ID	UDINT			Connection ID for consumption
	Production RPI	UDINT			RPI for production
	Production API	UDINT			API for production
	Consumption RPI	UDINT			RPI for consumption
	Consumption API	UDINT			API for consumption
	Production Connection Parameters	UDINT			Connection parameters for production
	Consumption Connection Parameters	UDINT			Connection parameters for consumption
	Local IP	UDINT			—
	Local UDP Port	UINT			—
	Remote IP	UDINT			—
	Remote UDP Port	UINT			—
	Production Multicast IP	UDINT			Multicast IP used for production (or 0)
	Consumption Multicast IP	UDINT			Multicast IP used for consumption (or 0)
	Protocols Supported	UDINT			Protocol supported on the connection: 1 = EtherNet/IP
X = supported — = not supported					

The following values describe the structure of the instance attributes: *CIP Connection State*, *Input Communication Status*, and *Output Communication Status*:

Bit Number	Description	Values
15...3	<i>Reserved</i>	0
2	Idle	0 = no idle notification 1 = idle notification
1	Consumption inhibited	0 = consumption started 1 = no consumption
0	Production inhibited	0 = production started 1 = no production

Services

The EtherNet/IP Interface Diagnostics object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
0E	14	Get_Attribute_Single	—	X	Returns the value of the specified attribute.
4C	76	Get_and_Clear	—	X	Returns and clears the values of all instance attributes.
X = supported — = not supported					

EtherNet/IP Explicit Connection Diagnostics Object

Overview

The EtherNet/IP Explicit Connection Diagnostics object presents the instances, attributes and services described below.

Class ID

353 (hex), 851 (decimal)

Instance IDs

The EtherNet/IP Explicit Connection Diagnostics object presents two instance values:

- 0: class
- 1...N: instance (N= maximum concurrent number of explicit connections)

Attributes

EtherNet/IP Explicit Connection Diagnostics object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID hex	Description	Value	GET	SET
01	Revision	1	X	—
02	Max Instance	0...N	X	—
X = supported — = not supported				

Instance ID = 1 to N (instance attributes):

Attribute ID hex	Description	Type	GET	SET	Value
01	Originator connection ID	UDINT	X	—	Originator to target connection ID
02	Originator IP	UINT	X	—	
03	Originator TCP Port	UDINT	X	—	
04	Target connection ID	UDINT	X	—	Target to originator connection ID
05	Target IP	UDINT	X	—	
06	Target TCP Port	UDINT	X	—	
07	Msg Send Counter	UDINT	X	—	Incremented each time a Class 3 CIP message is sent on the connection
08	Msg Receive counter	UDINT	X	—	Increments each time a Class 3 CIP message is received on the connection
X = supported — = not supported					

Services

The EtherNet/IP Explicit Connection Diagnostics object performs the following services upon the listed object type:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	Returns the value of all attributes.
X = supported — = not supported					

EtherNet/IP Explicit Connection Diagnostics List Object

Overview

The EtherNet/IP Explicit Connection Diagnostics List object presents the instances, attributes and services described below.

Class ID

354 (hex), 852 (decimal)

Instance IDs

The EtherNet/IP Explicit Connection Diagnostics List object presents two instance values:

- 0: class
- 1: instance

Attributes

EtherNet/IP Explicit Connection Diagnostics List object attributes are associated with each instance, as follows:

Instance ID = 0 (class attributes):

Attribute ID	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

Instance ID = 1 to 2 (instance attributes):

Attribute ID	Description	Type	GET	SET	Value
01	Number of connections	UINT	X	—	Total number of opened explicit connections
02	Explicit Messaging Connections Diagnostic List	ARRAY of STRUCT	X	—	
	Originator connection ID	UDINT			O->T connection ID
	Originator IP	UINT			—
	Originator TCP port	UDINT			—
	Target connection ID	UDINT			T->O connection ID
	Target IP	UDINT			—
	Target TCP port	UDINT			—
	Msg Send counter	UDINT			Increments each time a Class 3 CIP message is sent on the connection
	Msg Receive counter	UDINT			Increments each time a Class 3 CIP message is received on the connection
X = supported — = not supported					

Services

The EtherNet/IP Explicit Connection Diagnostics object performs the following services upon the listed object types:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	—	Returns the value of all attributes.
08	08	Create	X	—	—
09	09	Delete	—	X	—
4B	75	Explicit_Connections_Diagnostic_Read	—	X	—
X = supported — = not supported					

RSTP Diagnostics Object

Overview

The RSTP Diagnostics object presents the instances, attributes and services described below.

Class ID

355 (hex), 853 (decimal)

Instance IDs

The RSTP Diagnostics object presents these instance values:

- 0: class
- 1: instance

Attributes

RSTP Diagnostics object attributes are associated with each instance.

Instance ID = 0 (class attributes):

Attribute ID	Description	Type	GET	SET
01	Revision: This attribute specifies the current revision of the RSTP Diagnostic Object. The revision is increased by 1 at each new update of the object.	UINT	X	—
02	Max Instance: This attribute specifies the maximum number of instances that may be created for this object on a per device basis (for example, an RSTP Bridge). There is 1 instance for each RSTP port on a device.	UINT	X	—
X = supported — = not supported				

Instance ID = 1 to N (instance attributes):

Attribute ID	Description	Type	GET	CLEAR	Value
01	Switch Status	STRUCT	X	—	—
	Protocol Specification	UINT	X	—	Refer to RFC-4188 for attribute definitions and value range. In addition, the following value is defined: [4]: the protocol is IEEE 802.1D-2004 and IEEE 802.1W
	Bridge Priority	UDINT	X	—	Refer to RFC-4188 for attribute definitions and value range.
	Time Since Topology Change	UDINT	X	—	
	Topology Change Count	UDINT	X	—	Refer to RFC-4188 for attribute definitions and value range.
	Designated Root	String	X	—	Refer to RFC-4188 for attribute definitions and value range.
	Root Cost	UDINT	X	—	
	Root Port	UDINT	X	—	
	Max Age	UINT	X	—	
	Hello Time	UINT	X	—	
	Hold Time	UDINT	X	—	
	Forward Delay	UINT	X	—	
	Bridge Max Age	UINT	X	—	
	Bridge Hello Time	UINT	X	—	
	Bridge Forward Delay	UINT	X	—	
X = supported — = not supported					

Attribute ID	Description	Type	GET	CLEAR	Value
02	Port Status	STRUCT	X	X	—
	Port	UDINT	X	X	Refer to RFC-4188 for attribute definitions and value range.
	Priority	UDINT	X	X	
	State	UINT	X	X	
	Enable	UINT	X	X	
	Path Cost	UDINT	X	X	
	Designated Root	String	X	X	
	Designated Cost	UDINT	X	X	
	Designated Bridge	String	X	X	
	Designated Port	String	X	X	
	Forward Transitions Count	UDINT	X	X	Refer to RFC-4188 for attribute definitions and value range. Services: <ul style="list-style-type: none"> ● Get_and_Clear: The current value of this parameter is returned with the response message. ● other services: The current value of this parameter is returned without being cleared.
03	Port Mode	STRUCT	X	—	—
	Port Number	UINT	X	—	This attribute indicates the port number for a data query. The value range is configuration dependent. For a 4-port Ethernet device, as an instance, the valid range is 1...4.
	Admin Edge Port	UINT	X	—	This attribute indicates if this is a user-configured edge port: <ul style="list-style-type: none"> ● 1: true ● 2: false Other values are not valid.
	Oper Edge Port	UINT	X	—	This attribute indicates if this port is currently an edge port: <ul style="list-style-type: none"> ● 1: true ● 2: false Other values are not valid.
	Auto Edge Port	UINT	X	—	This attribute indicates if this port is a dynamically determined edge port: <ul style="list-style-type: none"> ● 1: true ● 2: false Other values are not valid.
X = supported — = not supported					

Services

The RSTP Diagnostics object performs these services:

Service ID		Description	Class	Instance	Notes
hex	dec				
01	01	Get_Attributes_All	X	X	This service returns: <ul style="list-style-type: none"> all attributes of the class all attributes of the instance of the object
02	02	Get_Attribute_Single	X	X	This service returns: <ul style="list-style-type: none"> the contents of a single attribute of the class the contents of the instance of the object as specified Specify the attribute ID in the request for this service.
32	50	Get_and_Clear	—	X	This service returns the contents of a single attribute of the instance of the object as specified. Then the relevant counter-like parameter(s) within the specified attribute are cleared. (Specify the attribute ID in the request for this service.)
X = supported — = not supported					

Service Port Control Object

Overview

The Service Port Control object is defined for port control purposes.

Class ID

400 (hex), 1024 (decimal)

Instance IDs

The Service Port Control object presents these instance Values:

- 0: class
- 1: instance

Attributes

Service Port Control object attributes are associated with each instance.

Required class attributes (instance 0):

Attribute ID	Description	Type	Get	Set
01	Revision	UINT	X	—
02	Max Instance	UINT	X	—
X = supported — = not supported				

Required instance attributes (instance 1):

Attribute ID		Description	Type	Get	Set	Value
hex	dec					
01	01	Port Control	UINT	X	X	0 (default): disabled 1: access port 2: port mirroring
02	02	Mirror	UINT	X	X	bit 0 (default): ETH 2 port bit 1: ETH 3 port bit 2: backplane port bit 3: internal port
X = supported — = not supported						

NOTE:

- If the SERVICE port is not configured for port mirroring, the mirror attribute is ignored. If the value of a parameter request is outside the valid range, the service request is ignored.
- In port mirroring mode, the SERVICE port acts like a read-only port. That is, you cannot access devices (ping, connection to Unity Pro, etc.) through the SERVICE port.

Services

The Service Port Control object performs these services for these object types:

Service ID		Name	Class	Instance	Description
hex	dec				
01	01	Get_Attributes_All	X	X	Get all attributes in a single message.
02	02	Set_Attributes_All	—	X	Set all attributes in a single message.
0E	14	Get_Attribute_Single	X	X	Get a single specified attribute.
10	16	Set_Attribute_Single	—	X	Set a single specified attribute.
X = supported — = not supported					

Section 5.8

DTM Device Lists

Introduction

This section describes the connection of an M580 CPU to other network nodes through the Unity Pro **DTM Browser**.

What Is in This Section?

This section contains the following topics:

Topic	Page
Device List Configuration and Connection Summary	203
Device List Parameters	206
Standalone DDT Data Structure for M580 CPUs	211
Hot Standby DDT Data Structure	219

Device List Configuration and Connection Summary

Introduction

The Device List contains read-only properties that summarize these items:

- configuration data:
 - input data image
 - output data image
 - maximum and actual numbers of devices, connections, and packets
- Modbus request and EtherNet/IP connection summary

Open the Page

View the read-only properties of the M580 CPU in the Unity Pro **Device List**:

Step	Action
1	Open your Unity Pro project.
2	Open the DTM Browser (Tools → DTM Browser).
3	Double-click the CPU DTM in the DTM Browser to open the configuration window. NOTE: You can also right-click the CPU DTM and select Open .
4	Select Device List in the navigation tree.

Configuration Summary Data

Select **Device List** and view the **Configuration Summary** table on the **Summary** tab to see values for these items:

- **Input**
- **Output**
- **Configuration Size**

Expand (+) the **Input** row to view the **Input Current Size** values:

Description	Source
This value is the sum of Modbus requests and EtherNet/IP connection sizes.	This value is configured in the General page for a selected distributed device and connection.

Expand (+) the **Output** row to view the **Output Current Size** values:

Description	Source
This value is the sum of Modbus requests and EtherNet/IP connection sizes.	This value is configured in the General page for a selected distributed device and connection.

The maximum size of the X Bus input or output memory variable is 4 KB (2048 words). The variable contains a 16-byte descriptor followed by a value that represents the number of input or output data objects. Each data object contains a 3-byte object header followed by the input or output data. The number of data objects and the size of the input or output data depend on the configuration. The maximum overhead in the variable is 403 bytes (16 + 387), where 16 is the number of bytes in the descriptor and 387 is the product of 3 x 129, where 3 is the number of bytes in the header and 129 is the number of input or output objects (128 maximum scanned devices or local slaves that the BMENOC03*1 module supports plus one input or output object for the scanner DDDT). Therefore, at least 3.6 KB of the 4-KB variable is available for the input or output current size.

NOTE: The input current size also includes 28 words of scanner DDT input data. The output current size also includes 24 words of scanner DDT output data.

Expand (+) the **Configuration Size** row in the **Connection Summary** table to view these values:

Name	Description	Source
Maximum Number of DIO Devices	the maximum number of distributed devices that can be added to the configuration	predefined
Current Number of DIO Devices	the number of distributed devices in the current configuration	network design in the Unity Pro device editor
Maximum Number of DIO Connections	the maximum number of connections to distributed devices that can be managed by the CPU	predefined
Current Number of DIO Connections	the number of connections to distributed devices in the current configuration	network design in the Unity Pro device editor
Maximum Number of Packets	the maximum number of packets per second the module is able to manage	predefined
Current Number of Input Packets	total number of input packets (traffic) per second, based on the current number of modules and its configured input data	network design in the Unity Pro device editor
Current Number of Output Packets	total number of output packets (traffic) per second, based on the current number of modules and its configured output data	network design in the Unity Pro device editor
Current Number of Total Packets	total number of packets (traffic in both directions) per second, based on the current number of modules and its configured I/O data	network design in the Unity Pro device editor

Request / Connection Summary Data

Select **Device List** and view the **Request / Connection Summary** table on the **Summary** tab. The Unity Pro DTM uses this information to calculate the total bandwidth that distributed equipment consumes:

Column	Description
Connection Bit	<ul style="list-style-type: none"> Connection health bits display the status of each device with one or more connections. Connection control bits can be toggled on and off using object IDs.
Task	The task that is associated with this connection.
Input Object	The ID of the input object associated with the connection (see the note following the table).
Output Object	The ID of the output object associated with the connection (see the note following the table).
Device	The device Number is used for the health and control bit index.
Device Name	A unique name associated with the device that owns the connection.
Type	The target device type: <ul style="list-style-type: none"> EtherNet/IP Local Slave Modbus TCP
Address	The target device IP address for remote devices (does not apply to local slaves).
Rate (msec)	The RPI (for EtherNet/IP) or the repetitive rate (for Modbus TCP), in ms.
Input Packets per Second	The number of input (T->O) packets per second exchanged over this connection.
Output Packets per Second	The number of output (O->T) packets per second exchanged over this connection.
Packets per Second	The total number of packets per second exchanged over this connection in both Input and output directions.
Bandwidth Usage	The total bandwidth used by this connection (total bytes per second traffic).
Size In	The number of input words configured for this remote device.
Size Out	The number of output words configured for this remote device.

NOTE: The numeric identifiers in the **Input Object** and **Output Object** columns represent the objects associated with a single device connection (scan line). For example, if an EtherNet/IP connection has an input object of 260 and an output object of 261, the corresponding control bits for this connection are in the DIO_CTRL field in the M580 CPU device DDT. Object 260 is the fifth bit and object 261 is the sixth bit in this field. There can be multiple connections for a device. Set the corresponding bits to control the input and output objects for these connections.

Device List Parameters

Introduction

Configure parameters for devices in the **Device List** on these tabs:

- **Properties**
- **Address Setting**
- **Request Setting** (Modbus devices only)

View the Configuration Tabs

Navigate to the **Device List** configuration tabs

Step	Action
1	In the DTM Browser (Tools → DTM Browser), double-click the DTM that corresponds to the CPU.
2	In the navigation pane, expand (+) the Device List (<i>see page 202</i>) to see the associated Modbus TCP and EtherNet/IP devices.
3	Select a device from the Device List to view the Properties , Address Setting , and Request Setting tabs tabs. NOTE: These tabs are described in detail below.

Properties Tab

Configure the **Properties** tab to perform these tasks:

- Add the device to the configuration.
- Remove the device from the configuration.
- Edit the base name for variables and data structures used by the device.
- Indicate how input and output items are created and edited.

Configure the **Properties** tab:

Field	Parameter	Description
Properties	Number	The relative position of the device in the list.
	Active Configuration	Enabled: Add this device to the Unity Pro project configuration. Disabled: Remove this device from the Unity Pro project configuration.
IO Structure Name	Structure Name	Unity Pro automatically assigns a structure name based on the variable name.
	Variable Name	Variable Name: An auto-generated variable name is based on the alias name.
	Default Name	Press this button to restore the default variable and structure names.

Field	Parameter	Description
Items Management	Import Mode	Manual: I/O items are manually added in the Device Editor . The I/O items list is not affected by changes to the device DTM. Automatic: I/O items are taken from the device DTM and updated if the items list in the device DTM changes. Items cannot be edited in the Device Editor .
	Reimport Items	Press this button to import the I/O items list from the device DTM, overwriting any manual I/O item edits. Enabled only when Import mode is set to Manual .

Click **Apply** to save your edits and leave the window open for further edits.

Address Setting Tab

Configure the **Address Setting** page to perform these tasks:

- Configure the IP address for a device.
- Enable or disable DHCP client software for a device.

NOTE: When the DHCP client software is enabled in a Modbus device, it obtains its IP address from the DHCP server in the CPU.

In the **Address Setting** page, edit these parameters to conform to your application's design and functionality:

Field	Parameter	Description
Change Address	IP Address	By default: <ul style="list-style-type: none"> The first three octet values equal the first three octet values of the CPU. The fourth octet value equals this device Number setting. In this case, the default value is 004. In our continuing example, type in the address 192.168.1.17 .
Address Server	DHCP for this Device	Enabled: Activate the DHCP client in this device. The device obtains its IP address from the DHCP service provided by the CPU appears on the auto-generated DHCP client list (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
		Disabled (default): Deactivates the DHCP client in this device.
		NOTE: For this example, select Enabled .
	Identified by	If DHCP for this Device is Enabled , it indicates the device identifier type: <ul style="list-style-type: none"> MAC Address Device Name NOTE: For this example, select Device Name .
	Identifier	If DHCP for this Device is Enabled, the specific device MAC Address or Name value. <p>NOTE: For this example, accept the default setting of NIP2212_01 (based on the Alias name).</p>
	Subnet Mask	The device subnet mask. <p>NOTE: For this example, accept the default value (255.255.255.0).</p>
	Gateway	The gateway address used to reach this device. The default of 0.0.0.0 indicates this device is located on the same subnet as the CPU. <p>NOTE: For this example, accept the default value.</p>

Click **Apply** to save your edits, and leave the window open for further edits.

Request Setting Tab

Configure the **Request Setting** tab to add, configure, and remove Modbus requests for the Modbus device. Each request represents a separate link between the CPU and the Modbus device.

NOTE: The **Request Setting** tab is available only when a Modbus TCP device is selected in the **Device List**.

Create a request:

Step	Action
1	Press the Add Request button to see a new request in the table. Press the Add Request button: <ul style="list-style-type: none"> • The new request appears in the table. • The corresponding request items appear in the Device List. NOTE: The Add Request function is enabled only when Import Mode on the Properties tab is set to Manual .
2	Configure the request settings according to the table below.
3	Repeat these steps to create additional requests.
4	Press the Apply to save the request.

This table describes the **Request Settings** parameters for Modbus devices:

Setting	Description
Connection Bit	This bit indicates the read-only offset for the health bit for this connection. Offset values (starting at 0) are auto-generated by the Unity Pro DTM based on the connection type.
Unit ID	The Unit ID is the number used to identify the target of the connection. NOTE: Consult the manufacturer's user manual for the specific target device to find its Unit ID.
Health Time Out (ms)	This value represents the maximum allowed interval between device responses before a time out is detected: <ul style="list-style-type: none"> • valid range: 5 ... 65535 ms • interval: 5 ms • default: 1500 ms
Repetitive Rate (ms)	This value represents the data scan rate in intervals of 5 ms. (The valid range is 0...60000 ms. The default is 60 ms.)
RD Address	This is the address of the input data image in the Modbus device.
RD Length	This value represents the number of words (0...125) in the Modbus device that the CPU reads.
Last Value	This value represents the behavior of input data in the application if communications are lost: <ul style="list-style-type: none"> • Hold Value (default) • Set To Zero

Setting	Description
WR Address	This is the address of the output data image in the Modbus device.
WR Length	This value represents the number of words (0...120) in the Modbus device to which the CPU writes.

Remove a request:

Step	Action
1	Click a row in the table.
2	Press the Remove button to remove the request. NOTE: The corresponding request items disappear from the Device List .
3	Press the Apply to save the configuration.

The next step is to connect the Unity Pro project to the Modbus device.

Standalone DDT Data Structure for M580 CPUs

Introduction

This topic describes the Unity Pro **Device DDT** tab for an M580 CPU in a local rack. A derived data type (DDT) is a set of elements with the same type (ARRAY) or with different types (structure).

NOTE: The device DDT type supported by a standalone M580 CPU depends on its firmware version, and can be either T_BMEP58_ECPU or T_BMEP58_ECPU_EXT.

Access the Device DDT Tab

Access the device DDT for the CPU in Unity Pro:

Step	Action
1	Open a Unity Pro project that includes an M580 CPU in the configuration.
2	Rebuild the project (Build → Rebuild All Project .)
3	Open the Data Editor in the Unity Pro Project Browser (Tools → Data Editor).
4	Select the Device DDT checkbox.
5	Expand (+) the Device DDT in the Name column.

You can add this variable to an Animation Table (*see page 237*) to read the status and set the object control bit.

NOTE: The red arrow and lock icons in the **Device DDT** table indicate that the variable name was auto-generated by Unity Pro based on the configuration of the communication module, local slave, or distributed device. The variable name cannot be edited.

Input and Output Freshness

This table describes the inputs and outputs that are associated with EtherNet/IP or Modbus devices:

Name	Description
Freshness	<p>This is a global bit:</p> <ul style="list-style-type: none"> ● 1: All input objects below (Freshness_1, Freshness_2, etc.) for the associated device are true (1) and provide up-to-date data. ● 0: One or more inputs (below) is not connected and does not provide up-to-date data.
Freshness_1	<p>This bit represents individual input objects for the connection:</p> <ul style="list-style-type: none"> ● 1: The input object is connected and provides up-to-date data. ● 0: The input object is not connected and does not provide up-to-date data.
Freshness_2	<p>This bit represents an individual input object for the device:</p> <ul style="list-style-type: none"> ● 1: The input object is true (1) and provides up-to-date data. ● 0: The input object is not connected (0) and does not provide up-to-date data.
Freshness_3	
...	

Name	Description
(available)	The rows after the Freshness data are organized in groups of Inputs and Outputs that have user-defined names. The number of input and output rows depends on the number of input and output requests configured for a particular device.

Parameters

Use the Unity Pro **Device DDT** tab to configure parameters for the CPU RIO head on the local rack:

Parameter	Description				
Implicit device DDT	<table> <tr> <td>Name</td><td>the default name of the device DDT</td></tr> <tr> <td>Type</td><td>module type (uneditable)</td></tr> </table>	Name	the default name of the device DDT	Type	module type (uneditable)
Name	the default name of the device DDT				
Type	module type (uneditable)				
Goto details	link to the DDT data editor screen				

Standalone Configuration

These tables describe the fields in the `BMEP58_ECPU_EXT` implicit device DDT type that is used with the CPU RIO communication server in standalone configurations using Unity Pro 10.0 or later and M580 CPU version 2.01 or later.

Input Parameters

The following tables describe the input parameters in the device DDT for the CPU.

`ETH_STATUS` (WORD):

Name	Type	Bit	Description
PORT1_LINK	BOOL	0	0 = ETH 1 link is down. 1 = ETH 1 link is up.
PORT2_LINK	BOOL	1	0 = ETH 2 link is down. 1 = ETH 2 link is up.
PORT3_LINK	BOOL	2	0 = ETH 3 link is down. 1 = ETH 3 link is up.
ETH_BKP_PORT_LINK	BOOL	3	0 = Ethernet backplane link is down. 1 = Ethernet backplane link is up.
REDUNDANCY_STATUS (see the note below.)	BOOL	5	0 = Redundant path is not available. 1 = Redundant path is available.
SCANNER_OK	BOOL	6	0 = Scanner is not present. 1 = Scanner is present.

Name	Type	Bit	Description
GLOBAL_STATUS	BOOL	7	<p>0 = At least one service is not operating normally.</p> <p>NOTE: Refer to the footnotes for SERVICE_STATUS and SERVICE_STATUS2, below, to identify the services that set GLOBAL_STATUS to 0.</p> <p>1 = All services are operating normally.</p>
NETWORK_HEALTH	BOOL	8	<p>0 = A potential network broadcast storm is detected.</p> <p>NOTE: Check your wiring and your CPU and BMENOC0301/11 configurations.</p> <p>1 = A network broadcast storm is not detected.</p>
<p>NOTE: You can monitor breaks in the RIO main ring by diagnosing the REDUNDANCY_STATUS bits in the CPU module device DDT. The system detects and reports in this bit a main ring cable break that persists for at least 5 seconds.</p> <p>REDUNDANCY_STATUS bit value:</p> <p>0: The cable is broken or the device is stopped.</p> <p>1: The loop is present and healthy.</p>			

NOTICE

UNEXPECTED EQUIPMENT BEHAVIOR

Confirm that each module has a unique IP address. Duplicate IP addresses can cause unpredictable module/network behavior.

Failure to follow these instructions can result in equipment damage.

SERVICE_STATUS (WORD):

Name	Type	Bit	Description
RSTP_SERVICE ¹	BOOL	0	0 = RSTP service is not operating normally.
			1 = RSTP service is operating normally or disabled.
PORT502_SERVICE ¹	BOOL	2	0 = Port 502 service is not operating normally.
			1 = Port 502 service is operating normally or disabled.
SNMP_SERVICE ¹	BOOL	3	0 = SNMP service is not operating normally.
			1 = SNMP service is operating normally or disabled.
MAIN_IP_ADDRESS_STATUS	BOOL	4	0 = The main IP address is a duplicate or unassigned.
			1 = The main IP address is unique and valid.
ETH_BKP_FAILURE	BOOL	5	0 = Ethernet backplane hardware is not operating properly.
			1 = Ethernet backplane hardware is operating properly.
ETH_BKP_ERROR	BOOL	6	0 = Ethernet backplane error detected.
			1 = Ethernet backplane is operating properly.
EIP_SCANNER ¹	BOOL	7	0 = Service not operating normally.
			1 = Service operating normally.
MODBUS_SCANNER ¹	BOOL	8	0 = Service not operating normally.
			1 = Service operating normally.
NTP_SERVER ¹	BOOL	9	0 = SNTP server not operating normally.
			1 = SNTP server operating normally.
SNTP_CLIENT ¹	BOOL	10	0 = Service not operating normally.
			1 = Service operating normally.
WEB_SERVER ¹	BOOL	11	0 = Service not operating normally.
			1 = Service operating normally.
FIRMWARE_UPGRADE	BOOL	12	0 = Service not operating normally.
			1 = Service operating normally.
FTP	BOOL	13	0 = Service not operating normally.
			1 = Service operating normally.
FDR_SERVER ¹	BOOL	14	0 = Service not operating normally.
			1 = Service operating normally.
EIP_ADAPTER ¹	BOOL	15	0 = EIP adapter (server) service not operating normally.
			1 = EIP adapter (server) service operating normally.
1. When this service is set to 0, GLOBAL_STATUS is also set to 0.			

SERVICE_STATUS2 (WORD):

Name	Type	Bit	Description
A_B_IP_ADDRESS_STATUS	BOOL	0	0 = Duplicate IP or no IP address assigned. 1 = IP addresses (A/B status) correctly assigned.
LLDP_SERVICE ¹	BOOL	1	0 = LLDP service is not operating normally. 1 = LLDP service is operating normally or disabled.
EVENT_LOG_STATUS	BOOL	2	0 = Event log service is not operating normally. 1 = Event log service is operating normally or is disabled.
LOG_SERVER_NOT_REACHABLE	BOOL	3	1 = No acknowledgment received from the syslog server. 0 = Acknowledgment received from the syslog server
(reserved)	–	2–15	(reserved)
1. When this service is set to 0, GLOBAL_STATUS is also set to 0.			

ETH_PORT_1_2_STATUS (BYTE):

Name	Type	Description
Ethernet ports function and RSTP role coded on 2 bits	Bits 1...0	0: ETH 1 disabled
		1: ETH 1 access port
		2: ETH 1 port mirroring
		3: ETH 1 device network port
	Bits 3...2	reserved (0)
	Bits 5...4	0: ETH 2 disabled
		1: ETH 2 access port
		2: ETH 2 port mirroring
		3: ETH 2 device network port
	Bits 7...6	0: ETH 2 alternate RSTP port
		1: ETH 2 backup RSTP port
		2: ETH 2 designated RSTP port
		3: ETH 2 root RSTP port

ETH_PORT_3_BKP_STATUS (BYTE):

Name	Bit	Description
Ethernet ports function and RSTP role coded on 2 bits	Bits 1...0	0: ETH 3 disabled
		1: ETH 3 access port
		2: ETH 3 port mirroring
		3: ETH 3 device network port
	Bits 3...2	0: ETH 3 alternate RSTP port
		1: ETH 3 backup RSTP port
		2: ETH 3 designated RSTP port
		3: ETH 3 root RSTP port
	Bits 5...4	0: The Ethernet backplane port is disabled.
		1: The Ethernet backplane port is enabled to support Ethernet communications.
	Bits 7...6	reserved (0)

FDR_USAGE:

Type	Type	Description
FDR_USAGE	BYTE	% of FDR server usage

IN_PACKETS (UINT):

Type	Bit	Description
UINT	0-7	number of packets received on the interface (internal ports)

IN_ERRORS (UINT):

Type	Bit	Description
UINT	0-7	number of inbound packets that contain detected errors

OUT_PACKETS (UINT):

Type	Bit	Description
UINT	0-7	number of packets sent on the interface (internal ports)

OUT_ERRORS (UINT):

Type	Bit	Description
UINT	0-7	number of outbound packets that contain detected errors

CONF_SIG (UDINT):

Type	Bit	Description
UDINT	0-15	Signatures of all files on local module FDR server

Output Parameters

Although the complete Hot Standby Device DDT is not exchanged from the primary CPU to the standby CPU, these fields are transferred: DROP_CTRL; RIO_CTRL; DIO_CTRL

These tables describe those output parameters:

DROP_CTRL:

Name	Type	Rank	Description
DROP_CTRL	BOOL	1...32	1 bit per RIO drop (up to 32)

RIO_CTRL:

Name	Type	Rank	Description
RIO_CTRL	BOOL	257...384	1 bit per RIO (up to 128)

DIO_CTRL:

Name	Type	Rank	Description
DIO_CTRL	BOOL	513...640	1 bit per DIO (up to 128)

Device Health Status

Although the complete Hot Standby Device DDT is not exchanged from the primary CPU to the standby CPU, these fields are transferred: DROP_HEALTH; RIO_HEALTH; LS_HEALTH; DIO_HEALTH

This table describes the health of the devices that are scanned by the module. The data is presented as an array of boolean:

Parameter	Type	Health status of ...
DROP_HEALTH	ARRAY [1...32] of BOOL	BM•CRA312•0: One array element corresponds to one BM•CRA312•0 module (up to a maximum of 32 BM•CRA312•0 modules).
RIO_HEALTH	ARRAY [257...384] of BOOL	RIO devices: One array element corresponds to one RIO device (up to a maximum of 128 RIO devices).
LS_HEALTH	ARRAY [1...3] of BOOL	local slaves: One array element corresponds to one local slave (up to a maximum of three local slaves).
DIO_HEALTH	ARRAY [513...640] of BOOL	DIO devices: One array element corresponds to one DIO device (up to a maximum of 128 DIO devices).

Values:

- 1 (true): A device is healthy. The input data from the device is received within the pre-configured health timeout.
- 0 (false): A device is not healthy. The input data from the device is not received within the pre-configured health timeout.

Hot Standby DDT Data Structure

Introduction

The `T_M_ECPU_HSBY` DDT is the exclusive interface between the M580 Hot Standby system and the application running in a BMEH58•040 CPU. The DDT instance should appear as:

`ECPU_HSBY_1.`

NOTICE

RISK OF UNINTENDED OPERATION

Review and manage the `T_M_ECPU_HSBY` DDT for proper operation of the redundant system.

Failure to follow these instructions can result in equipment damage.

The `T_M_ECPU_HSBY` DDT presents three distinct sections:

- `LOCAL_HSBY_STS`: Provides information about the local PAC. Data is both auto-generated by the Hot Standby system, and provided by the application. This data is exchanged with the remote PAC.
- `REMOTE_HSBY_STS`: Provides information about the remote PAC, and contains the image of the last received exchange from the counterpart PAC. The validity of this information is represented by the `REMOTE_STS_VALID` flag in the common part of this DDT.

NOTE: The structure of both the `LOCAL_HSBY_STS` and `Remote_HSBY_STS` sections are determined by the `HSBY_STS_T` data type, and are therefore identical. Each is used to describe data relating to one of the two Hot Standby PACs.

- A common part of the DDT: Consists of several objects, including status data, system control objects, and command objects:
 - Status data is provided by the Hot Standby system as a result of diagnostic checking.
 - System control objects enable you to define and control system behavior.
 - Command data objects include executable commands you can use to modify the system state.

Local PAC versus Remote PAC

The `T_M_ECPU_HSBY` DDT employs the terms *local* and *remote*:

- *Local* refers to the Hot Standby PAC to which your PC is connected.
- *Remote* refers to the other Hot Standby PAC.

Data Boundary Alignment

M580 BMEH58•040 CPUs feature a 32-bit data design. For this reason, stored data objects are placed on a four-byte boundary.

T_M_ECPU_HSBY DDT

⚠ CAUTION**RISK OF UNINTENDED OPERATION**

Before you execute a swap command (either by application logic or in the Unity Pro GUI) confirm that the standby PAC is ready to assume the primary role by verifying that the value of its REMOTE_HSBY_STS.EIO_ERROR bit is 0.

Failure to follow these instructions can result in injury or equipment damage.

The T_M_ECPU_HSBY DDT consists of these objects:

Element	Type	Description	Written by
REMOTE_STS_VALID	BOOL	<ul style="list-style-type: none"> True: Both HSBY_LINK_ERROR and HSBY_SUPPLEMENTARY_LINK_ERROR are set to 0. False: Both HSBY_LINK_ERROR and HSBY_SUPPLEMENTARY_LINK_ERROR are set to 1. 	System
APP_MISMATCH	BOOL	The original application in the two PACs is different.	System
LOGIC_MISMATCH_ALLOWED	BOOL	<ul style="list-style-type: none"> True: The standby remains standby in case of logic mismatch. False: The standby goes into wait state in case of logic mismatch. 	Application
LOGIC_MISMATCH	BOOL	Different revisions of the same application exist in the two PACs.	System
SFC_MISMATCH	BOOL	<ul style="list-style-type: none"> True: The applications in the primary PAC and the standby PAC are different in at least one SFC section. In the event of a switchover, the graphs that are different are reset to their initial state. False (default): All SFC sections are identical. 	System
OFFLINE_BUILD_MISMATCH	BOOL	<p>The two PACs are running different revisions of the same application. In this condition:</p> <ul style="list-style-type: none"> A data exchange between the two PACs may not be possible. A swap or switchover may not be bumpless. Neither PAC can be standby 	System
APP_BUILDCHANGE_DIFF	UINT	The number of build change differences between the applications in the primary PAC versus the standby PAC. Evaluated by the primary.	System

Element	Type	Description	Written by
MAX_APP_BUILDCHANGE_DIFF	UINT	Maximum number of build change differences permitted by the Hot Standby system, from 0...50 (default = 20). Set in the Hot Standby tab as Number of modifications .	Application
FW_MISMATCH_ALLOWED	BOOL	Allows mismatched firmware between primary and standby CPUs: <ul style="list-style-type: none"> • True: the standby remains standby in case of FW mismatch. • False (default): the standby goes into wait state in case of FW mismatch. 	Application
FW_MISMATCH	BOOL	The OS are different in the two PACs.	System
DATA_LAYOUT_MISMATCH	BOOL	The Data layout are different on the two PACs. The data transfer is partially performed.	System
DATA_DISCARDED	UINT	Number of KB sent by the primary and discarded by the standby (rounded up to the next KB). Represents data for variables added to primary, but not to standby.	System
DATA_NOT_UPDATED	UINT	Number of KB not updated by the standby (rounded up to the next KB). Represents variables deleted from the primary that remain in the standby.	System
BACKUP_APP_MISMATCH	BOOL	<ul style="list-style-type: none"> • False: The backup application In the 2 Hot Standby PACs are equal. NOTE: The backup application resides in flash memory or on the SD memory card of the PAC. It is created either by the PLC → Project Backup... → Backup Save command, or by setting the %S66 system bit (Application Backup) to 1. • True: All other cases. 	System
PLCA_ONLINE	BOOL	PAC A is configured to enter the primary or standby state. NOTE: Executable only on PAC A.	Configuration
PLCB_ONLINE	BOOL	PAC B is configured to enter the primary or standby state. NOTE: Executable only on PAC B.	Configuration

Element	Type	Description	Written by
CMD_SWAP	BOOL	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to initiate a switchover. The primary goes into wait, then the standby goes primary, finally the wait goes standby. The command is ignored if there is no standby. <p>NOTE: Executable on both primary and standby.</p> <ul style="list-style-type: none"> Reset to 0 by the system on switchover completion or if there is no standby. <p>NOTE:</p> <ul style="list-style-type: none"> This command is designed to be used by the application in response to detected errors. It is not intended to be used for periodic switchovers. If the application has to switchover periodically, the period between switchovers must not be less than 120 seconds. 	Application / System
CMD_APP_TRANSFER	BOOL	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to start an application transfer from the primary to the standby. Executable only on the primary. <p>NOTE: The application transferred is the backup application, stored in flash memory or on the SD card. If the application running does not match the backup application, perform an application backup (PLC → Project Backup... → Backup Save or set the %S66 system bit to 1) before performing the transfer.</p> <ul style="list-style-type: none"> Reset to 0 by the system on transfer completion. 	Application / System
CMD_RUN_AFTER_TRANSFER	BOOL[0...2]	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to automatically start in Run after a transfer. <p>NOTE: Executable only on the primary.</p> <ul style="list-style-type: none"> Reset to 0 by the system after transfer completion and: <ul style="list-style-type: none"> remote PAC is in Run PAC is not primary by animation table or logic command 	Application / System

Element	Type	Description	Written by
CMD_RUN_REMOTE	BOOL	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to run the remote PAC. This command is ignored if the CMD_STOP_REMOTE is true. NOTE: Executable only on the primary. Reset to 0 by the system when the remote PAC enters standby or wait state. 	Application / System
CMD_STOP_REMOTE	BOOL	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to stop the remote PAC. NOTE: Executable on the primary, the standby, or a stopped PAC. Reset to 0 by the application to end the stop command. 	Application
CMD_COMPARE_INITIAL_VALUE	BOOL	<ul style="list-style-type: none"> Set to 1 by program logic or animation table to begin a comparison of the initial values of variables exchanged by the two Hot Standby PACs. NOTE: Executable on both primary and standby only in Run mode. Reset to 0 by the system when the comparison is complete, or if the comparison is not possible. 	Application / System
INITIAL_VALUE_MISMATCH	BOOL	<ul style="list-style-type: none"> True: if the initial values for exchanged variables are different or if the comparison is not possible. False: if the initial values for exchanged variables are identical. 	System
MAST_SYNCHRONIZED ⁽¹⁾	BOOL	<ul style="list-style-type: none"> True: if the exchanged data from the previous MAST cycle was received by the standby. False (default): if the exchanged data from at least the previous MAST cycle was not received by the standby. <p>NOTE: Closely monitor the MAST_SYNCHRONIZED and FAST_SYNCHRONIZED variables related to the MAST and FAST tasks as indicated at the end of this table.</p>	System

Element	Type	Description	Written by
FAST_SYNCHRONIZED ⁽¹⁾	BOOL	<ul style="list-style-type: none"> True: if the exchanged data from the previous FAST cycle was received by the standby. False (default): if the exchanged data from at least the previous FAST cycle was not received by the standby. <p>NOTE: Closely monitor the MAST_SYNCHRONIZED and FAST_SYNCHRONIZED variables related to the MAST and FAST tasks as indicated at the end of this table.</p>	System
LOCAL_HSBY_STS	T_M_ECPU_HSBY_STS	Hot Standby status for the local PAC	(see below)
REMOTE_HSBY_STS	T_M_ECPU_HSBY_STS	Hot Standby status for the remote PAC	(see below)
<p>(1):</p> <ul style="list-style-type: none"> Closely monitor the MAST_SYNCHRONIZED and FAST_SYNCHRONIZED variables related to the MAST and FAST tasks. If its value is zero (False), then the database exchanged between the primary and the standby PACs is not fully transmitted at every cycle of these tasks. In this situation, change the configured period of this task with a higher value than its current execution time (for the MAST task: %SW0 > %SW30, and for the FAST task %SW1 > %SW33. More details on %SW0 + %SW1 and %SW30 + %SW31 in Unity Pro, System Bits and Words, Reference Manual). Example of consequence: upon an Application Program Transfer (APT) command, the primary PAC might not be able to transfer the program to the standby PAC. 			

T_M_ECPU_HSBY_STS Data Type

The T_M_ECPU_HSBY_STS data type presents the following elements:

Element	Type	Description	Written by
HSBY_LINK_ERROR	BOOL	<ul style="list-style-type: none"> True: No connection on the Hot Standby link. False: The Hot Standby link is operational. 	System
HSBY_SUPPLEMENTARY_LINK_ERROR	BOOL	<ul style="list-style-type: none"> True: No connection on the Ethernet RIO link. False: The Ethernet RIO link is operational. 	System
WAIT	BOOL	<ul style="list-style-type: none"> True: The PAC is in Run state but waiting to go primary or standby. False: The PAC is in standby, primary or stop state. 	System
RUN_PRIMARY	BOOL	<ul style="list-style-type: none"> True: The PAC is in primary state. False: The PAC is in standby, wait or stop state. 	System
RUN_STANDBY	BOOL	<ul style="list-style-type: none"> True: The PAC is in standby state. False: The PAC is in primary, wait or stop state. 	System

Element	Type	Description	Written by
STOP	BOOL	<ul style="list-style-type: none"> True: The PAC is in stop state. False: The PAC is in primary, standby or wait state. 	System
PLC_A	BOOL	<ul style="list-style-type: none"> True: the PAC A/B/Clear switch (<i>see page 44</i>) is in "A" position. False: the PAC switch is not in "A" position. 	System
PLC_B	BOOL	<ul style="list-style-type: none"> True: the PAC A/B/Clear switch (<i>see page 44</i>) is in "B" position. False: the PAC switch is not in "B" position. 	System
EIO_ERROR	BOOL	<ul style="list-style-type: none"> True: The PAC does not detect any of the configured Ethernet RIO drops. False: The PAC detects at least one configured Ethernet RIO drop. <p>NOTE: This bit is always false when no drop is configured.</p>	System
SD_CARD_PRESENT	BOOL	<ul style="list-style-type: none"> True: A valid SD card is inserted. False: No SD card, or an invalid SD card is inserted. 	System
LOCAL_RACK_STS	BOOL]	<ul style="list-style-type: none"> True: The local rack configuration is OK. False: The local rack configuration is not OK (for example, modules missing or in incorrect slots, etc.) 	Application
REGISTER	WORD[0...63]	Unmanaged data added to the application via the Exchange on STBY attribute.	Application

Section 5.9

Explicit Messaging

Introduction

You can configure EtherNet/IP and Modbus TCP explicit messages for the M580 CPU in the following ways:

- Connect the CPU to a Unity Pro project (see *Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).
- Use the DATA_EXCH function block in application logic to transmit EtherNet/IP or Modbus TCP explicit messages.
- Use a WRITE_VAR or a READ_VAR function block to exchange Modbus TCP explicit messages, for example, service data objects (SDOs).

NOTE: A single Unity Pro application can contain more than 16 explicit messaging blocks, but only 16 explicit messaging blocks can be active at the same time.

What Is in This Section?

This section contains the following topics:

Topic	Page
Configuring Explicit Messaging Using DATA_EXCH	227
Configuring the DATA_EXCH Management Parameter	229
Explicit Messaging Services	231
Configuring EtherNet/IP Explicit Messaging Using DATA_EXCH	233
EtherNet/IP Explicit Message Example: Get_Attribute_Single	235
EtherNet/IP Explicit Message Example: Read Modbus Object	238
EtherNet/IP Explicit Message Example: Write Modbus Object	242
Modbus TCP Explicit Messaging Function Codes	246
Configuring Modbus TCP Explicit Messaging Using DATA_EXCH	247
Modbus TCP Explicit Message Example: Read Register Request	249
Sending Explicit Messages to EtherNet/IP Devices	252
Sending Explicit Messages to Modbus Devices	254

Configuring Explicit Messaging Using DATA_EXCH

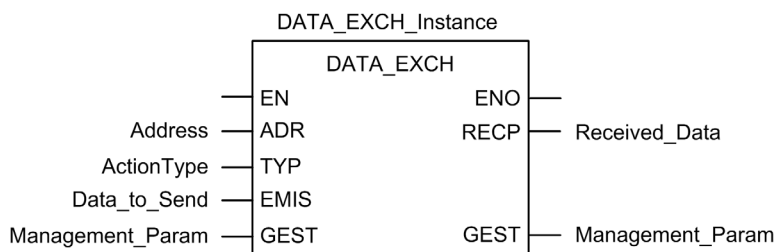
Overview

Use the `DATA_EXCH` function block to configure both Modbus TCP explicit messages and connected and unconnected EtherNet/IP explicit messages.

The `Management_Param`, the `Data_to_Send`, and the `Received_Data` parameters define the operation.

`EN` and `ENO` can be configured as additional parameters.

FBD Representation



Input Parameters

Parameter	Data type	Description
EN	BOOL	This parameter is optional. When this input is set to one, the block is activated and can solve the function blocks algorithm. When this input is set to zero, the block is deactivated and won't solve the function block algorithm.
Address	Array [0...7] of INT	The path to the destination device, the content of which can vary depending on the message protocol. Use the <code>Address</code> function as an input to the block parameter <code>ADR</code> .. Refer to a description of the <code>Address</code> parameter for: <ul style="list-style-type: none"> • EtherNet/IP messages (see page 233) • Modbus TCP messages (<i>see Modicon M340, BMX NOC 0401 Ethernet Communication Module, User Manual</i>)
ActionType	INT	The type of action to perform. For both the EtherNet/IP and Modbus TCP protocols, this setting = 1 (transmission followed by await reception).
Data_to_Send	Array [n...m] of INT	The content of this parameter is specific to the protocol, either EtherNet/IP or Modbus TCP. For EtherNet/IP explicit messaging, refer to the topic Configuring the Data_To_Send Parameter (see page 233). For Modbus TCP explicit messaging, refer to Unity Pro online help.

Input/Output Parameters

The Management_Param array is local:

Parameter	Data type	Description
Management_Param	Array [0...3] of INT	The management parameter (<i>see page 229</i>), consisting of four words.

Do not copy this array during a switchover from a primary to a standby CPU in a Hot Standby system. Uncheck the **Exchange On STBY** variable in Unity Pro when you configure a Hot Standby system.

NOTE: Refer to the description of Hot Standby system data management and the T_M_ECPU_HSBY DDT (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*) in the M580 Hot Standby System Planning Guide (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*).

Output Parameters

Parameter	Data type	Description
ENO	BOOL	This parameter is optional. When you select this output you also get the EN input. ENO output is activated upon successful execution of the function block.
Received_Data	Array [n...m] of INT	The EtherNet/IP (CIP) response (<i>see page 234</i>) or the Modbus TCP response (<i>see Modicon M340, BMX NOC 0401 Ethernet Communication Module, User Manual</i>). The structure and content depends upon the specific protocol.

Configuring the DATA_EXCH Management Parameter

Introduction

The structure and content of the management parameter of the `DATA_EXCH` block is common to both EtherNet/IP and Modbus TCP explicit messaging.

Configuring the Management Parameter

The management parameter consists of four contiguous words:

Data source	Register	Description	
		High Byte (MSB)	Low Byte (LSB)
Data managed by the system	Management_Param[0]	Exchange number	Two read-only bits: <ul style="list-style-type: none"> ● Bit 0 = Activity bit (<i>see page 230</i>) ● Bit 1 = Cancel bit
	Management_Param[1]	Operation report (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>)	Communication report (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>)
Data managed by the user	Management_Param[2]	Block timeout. Values include: <ul style="list-style-type: none"> ● 0 = infinite wait ● other values = timeout x 100 ms, for example: <ul style="list-style-type: none"> ○ 1 = 100 ms ○ 2 = 200 ms 	
	Management_Param[3]	Length of data sent or received: <ul style="list-style-type: none"> ● Input (before sending the request): length of data in the <code>Data_to_Send</code> parameter, in bytes ● Output (after response): length of data in the <code>Received_Data</code> parameter, in bytes 	

Activity Bit

The activity bit is the first bit of the first element in the table. The value of this bit indicates the execution status of the communication function:

- **1:** The bit is set to 1 when the function launches.
- **0:** The bit returns to 0 upon the completion of the execution. (The transition from 1 to 0 increments the exchange number. If an error is detected during the execution, search for the corresponding code in the operation and communication report (*see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*).)

For example, you can make this declaration in the management table:

```
Management_Param[0] ARRAY [0..3] OF INT
```

For that declaration, the activity bit corresponds to this notation:

```
Management_Param[0].0
```

NOTE: The notation previously used requires configuration of the project properties in such a way as to authorize the extraction of bits on integer types. If this is not the case, `Management_Param[0].0` cannot be accessed in this manner.

Explicit Messaging Services

Overview

Every explicit message performs a service. Each service is associated with a service code. Identify the explicit messaging service by its name, decimal number, or hexadecimal number.

You can execute explicit messages using the `DATA_EXCH` function block in the Unity Pro DTM.

Services

The services available in Unity Pro include, but are not limited to, these service codes:

Service Code		Description	Available in...	
Hex	Dec		DATA_EXCH block	Unity Pro GUI
1	1	Get_Attributes_All	X	X
2	2	Set_Attributes_All	X	X
3	3	Get_Attribute_List	X	—
4	4	Set_Attribute_List	X	—
5	5	Reset	X	X
6	6	Start	X	X
7	7	Stop	X	X
8	8	Create	X	X
9	9	Delete	X	X
A	10	Multiple_Service_Packet	X	—
B-C	11-12	(Reserved)	—	—
D	13	Apply_Attributes	X	X
E	14	Get_Attribute_Single	X	X
10	16	Set_Attribute_Single	X	X
11	17	Find_Next_Object_Instance	X	X
14	20	Error Response (DeviceNet only)	—	—
15	21	Restore	X	X
16	22	Save	X	X
17	23	No Operation (NOP)	X	X
18	24	Get_Member	X	X
19	25	Set_Member	X	X
1A	26	Insert_Member	X	X
"X" indicates the service is available. "—" indicates the service is not available.				

Service Code		Description	Available in...	
Hex	Dec		DATA_EXCH block	Unity Pro GUI
1B	27	Remove_Member	X	X
1C	28	GroupSync	X	—
1D-31	29-49	(Reserved)	—	—
"X" indicates the service is available. "—" indicates the service is not available.				

Configuring EtherNet/IP Explicit Messaging Using DATA_EXCH

Configuring the Address Parameter

To configure the Address parameter, use the `ADDM` function to convert the character string, described below, to an address that is input into the `ADR` parameter of the `DATA_EXCH` block:

`ADDM('rack.slot.channel{ip_address}message_type.protocol')`, where:

This field...	Represents...
rack	the number assigned to the rack containing the communication module
slot	the position of the communication module in the rack
channel	the communication channel—set to a value of 0
ip_address	the IP address of the remote device, for example 193.168.1.6
message_type	the type of message, presented as a three character string—either: <ul style="list-style-type: none"> ● UNC (indicating an unconnected message), or ● CON (indicating a connected message)
protocol	the protocol type—the three character string CIP

Configuring the Data_to_Send Parameter

The `Data_to_Send` parameter varies in size. It consists of contiguous registers that include—in sequence—both the message type and the CIP request:

Offset (words)	Length (bytes)	Data Type	Description
0	2 bytes	Bytes	Message type: <ul style="list-style-type: none"> ● High byte = size of the request in words ● Low byte = EtherNet/IP service code
1	Management_Param[3] (size of <code>Data_to_Send</code>) minus 2	Bytes	The CIP request ¹ . NOTE: The structure and size of the CIP request depends on the EtherNet/IP service.
1 Structure the CIP request in little endian order.			

Contents of the Received_Data Parameter

The `Received_Data` parameter contains only the CIP response. The length of the CIP response varies, and is reported by `Management_Param[3]` after the response is received. The format of the CIP response is described, below:

Offset (words)	Length (bytes)	Data Type	Description
0	2	Byte	<ul style="list-style-type: none"> High byte (MSB) = reserved Low byte (LSB): reply service
1	2	Byte	<ul style="list-style-type: none"> High byte (MSB): length of additional status Low byte (LSB): EtherNet/IP general status (see <i>Modicon M340, BMX NOC 0401 Ethernet Communication Module, User Manual</i>)
2	length of additional status	Byte array	Additional Status ¹
...	<code>Management_Param[3]</code> (size of <code>Received_Data</code>) minus 4, and minus the additional status length	Byte array	Response data
¹ Refer to <i>The CIP Networks Library, Volume 1, Common Industrial Protocol</i> at section 3-5.6 <i>Connection Manager Object Instance Error Codes</i> .			

NOTE: The response is structured in little endian order.

Checking the Received_Data Response for System and CIP Status

Use the contents of the `Received_Data` parameter to check both the system status and the CIP status of the Ethernet communication module when handling the explicit message.

First: Check the value of the high byte (MSB) of the first response word, positioned at offset 0. If the value of this byte is:

- equal to 0: the system properly handled the explicit message
- not equal to 0: a system-based event occurred

Refer to the list of EtherNet/IP Explicit Messaging Event Codes (see *Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures*) for an explanation of the system-based event code contained in the second response word, positioned at offset 1.

Next: If the system properly handled the explicit message, and the high byte of the first response word equals 0, check the value of the second response word, positioned at offset 1. If the value of this word is:

- equal to 0: the explicit message was properly handled by the CIP protocol
- not equal to 0: a CIP protocol-based event occurred

Refer to your CIP documentation for an explanation of the CIP status displayed in this word.

EtherNet/IP Explicit Message Example: Get_Attribute_Single

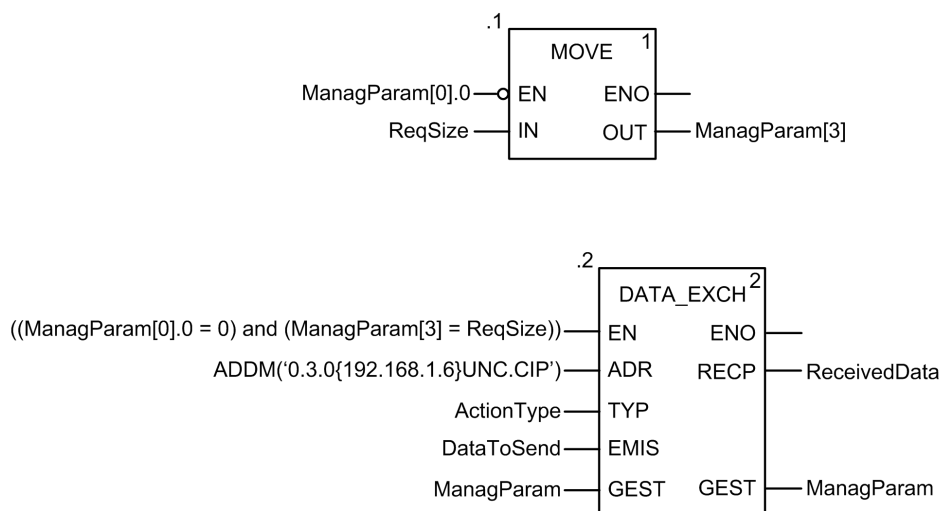
Overview

The following unconnected explicit messaging example shows you how to use the `DATA_EXCH` function block to retrieve diagnostic data from a remote device (at IP address 192.168.1.6). This example is executing a `Get_Attribute_Single` of assembly instance 100, attribute 3.

You can perform the same explicit messaging service using the **EtherNet/IP Explicit Message** window (see *Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*).

Implementing the DATA_EXCH Function Block

To implement the `DATA_EXCH` function block, create and assign variables for the following blocks:



Configuring the Address Variable

The Address variable identifies the explicit message originating device (in this example, the communication module) and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because we are not bridging through another PLC station. As an example, use the `ADDM` function to convert the following character string to an address:

`ADDM('0.1.0{192.168.1.6}UNC.CIP')`, where:

- rack = 0
- module (slot number) = 1
- channel = 0
- remote device IP address = 192.168.1.6
- message type = unconnected
- protocol = CIP

Configuring the ActionType Variable

The ActionType variable identifies the function type for the `DATA_EXCH` function block:

Variable	Description	Value (hex)
ActionType	Transmission followed by wait for response	16#01

Configuring the DataToSend Variable

The DataToSend variable identifies the details of the CIP explicit message request:

Variable	Description	Value (hex)
DataToSend[0]	CIP request service information: <ul style="list-style-type: none">• High byte = request size in words: 16#03 (3 decimal)• Low byte = service code: 16#0E (14 decimal)	16#030E
DataToSend[1]	CIP request class information: <ul style="list-style-type: none">• High byte = class: 16#04 (4 decimal)• Low byte = class segment: 16#20 (32 decimal)	16#0420
DataToSend[2]	CIP request instance information: <ul style="list-style-type: none">• High byte = instance: 16#64 (100 decimal)• Low byte = instance segment: 16#24 (36 decimal)	16#6424
DataToSend[3]	CIP request attribute information: <ul style="list-style-type: none">• High byte = attribute: 16#03 (3 decimal)• Low byte = attribute segment: 16#30 (48 decimal)	16#0330

Viewing the Response

Use a Unity Pro Animation table to display the ReceivedData variable array. Note that the ReceivedData variable array consists of the entire data buffer.

To display the CIP response, follow these steps:

Step	Action
1	In Unity Pro, select Tools → Project Browser to open the Project Browser.
2	In the Project Browser, select the Animation Tables folder, then click the right mouse button. A pop-up menu appears.
3	Select New Animation Table in the pop-up menu. A new animation table and its properties dialog both open.
4	In the Properties dialog, edit the following values:
	Name Type in a table name. For this example: ReceivedData .
	Functional module Accept the default <None> .
	Comment (Optional) Type your comment here.
	Number of animated characters Type in 100 , representing the size of the data buffer in words.
5	Click OK to close the dialog.
6	In the animation table's Name column, type the name of the variable assigned to the RECP pin: ReceivedData and press Enter . The animation table displays the ReceivedData variable.
7	Expand the ReceivedData variable to display its word array, where you can view the CIP response contained in the ReceivedData variable. NOTE: Each array entry presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, '8E' in word[0] is the lower byte, and '00' is the upper byte.

EtherNet/IP Explicit Message Example: Read Modbus Object

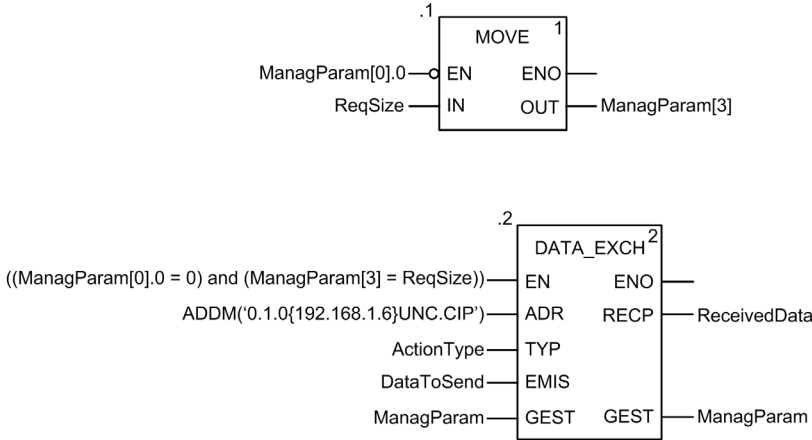
Overview

The following unconnected explicit messaging example shows you how to use the `DATA_EXCH` function block to read data from a remote device (for example, the STB NIP 2212 network interface module at IP address 192.168.1.6) using the Read_Holding_Registers service of the Modbus Object.

You can perform the same explicit messaging service using the **EtherNet/IP Explicit Message** window (see *Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*).

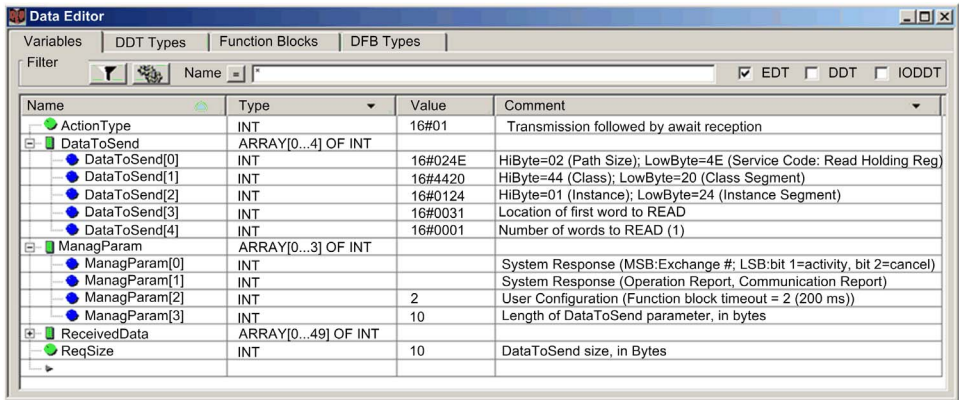
Implementing the DATA_EXCH Function Block

To implement the `DATA_EXCH` function block, you need to create and assign variables for the following blocks:



Declaring Variables

In this example, the following variables were defined. You can, of course, use different variable names in your explicit messaging configurations.



Configuring the Address Variable

The Address variable identifies the explicit message originating device (in this example, the Ethernet communication module) and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because we are not bridging through another PLC station. Use the ADDM function to convert the following character string to an address:

ADDM('0.1.0{192.168.1.6}UNC.CIP'), where:

- rack = 0
- module (slot number) = 1
- channel = 0
- remote device IP address = 192.168.1.6
- message type = unconnected
- protocol = CIP

Configuring the ActionType Variable

The ActionType variable identifies the function type for the DATA_EXCH function block:

Variable	Description	Value (hex)
ActionType	Transmission followed by wait for response	16#01

Configuring the DataToSend Variable

The DataToSend variable identifies the type of explicit message and the CIP request:

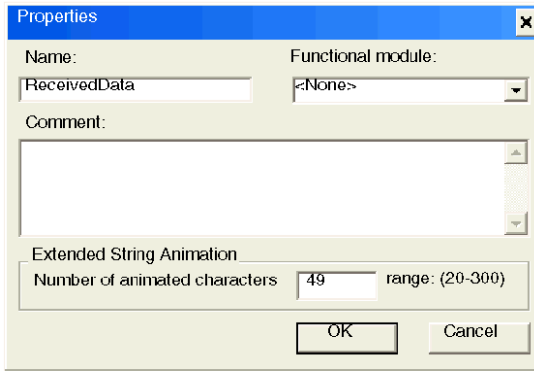
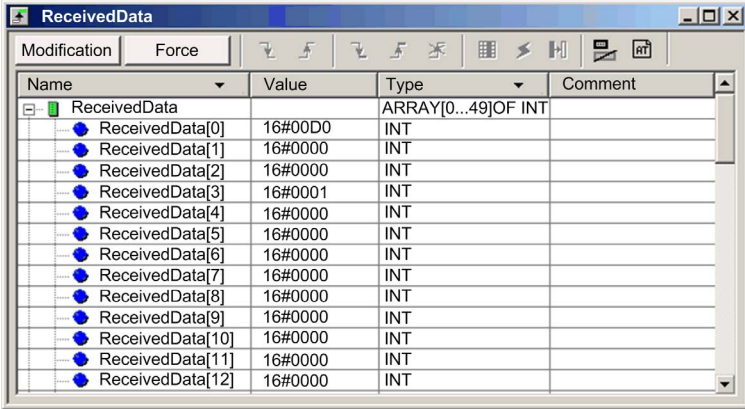
Variable	Description	Value (hex)
DataToSend[0]	CIP request service information: <ul style="list-style-type: none">High byte = request size in words: 16#02 (2 decimal)Low byte = service code: 16#4E (78 decimal)	16#024E
DataToSend[1]	CIP request class information: <ul style="list-style-type: none">High byte = class: 16#44 (68 decimal)Low byte = class segment: 16#20 (32 decimal)	16#4420
DataToSend[2]	CIP request instance information: <ul style="list-style-type: none">High byte = instance: 16#01 (1 decimal)Low byte = instance segment: 16#24 (36 decimal)	16#0124
DataToSend[3]	Location of first word to be read: <ul style="list-style-type: none">High byte = 16#00 (0 decimal)Low byte = 16#31 (49 decimal)	16#0031
DataToSend[4]	Number of words to read: <ul style="list-style-type: none">High byte = attribute: 16#00 (0 decimal)Low byte = attribute segment: 16#01 (1 decimal)	16#0001

Viewing the Response

Use a Unity Pro Animation table to display the ReceivedData variable array. Note that the ReceivedData variable array consists of the entire data buffer.

To display the CIP response, follow these steps:

Step	Action
1	In Unity Pro, select Tools → Project Browser to open the Project Browser.
2	In the Project Browser, select the Animation Tables folder, then click the right mouse button. A pop-up menu appears.
3	Select New Animation Table in the pop-up menu. A new animation table and its properties dialog both open.
4	In the Properties dialog, edit the following values:
	Name Type in a table name. For this example: ReceivedData .
	Functional module Accept the default <None> .
	Comment (Optional) Type your comment here.
	Number of animated characters Type in 49 , representing the size of the data buffer in words.

Step	Action
5	<p>The completed Properties dialog looks like this:</p>  <p>Click OK to close the dialog.</p>
6	<p>In the animation table's Name column, type in the name of the variable assigned to the RECP pin: ReceivedData and hit Enter. The animation table displays the ReceivedData variable.</p>
7	<p>Expand the ReceivedData variable to display its word array, where you can view the CIP response contained in the ReceivedData variable:</p>  <p>Note: Each array entry presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, 'CE' in word[0] is the lower byte, and '00' is the upper byte.</p>

EtherNet/IP Explicit Message Example: Write Modbus Object

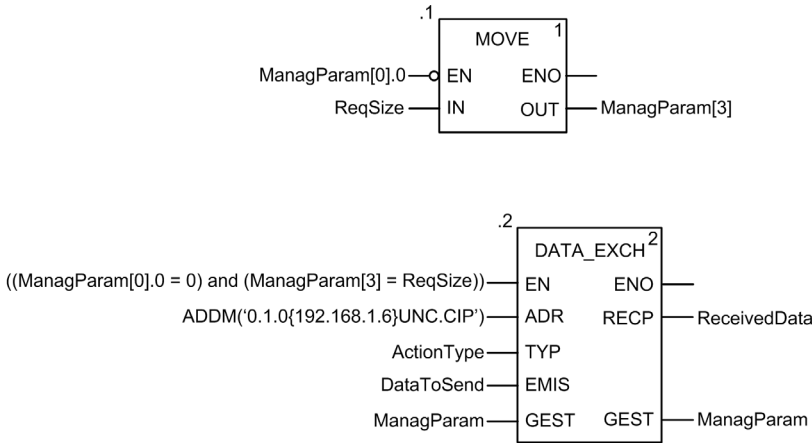
Overview

The following unconnected explicit messaging example shows you how to use the `DATA_EXCH` function block to write data to a remote device at IP address 192.168.1.6 using the `Write_Holding_Registers` service of the Modbus object.

You can perform the same explicit messaging service using the **EtherNet/IP Explicit Message** window (see *Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*) in the Unity Pro DTM.

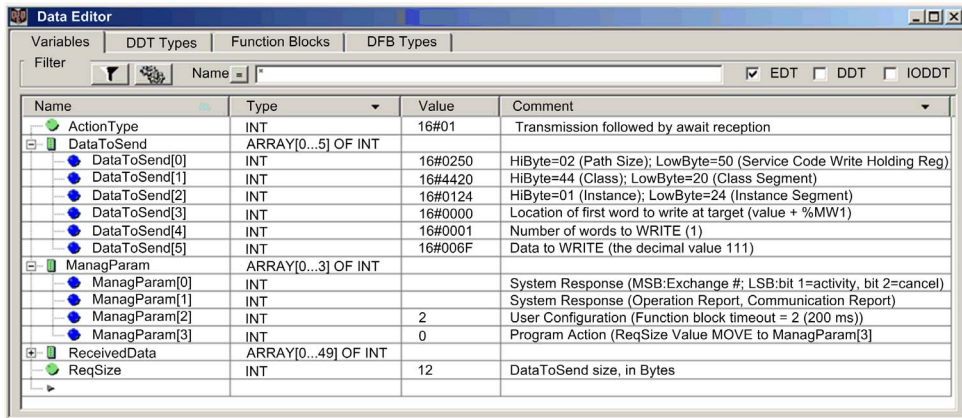
Implementing the DATA_EXCH Function Block

To implement the `DATA_EXCH` function block, you need to create and assign variables for the following blocks:



Declaring Variables

In this example, the following variables were defined. You can, of course, use different variable names in your explicit messaging configurations.



Configuring the Address Variable

The Address variable identifies the explicit message originating device (in this example, the communication module) and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because we are not bridging through another PLC station. Use the `ADDMM` function to convert the following character string to an address:

`ADDMM('0.1.0{192.168.1.6}UNC.CIP')`, where:

- rack = 0
- module (slot number) = 1
- channel = 0
- remote device IP address = 192.168.1.6
- message type = unconnected
- protocol = CIP

Configuring the ActionType Variable

The ActionType variable identifies the function type for the `DATA_EXCH` function block:

Variable	Description	Value (hex)
ActionType	Transmission followed by wait for response	16#01

Configuring the DataToSend Variable

The DataToSend variable identifies the type of explicit message and the CIP request:

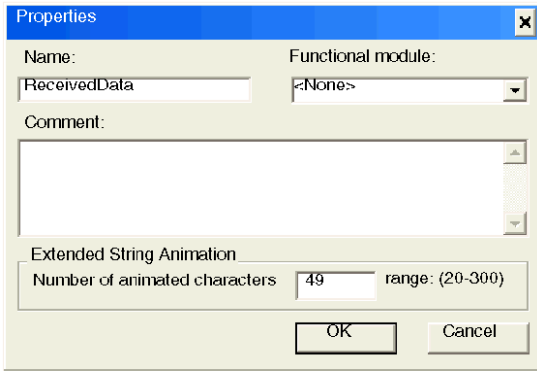
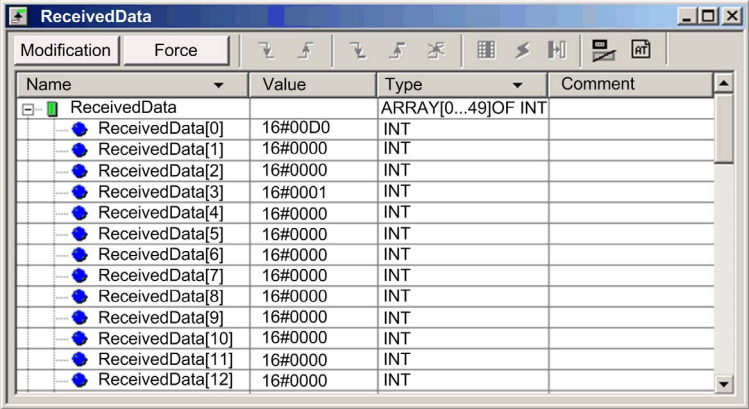
Variable	Description	Value (hex)
DataToSend[0]	CIP request service information: <ul style="list-style-type: none"> High byte = request size in words: 16#02 (2 decimal) Low byte = service code: 16#50 (80 decimal) 	16#0250
DataToSend[1]	CIP request class information: <ul style="list-style-type: none"> High byte = class: 16#44 (68 decimal) Low byte = class segment: 16#20 (32 decimal) 	16#4420
DataToSend[2]	CIP request instance information: <ul style="list-style-type: none"> High byte = instance: 16#01 (1 decimal) Low byte = instance segment: 16#24 (36 decimal) 	16#0124
DataToSend[3]	Location of first word to write (+ %MW1): <ul style="list-style-type: none"> High byte = 16#00 (0 decimal) Low byte = 16#00 (0 decimal) 	16#0000
DataToSend[4]	Number of words to write: <ul style="list-style-type: none"> High byte = attribute: 16#00 (0 decimal) Low byte = attribute segment: 16#01 (1 decimal) 	16#0001
DataToSend[5]	Data to write: <ul style="list-style-type: none"> High byte = attribute: 16#00 (0 decimal) Low byte = attribute segment: 16#6F (111 decimal) 	16#006F

Viewing the Response

Use a Unity Pro Animation table to display the ReceivedData variable array. Note that the ReceivedData variable array consists of the entire data buffer.

To display the CIP response, follow these steps:

Step	Action								
1	In Unity Pro, select Tools → Project Browser to open the Project Browser.								
2	In the Project Browser, select the Animation Tables folder, then click the right mouse button. A pop-up menu appears.								
3	Select New Animation Table in the pop-up menu. A new animation table and its properties dialog both open.								
4	In the Properties dialog, edit the following values: <table border="1" data-bbox="285 1276 1229 1448"> <tr> <td>Name</td><td>Type in a table name. For this example: ReceivedData.</td></tr> <tr> <td>Functional module</td><td>Accept the default <None>.</td></tr> <tr> <td>Comment</td><td>(Optional) Type your comment here.</td></tr> <tr> <td>Number of animated characters</td><td>Type in 49, representing the size of the data buffer in words.</td></tr> </table>	Name	Type in a table name. For this example: ReceivedData .	Functional module	Accept the default <None> .	Comment	(Optional) Type your comment here.	Number of animated characters	Type in 49 , representing the size of the data buffer in words.
Name	Type in a table name. For this example: ReceivedData .								
Functional module	Accept the default <None> .								
Comment	(Optional) Type your comment here.								
Number of animated characters	Type in 49 , representing the size of the data buffer in words.								

Step	Action
5	<p>The completed Properties dialog looks like this:</p>  <p>Click OK to close the dialog.</p>
6	<p>In the animation table's Name column, type in the name of the variable assigned to the RECP pin: ReceivedData and hit Enter. The animation table displays the ReceivedData variable.</p>
7	<p>Expand the ReceivedData variable to display its word array, where you can view the CIP response contained in the ReceivedData variable:</p>  <p>Note: Each array entry presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, 'D0' in word[0] is the lower byte, and '00' is the upper byte.</p>

Modbus TCP Explicit Messaging Function Codes

Overview

You can execute Modbus TCP explicit messages using either a Unity Pro `DATA_EXCH` function block or the Modbus Explicit Message Window.

NOTE: Configuration edits made to an Ethernet module are not saved to the operating parameters stored in the CPU and, therefore, are not sent by the CPU to the module on startup.

Function Codes

The function codes supported by the Unity Pro graphical user interface include the following standard explicit messaging functions:

Function Code (dec)	Description
1	Read bits (%M)
2	Read input bits (%I)
3	Read words (%MW)
4	Read input words (%IW)
15	Write bits (%M)
16	Write words (%MW)

NOTE: You can use the `DATA_EXCH` function block to execute any Modbus function, via program logic. Because the available function codes are too numerous to list here, refer instead to the Modbus IDA website for more information about these Modbus functions, at <http://www.Modbus.org>.

Configuring Modbus TCP Explicit Messaging Using DATA_EXCH

Introduction

When you use the `DATA_EXCH` block to create an explicit message for a Modbus TCP device, configure this block the same way you would configure it for any other Modbus communication. Refer to the Unity Pro online help for instructions on how to configure the `DATA_EXCH` block.

Configuring ADDM Block Unit ID Settings

When you configure the `DATA_EXCH` block, use the `ADDM` block to set the `DATA_EXCH` block's Address parameter. The `ADDM` block presents the configuration format `ADDM('rack.slot.channel[ip_address]UnitID.message_type.protocol')` where:

Parameter	Description
rack	the number assigned to the rack containing the communication module
slot	the position of the communication module in the rack
channel	the communication channel (set to a value of 0)
ip_address	the IP address of the remote device (for example, 192.168.1.7)
Unit ID	the destination node address, also known as the Modbus Plus on Ethernet Transporter (MET) mapping index value
message_type	the three-character string TCP
protocol	the three-character string MBS

The Unit ID value in a Modbus message indicates the destination of the message.

Refer to the Modbus diagnostic codes.

Contents of the Received_Data Parameter

The `Received_Data` parameter contains the Modbus response. The length of the response varies, and is reported by `Management_Param[3]` after the response is received. The format of the Modbus response is described, below:

Offset (words)	Length (bytes)	Description
0	2	First word of the Modbus response: <ul style="list-style-type: none">● High byte (MSB):<ul style="list-style-type: none">○ if successful: Modbus Function Code○ if not: Modbus function code + 16#80● Low byte (LSB):<ul style="list-style-type: none">○ if successful: depends on the request○ if not: Modbus exception code
1	Length of the <code>Received_Data</code> parameter – 2	Remainder of the Modbus response: depends on the specific Modbus request)

NOTE:

- Structure the response in little endian order.
- In some cases of detected errors, `Received_Data` is also used to judge the type of detected error along with `Management_Param`.

Modbus TCP Explicit Message Example: Read Register Request

Introduction

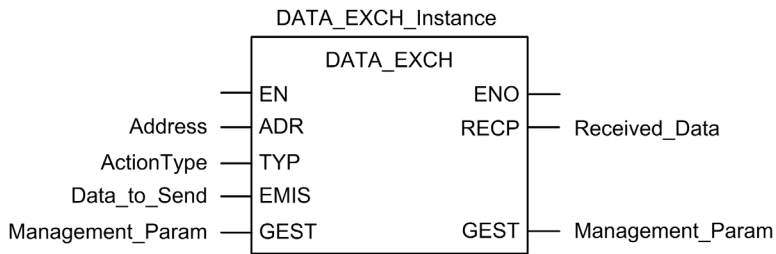
Use the `DATA_EXCH` function block to send a Modbus TCP explicit message to a remote device at a specific IP address to read a single word located in the remote device.

The `Management_Param`, the `Data_to_Send`, and the `Received_Data` parameters define the operation.

`EN` and `ENO` can be configured as additional parameters.

Implementing the DATA_EXCH Function Block

To implement the `DATA_EXCH` function block, create and assign variables for the for following:



Configuring the Address Variable

The Address variable identifies the explicit message originating device and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because you are not bridging through another PAC station. Use the `ADDM` function to convert the following character string to an address:

`ADDM('0.1.0{192.168.1.7}TCP.MBS')`, where:

- rack = 0
- module (slot number) = 1
- channel = 0
- remote device IP address = 192.168.1.7
- message type = TCP
- protocol = Modbus

Configuring the ActionType Variable

The `ActionType` variable identifies the function type for the `DATA_EXCH` function block:

Variable	Description	Value (hex)
ActionType	Transmission followed by wait for response	16#01

Configuring the DataToSend Variable

The DataToSend variable contains the target register address and the number of registers to read:

Variable	Description	Value (hex)
DataToSend[0]	<ul style="list-style-type: none"> High byte = Most significant byte (MSB) of register address 16#15 (21 decimal) Low byte = function code: 16#03 (03 decimal) 	16#1503
DataToSend[1]	<ul style="list-style-type: none"> High byte = Most significant byte (MSB) of the number of registers to read: 16#00 (0 decimal) Low byte = Least significant byte (LSB) of register address: 16#0F (15 decimal) 	16#000F
DataToSend[2]	CIP request instance information: <ul style="list-style-type: none"> High byte = not used: 16#00 (0 decimal) Low byte = Least significant byte (LSB) of the number of registers to read: 16#01 (1 decimal) 	16#0001

NOTE: For detailed information about M580 network topologies, refer to the *Modicon M580 Standalone System Planning Guide for Frequently Used Architectures* and *Modicon M580 System Planning Guide for Complex Topologies*.

Viewing the Response

Use a Unity Pro Animation table to display the ReceivedData variable array. Note that the ReceivedData variable array consists of the entire data buffer.

To display the Modbus TCP response, follow these steps:

Step	Action	
1	In Unity Pro, select Tools → Project Browser .	
2	In the Project Browser, select the Animation Tables folder, and click the right mouse button. Result: A pop-up menu appears.	
3	Select New Animation Table in the pop-up menu. Result: A new animation table and its properties dialog open.	
4	In the Properties dialog, edit the following values:	
	Name	Type in a table name. For this example: ReceivedData .
	Functional module	Accept the default <None> .
	Comment	(Optional) Type your comment here.
	Number of animated characters	Type in 100 , representing the size of the data buffer in words.
5	Click OK to close the dialog.	

Step	Action
6	<p>In the animation table's Name column, type in the name of the variable assigned to the databuffer: ReceivedData and presst Enter.</p> <p>Result: The animation table displays the ReceivedData variable.</p>
7	<p>Expand the ReceivedData variable to display its word array, where you can view the CIP response contained in the ReceivedData variable.</p> <p>NOTE: Each array entry presents 2 bytes of data in little endian format. For example, '03' in word[0] is the low byte, and '02' is the high byte.</p>

Sending Explicit Messages to EtherNet/IP Devices

Introduction

Use the **EtherNet/IP Explicit Message** window to send an explicit message from Unity Pro to the M580 CPU.

An explicit message can be connected or unconnected:

- **connected:** A connected explicit message contains both path information and a connection identifier to the target device.
- **unconnected:** An unconnected message requires path (addressing) information that identifies the destination device (and, optionally, device attributes).

You can use explicit messaging to perform many different services. Not every EtherNet/IP device supports every service.

Accessing the Page

Before you can perform explicit messaging, connect the DTM for the M580 CPU to the CPU itself:

Step	Action
1	Open the DTM Browser in Unity Pro (Tools → DTM Browser).
2	Select the M580 DTM in the DTM Browser .
3	Right-click the M580 DTM.
4	Scroll to the EtherNet/IP explicit messaging page (Device menu → Additional functions → EtherNet/IP Explicit Message).

Configuring Settings

Configure the explicit message using these settings on the **EtherNet/IP Explicit Messaging** page:

Field	Setting
Address	IP Address: The IP address of the target device that is used to identify the target of the explicit message.
	Class: The Class integer (1 ... 65535) is the identifier of the target device that is used in the construction of the message path.
	Instance: The Instance integer (0 ... 65535) is the class instance of the target device that is used in the construction of the message path.
	Attribute: Check this box to enable the Attribute integer (0 ... 65535), which is the specific device property that is the target of the explicit message that is used in the construction of the message path.
Service	Number: The Number is the integer (1 ... 127) associated with the service to be performed by the explicit message.
	NOTE: If you select Custom Service as the named service, type in a service number. This field is read-only for all other services.
	Name: Select the service that the explicit message is intended to perform.
	Enter Path(hex): Check this box to enable the message path field, where you can manually enter the entire path to the target device.
Data(hex)	Data(hex): This value represents the data to be sent to the target device for services that send data.
Messaging	Connected: Select this radial button to make the connection.
	Unconnected: Select this radial button to end the connection.
Response(hex)	The Response area contains the data sent to the configuration tool by the target device in hexadecimal format.
Status	The Status area displays messages that indicate whether or not the explicit message has succeeded.
Button	Send to Device: When your explicit message is configured, click Send to Device .

Click the **Close** button to save the changes and close the window.

Sending Explicit Messages to Modbus Devices

Introduction

Use the Modbus explicit messaging window to send an explicit message from Unity Pro to the M580 CPU.

You can use explicit messaging to perform many different services. Not every Modbus TCP device supports every service.

Accessing the Page

Before you can perform explicit messaging, connect the DTM for the M580 CPU to the CPU itself:

Step	Action
1	Open the DTM Browser in Unity Pro (Tools → DTM Browser).
2	Select the M580 DTM in the DTM Browser .
3	Right-click the M580 DTM.
4	Scroll to the EtherNet/IP explicit messaging page (Device menu → Additional functions → Modbus Explicit Message).

Configuring Settings

Configure the explicit message using these settings on the **Modbus Explicit Messaging** page:

Field	Setting
Address	IP Address: The IP address of the target device that is used to identify the target of the explicit message.
	Start Address: This setting is a component of the addressing path.
	Quantity: This setting is a component of the addressing path.
	Read Device Id Code: This read-only code represents the service that the explicit message is intended to perform.
	Object Id: This read-only identifier specifies the object that the explicit message is intended to access.
	Unit Id: This integer represents the device or module that is the target of the connection: <ul style="list-style-type: none">● 255: (default): Use this value to access the M580 CPU itself.● 0 ... 254: Use these values to identify the device number of the target device behind a Modbus TCP to Modbus gateway.
Service	Number: This integer (0 ... 255) represents the service to be performed by the explicit message.
	Name: Select the integer (0 ... 255) that represents the service that the explicit message is intended to perform.
Data	Data(hex): This value represents the data to be sent to the target device for services that send data.

Field	Setting
Response	The Response area displays any data sent to the configuration tool by the target device in hexadecimal format.
Status	The Status area displays messages indicating whether or not the explicit message has succeeded.
Button	Send to Device: After your explicit message is configured, click Send to Device .

Click the **Close** button to save the changes and close the window.

Section 5.10

Explicit Messaging Using the MBP_MSTR Block in Quantum RIO Drops

Introduction

This section shows you how to configure both EtherNet/IP and Modbus TCP explicit messages in Quantum RIO drops by including the MBP_MSTR function block in the logic of your Unity Pro project.

What Is in This Section?

This section contains the following topics:

Topic	Page
Configuring Explicit Messaging Using MBP_MSTR	257
EtherNet/IP Explicit Messaging Services	259
Configuring the CONTROL and DATABUF Parameters	261
MBP_MSTR Example: Get_Attributes_Single	263
Modbus TCP Explicit Messaging Function Codes	268
Configuring the Control Parameter for Modbus TCP Explicit Messaging	269

Configuring Explicit Messaging Using MBP_MSTR

Overview

You can use the `MBP_MSTR` function block to configure both Modbus TCP and EtherNet/IP connected and unconnected explicit messages.

The operation begins when the input to the `EN` pin is turned ON. The operation ends if the `ABORT` pin is turned ON, or if the `EN` pin is turned OFF.

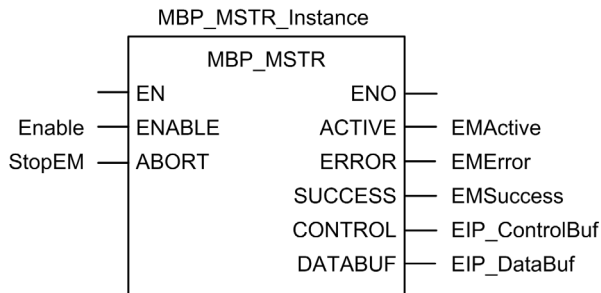
The `CONTROL` and `DATABUF` output parameters define the operation.

NOTE: The structure and content of the `CONTROL` and `DATABUF` output parameters differ for explicit messages configured using the EtherNet/IP and Modbus TCP protocols. Refer to the topics [Configuring the Control Parameter for EtherNet/IP](#) and [Configuring the Control Parameter for Modbus TCP](#) for instructions on how to configure these parameters for each protocol.

The `ACTIVE` output turns ON during operation; the `ERROR` output turns ON if the operation aborts without success; the `SUCCESS` output turns ON at the successful completion of the operation.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD



Input Parameters

Parameter	Data type	Description
ENABLE	BOOL	When ON, the explicit message operation (specified in the first element of the <code>CONTROL</code> pin) is executing.
ABORT	BOOL	When ON, the operation is aborted.

Output Parameters

Parameter	Data type	Description
ACTIVE	BOOL	ON when the operation is active. OFF at all other times.
ERROR	BOOL	ON when the operation is aborted without success. OFF before operation, during operation, and if operation succeeds.
SUCCESS	BOOL	ON when the operation concludes successfully. OFF before operation, during operation, and if operation does not conclude successfully.
CONTROL ¹	WORD	This parameter contains the control block. The first element contains a code describing the operation to be performed. The content of the control block depends on the operation. The structure of the control block depends on the protocol (EtherNet/IP or Modbus TCP). Note: Assign this parameter to a located variable.
DATABUF ¹	WORD	This parameter contains the data buffer. For operations that: <ul style="list-style-type: none"> ● provide data — e.g., a write operation — this parameter is the data source ● receive data — e.g., a read operation — this parameter is the data destination Note: Assign this parameter to a located variable.
<p>1. Refer to the topics Configuring the Control Block for EtherNet/IP and Configuring the Control Block for Modbus TCP for instructions on how to configure these parameters for the EtherNet/IP and Modbus TCP communication protocols.</p>		

EtherNet/IP Explicit Messaging Services

Overview

Every EtherNet/IP explicit message performs a service. Each service is associated with a service code (or number). You will need to identify the explicit messaging service by its name, decimal number, or hexadecimal number.

You can execute EtherNet/IP explicit messages using either a Unity Pro MBP_MSTR function block or the Unity Pro Ethernet Configuration Tool's **EtherNet/IP Explicit Message Window**.

NOTE: Configuration edits made to an Ethernet communication module from the Unity Pro Ethernet Configuration Tool's EtherNet/IP Explicit Message Window are not saved to the operating parameters stored in the CPU and, therefore, are not sent by the CPU to the module on startup.

You can use Unity Pro to construct a request that executes any service supported by the target device that is compliant with the EtherNet/IP protocol.

Services

The services supported by Unity Pro include the following standard explicit messaging services:

Service Code		Description	Available in...	
Hex	Dec		MBP_MSTR block	Unity Pro GUI
1	1	Get_Attributes_All	X	X
2	2	Set_Attributes_All	X	X
3	3	Get_Attribute_List	X	—
4	4	Set_Attribute_List	X	—
5	5	Reset	X	X
6	6	Start	X	X
7	7	Stop	X	X
8	8	Create	X	X
9	9	Delete	X	X
A	10	Multiple_Service_Packet	X	—
D	13	Apply_Attributes	X	X
E	14	Get_Attribute_Single	X	X
10	16	Set_Attribute_Single	X	X
11	17	Find_Next_Object_Instance	X	X
14	20	Detected Error Response (DeviceNet only)	—	—
15	21	Restore	X	X
"X" = the service is available. "—" = the service is not available.				

Service Code		Description	Available in...	
Hex	Dec		MBP_MSTR block	Unity Pro GUI
16	22	Save	X	X
17	23	No Operation (NOP)	X	X
18	24	Get_Member	X	X
19	25	Set_Member	X	X
1A	26	Insert_Member	X	X
1B	27	Remove_Member	X	X
1C	28	GroupSync	X	—
"X" = the service is available. "—" = the service is not available.				

Configuring the CONTROL and DATABUF Parameters

Overview

The **CONTROL** and **DATABUF** output parameters define the operation performed by the MBP_MSTR function block. For the EtherNet/IP protocol, the structure of the **CONTROL** and **DATABUF** output parameters remains the same for every explicit messaging service ([see page 259](#)).

Configuring the Control Parameter

The Control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[0]	Operation	<ul style="list-style-type: none"> 14 = unconnected 270 = connected
CONTROL[1]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only).
CONTROL[2]	Data buffer length	Data buffer length, in words
CONTROL[3]	Response offset	Offset for the beginning of the response in the data buffer, in 16-bit words Note: To avoid overwriting the request, confirm that the response offset value is greater than the request length CONTROL[7].
CONTROL[4]	Slot	High byte = slot location on backplane Low byte = 0 (not used)
CONTROL[5] ¹	IP address	High byte = byte 4 of the IP address (MSB)
		Low byte = byte 3 of the IP address
CONTROL[6] ¹		High byte = byte 2 of the IP address
		Low byte = byte 1 of the IP address (LSB)
CONTROL[7]	Request length	Length of the CIP request, in bytes
CONTROL[8]	Response length	Length of the response received, in bytes Read only—set after completion
1. For example, the Control parameter handles the IP address 192.168.1.6 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 6.		

Configuring the Data Buffer

The data buffer varies in size. It consists of contiguous registers that include—in sequence—both the CIP request and the CIP response. To avoid overwriting the request, confirm that the data buffer is large enough to simultaneously contain both the request and response data.

Data Buffer: Variable size: set in <code>CONTROL[2]</code>	CIP Request: Request size: set in <code>CONTROL[7]</code>
	CIP Response: Starting position: set in <code>CONTROL[3]</code> Response size: reported in <code>CONTROL[8]</code>
	NOTE: If the response offset is smaller than the request size, the response data overwrites part of the request.

The format of the data buffer's CIP request and CIP response is described, below.

NOTE: Structure both the request and response in little endian order.

Request:

Byte offset	Field	Data type	Description
0	Service	Byte	Service of the explicit message
1	Request_Path_Size	Byte	The number of words in the Request_Path field
2	Request_Path	Padded EPATH	This byte array describes the path of the request—including class ID, instance ID, etc.—for this transaction
...	Request_Data	Byte array	Service specific data to be delivered in the explicit message request—if none, this field is empty

Response:

Byte offset	Field	Data type	Description
0	Reply Service	Byte	Service of the explicit message + 16#80
1	Reserved	Byte	0
2	General Status	Byte	EtherNet/IP General Status (<i>see Modicon M340, BMX NOC 0401 Ethernet Communication Module, User Manual</i>)
3	Size of Additional Status	Byte	Additional Status array size—in words
4	Additional Status	Word array	Additional status ¹
...	Response Data	Byte array	Response data from request, or additional detected error data if General Status indicates a detected error
1. Refer to <i>The CIP Networks Library, Volume 1, Common Industrial Protocol</i> at section 3-5.6 <i>Connection Manager Object Instance Detected Error Codes</i> ,			

MBP_MSTR Example: Get_Attributes_Single

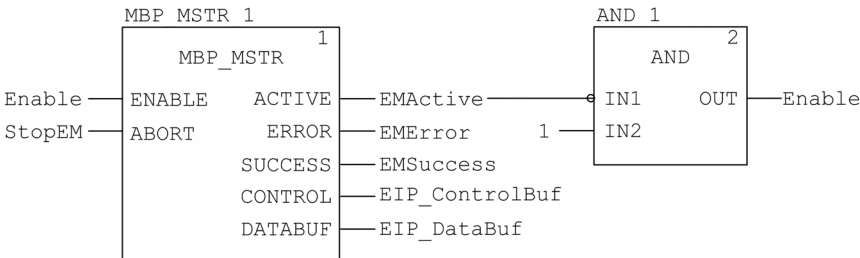
Overview

The following unconnected explicit messaging example shows you how to use the MBP_MSTR function block to retrieve diagnostic information for an STB island from an STB NIC 2212 network interface module, by using the Get_Attributes_Single service.

You can perform the same explicit messaging service using the **EtherNet/IP Explicit Message Window** of the Unity Pro Ethernet Configuration Tool (see *Quantum EIO, Control Network, Installation and Configuration Guide*).

Implementing the MBP_MSTR Function Block

To implement the MBP_MSTR function block, you need to create and assign variables, then connect it to an AND block. In the following example, the logic will continuously send an explicit message upon receiving notice of success:



Input Variables

Variables need to be created and assigned to input pins. For the purpose of this example, variables have been created — and named — as described below. (You can use different variable names in your explicit messaging configurations.)

Input Pin	Variable	Data Type
ENABLE	Enable	BOOL
ABORT	StopEM	BOOL

Output Variables

Variables also need to be created and assigned to output pins. (The names assigned to output variables apply only to this example, and can be changed in your explicit messaging configurations.)

Output Pin	Variable	Data Type
ACTIVE	EMActive	BOOL
ERROR	EMError	BOOL
SUCCESS	EMSuccess	BOOL
CONTROL	EIP_ControlBuf	Array of 10 WORDS
DATABUF	EIP_DataBuf	Array of 100 WORDS

NOTE: To simplify configuration, you can assign the `CONTROL` and `DATABUF` output pins to a byte array consisting of located variables. When configured in this manner, you will not need to be aware of the location of data within a word (for example, high versus low byte, and big or little endian format).

Control Array

The control array parameter (`EIP_ControlBuf`) consists of 9 contiguous words. You need to configure only some control words; other control words are read-only and are written to by the operation. In this example, the control array defines the operation as an unconnected explicit message, and identifies the target device:

Register	Description	Configure	Setting (hex)
CONTROL[0]	Operation: High byte = <ul style="list-style-type: none"> 00 (unconnected), or 01 (connected) Low byte = 0E (CIP explicit message)	Yes	16#000E (unconnected)
CONTROL[1]	Detected error status: read-only (written by operation)	No	16#0000
CONTROL[2]	Data buffer length = 100 words	Yes	16#0064
CONTROL[3]	Response offset: offset — in words — for the beginning of the explicit message response in the databuffer	Yes	16#0004
CONTROL[4]	High byte = slot location of the communication module in the backplane Low byte = 0 (not used)	Yes	16#0400
CONTROL[5] ¹	IP address of the Ethernet communication module: High byte = byte 4 of the IP address Low byte = byte 3 of the IP address	Yes	16#C0A8

Register	Description	Configure	Setting (hex)
CONTROL[6] ¹	IP address of the Ethernet communication module: High byte = byte 2 of the IP address Low byte = byte 1 of the IP address	Yes	16#0106
CONTROL[7]	CIP request length (in bytes)	Yes	16#0008
CONTROL[8]	Length of received response (written by operation)	No	16#0000
1. In this example, the control parameter handles the IP address 192.168.1.6 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 6.			

CIP Request

The CIP request is located at the beginning of the databuffer and is followed by the CIP response. In this example, the CIP request calls for the return of a single attribute value (diagnostic data), and describes the request path through the target device's object structure leading to the target attribute:

Request word	High byte		Low byte	
	Description	Value (hex)	Description	Value (hex)
1	Request path size (in words)	16#03	EM Service: Get_Attributes_Single	16#0E
2	Request path: class assembly object	16#04	Request path: logical class segment	16#20
3	Request path: instance	16#64	Request path: logical instance segment	16#24
4	Request path: attribute	16#03	Request path: logical attribute segment	16#30

Combining the high and low bytes, above, the CIP request would look like this:

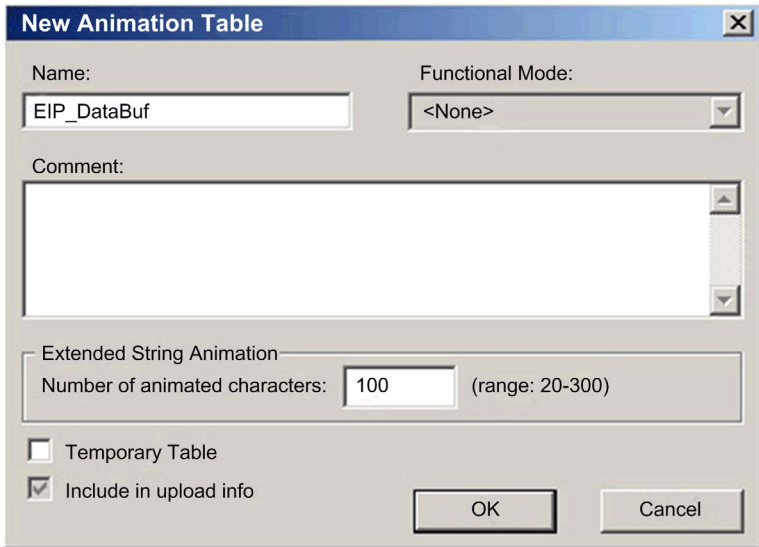
Request word	Value
1	16#030E
2	16#0420
3	16#6424
4	16#0330

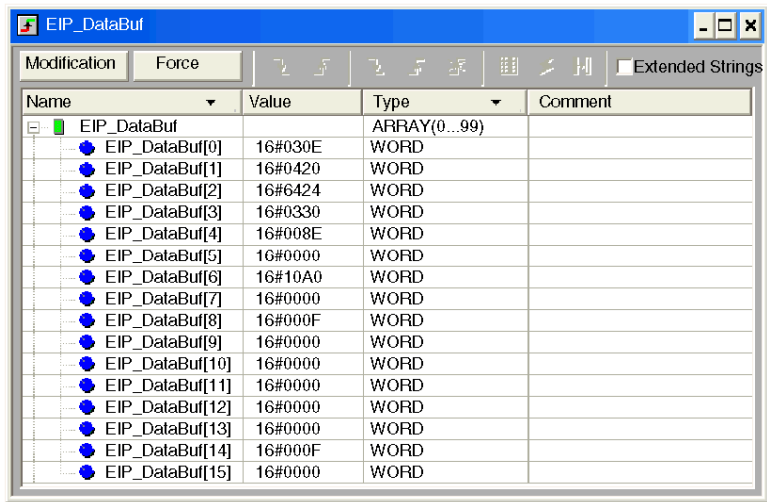
Viewing the Response

Use a Unity Pro Animation table to display the EIP_DataBuf variable array. Note that the EIP_DataBuf variable array consists of the entire data buffer, which includes the:

- CIP request (4 words) located in EIP_DataBuf(1-4)
- CIP service type (1 word) located in EIP_DataBuf(5)
- CIP request status (1 word) located in EIP_DataBuf(6)
- CIP response (in this case, 10 words) located in EIP_DataBuf(7-16)

To display the CIP response, follow these steps:

Step	Action								
1	In Unity Pro, select Tools → Project Browser to open the Project Browser .								
2	In the Project Browser , right-click Animation Tables → New Animation Table . Result: A new animation table opens.								
3	<div>In the New Animation Table dialog, edit the following values:</div> <table><tr><td>Name</td><td>Type in a table name. For this example: EIP_DataBuf.</td></tr><tr><td>Functional Mode</td><td>Accept the default <None>.</td></tr><tr><td>Comment</td><td>Leave blank.</td></tr><tr><td>Number of animated characters</td><td>Type 100, representing the size of the data buffer in words.</td></tr></table>	Name	Type in a table name. For this example: EIP_DataBuf .	Functional Mode	Accept the default <None> .	Comment	Leave blank.	Number of animated characters	Type 100 , representing the size of the data buffer in words.
Name	Type in a table name. For this example: EIP_DataBuf .								
Functional Mode	Accept the default <None> .								
Comment	Leave blank.								
Number of animated characters	Type 100 , representing the size of the data buffer in words.								
4	<div>The completed dialog looks like this:</div> <div></div> <div>Click OK to close the dialog.</div>								

Step	Action																																																																								
5	In the animation table's Name column, type in the name of the variable assigned to the databuffer: EIP_DataBuf and press Enter . The animation table displays the EIP_DataBuf variable.																																																																								
6	Expand the EIP_DataBuf variable to display its word array, where you can view the CIP response at words EIP_DataBuf(7-16):  <table><thead><tr><th>Name</th><th>Value</th><th>Type</th><th>Comment</th></tr></thead><tbody><tr><td>EIP_DataBuf</td><td></td><td>ARRAY(0...99)</td><td></td></tr><tr><td>EIP_DataBuf[0]</td><td>16#030E</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[1]</td><td>16#0420</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[2]</td><td>16#6424</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[3]</td><td>16#0330</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[4]</td><td>16#008E</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[5]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[6]</td><td>16#10A0</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[7]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[8]</td><td>16#000F</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[9]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[10]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[11]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[12]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[13]</td><td>16#0000</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[14]</td><td>16#000F</td><td>WORD</td><td></td></tr><tr><td>EIP_DataBuf[15]</td><td>16#0000</td><td>WORD</td><td></td></tr></tbody></table>	Name	Value	Type	Comment	EIP_DataBuf		ARRAY(0...99)		EIP_DataBuf[0]	16#030E	WORD		EIP_DataBuf[1]	16#0420	WORD		EIP_DataBuf[2]	16#6424	WORD		EIP_DataBuf[3]	16#0330	WORD		EIP_DataBuf[4]	16#008E	WORD		EIP_DataBuf[5]	16#0000	WORD		EIP_DataBuf[6]	16#10A0	WORD		EIP_DataBuf[7]	16#0000	WORD		EIP_DataBuf[8]	16#000F	WORD		EIP_DataBuf[9]	16#0000	WORD		EIP_DataBuf[10]	16#0000	WORD		EIP_DataBuf[11]	16#0000	WORD		EIP_DataBuf[12]	16#0000	WORD		EIP_DataBuf[13]	16#0000	WORD		EIP_DataBuf[14]	16#000F	WORD		EIP_DataBuf[15]	16#0000	WORD	
Name	Value	Type	Comment																																																																						
EIP_DataBuf		ARRAY(0...99)																																																																							
EIP_DataBuf[0]	16#030E	WORD																																																																							
EIP_DataBuf[1]	16#0420	WORD																																																																							
EIP_DataBuf[2]	16#6424	WORD																																																																							
EIP_DataBuf[3]	16#0330	WORD																																																																							
EIP_DataBuf[4]	16#008E	WORD																																																																							
EIP_DataBuf[5]	16#0000	WORD																																																																							
EIP_DataBuf[6]	16#10A0	WORD																																																																							
EIP_DataBuf[7]	16#0000	WORD																																																																							
EIP_DataBuf[8]	16#000F	WORD																																																																							
EIP_DataBuf[9]	16#0000	WORD																																																																							
EIP_DataBuf[10]	16#0000	WORD																																																																							
EIP_DataBuf[11]	16#0000	WORD																																																																							
EIP_DataBuf[12]	16#0000	WORD																																																																							
EIP_DataBuf[13]	16#0000	WORD																																																																							
EIP_DataBuf[14]	16#000F	WORD																																																																							
EIP_DataBuf[15]	16#0000	WORD																																																																							

Note: Each word presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, '0E' in EIP_DataBuf[0] is the low byte, and '03' is the high byte.

Modbus TCP Explicit Messaging Function Codes

Overview

Every Modbus TCP explicit message performs a function. Each function is associated with a code (or number). You will need to identify the explicit messaging function by its name, decimal number, or hexadecimal number.

You can execute Modbus TCP explicit messages using either a Unity Pro MBP_MSTR function block or the Unity Pro Ethernet Configuration Tool's **Modbus Explicit Message Window**.

NOTE: Configuration edits made to an Ethernet communication module from the Unity Pro Ethernet Configuration Tool are not saved to the operating parameters stored in the CPU and, therefore, are not sent by the CPU to the module on startup.

Services

The function codes supported by Unity Pro include the following standard explicit messaging functions:

Function Code		Description	Available in...	
Hex	Dec		MBP_MSTR block	Unity Pro GUI
1	1	Write data	X	X
2	2	Read data	X	X
3	3	Get local statistics	X	X
4	4	Clear local statistics	X	X
7	7	Get remote statistics	X	X
8	8	Clear remote statistics	X	X
A	10	Reset module	X	X
17	23	Read / write data	X	X
FFF0	65520	Enable / disable HTTP and FTP/TFTP services	X	-
"X" = the service is available. "—" = the service is not available.				

Configuring the Control Parameter for Modbus TCP Explicit Messaging

Overview

The `CONTROL` and `DATABUF` output parameters define the operation performed by the `MBP_MSTR` (see page 257) function block. For the Modbus TCP protocol, both the structure and the content of the `CONTROL` output parameter vary, depending upon the function code (see page 268).

The structure of the `CONTROL` parameter is described, below, for each supported function code.

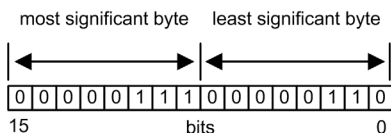
Refer to the *Quantum Ethernet I/O System Planning Guide* for an example of an `MSTR` block created in a Unity Pro application to read the ports of a dual-ring switch (DRS) to diagnose a sub-ring break.

Control Parameter Routing Register

The `CONTROL[5]` routing register specifies the source and destination node addresses for network data transfer, and consists of the following 2 bytes:

- Most Significant Byte (MSB): contains the source node address, for example, the slot number of the 140 NOC 78• 00
- Least Significant Byte (LSB): contains the destination node address — a value representing either a direct or a bridge address. The LSB is required for devices that are reached through a bridge, for example, an Ethernet to Modbus bridge or an Ethernet to Modbus Plus bridge. The values of the LSB are as follows:
 - If no bridge is used: LSB is set to zero(0).
 - If a bridge is used: LSB contains the Modbus Plus on Ethernet Transporter (MET) mapping index value. This value, also known as the Unit ID, indicates the device to which the message is directed.

The `CONTROL[5]` routing register:



When the Ethernet communication module acts as a server, the LSB indicates the destination of a message received by the communication module:

- messages with an LSB value from 0 to 254 are forwarded to and processed by the CPU
- messages with an LSB value of 255 are retained and processed by the Ethernet communication module

NOTE: Unit ID 255 should be used when requesting diagnostic data from the Ethernet communication module.

Write Data

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	1 = write data
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	Data buffer length	Number of addresses sent to the slave
CONTROL[4]	Starting register	Start address of the slave to which the data is written, in 16-bit words
CONTROL[5]	Routing register	High byte = Ethernet communication module slot
		Low byte = MBP on Ethernet transporter (MET) mapping index
CONTROL[6] ¹	IP address	Byte 4 of the IP address (MSB)
CONTROL[7] ¹		Byte 3 of the IP address
CONTROL[8] ¹		Byte 2 of the IP address
CONTROL[9] ¹		Byte 1 of the IP address (LSB)
1. For example, the control parameter handles the IP address 192.168.1.7 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 7.		

Read Data

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	2 = read data
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	Data buffer length	Number of addresses to be read from the slave
CONTROL[4]	Starting register	Determines the %MW starting register in the slave from which the data is read. For example: 1 = %MW1, 49 = %MW49)
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index

Register	Function	Description
CONTROL[6] ¹	IP address	Byte 4 of the IP address (MSB)
CONTROL[7] ¹		Byte 3 of the IP address
CONTROL[8] ¹		Byte 2 of the IP address
CONTROL[9] ¹		Byte 1 of the IP address (LSB)
1. For example, the control parameter handles the IP address 192.168.1.7 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 7.		

Get Local Statistics

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	3 = read local statistics
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	Data buffer length	Number of addresses to be read from local statistics (0...37)
CONTROL[4]	Starting register	First address from which the statistics table is read (Reg1=0)
CONTROL[5]	Routing register	High byte = Ethernet communication module slot
		Low byte = MBP on Ethernet transporter (MET) mapping index
CONTROL[6]	(not used)	—
CONTROL[7]		
CONTROL[8]		
CONTROL[9]		

Module Response: A TCP/IP Ethernet module responds to the `Get Local Statistics` command with the following information:

Word	Description			
00...02	MAC Address			
03	Board Status — this word contains the following bits:			
	Bit 15	0 = Link LED off; 1 = Link LED ON	Bit 3	Reserved
	Bits 14...13	Reserved	Bit 2	0 = half duplex; 1 = full duplex
	Bit 12	0 = 10 Mbit; 1 = 100 Mbit	Bit 1	0 = not configured; 1 = configured
	Bits 11...9	Reserved	Bit 0	0 = PLC not running; 1 = PLC or NOC running
	Bits 8...4	Module Type — this bit presents the following values:		
		<ul style="list-style-type: none"> ● 0 = NOE 2x1 ● 1 = ENT ● 2 = M1E ● 3 = NOE 771 00 ● 4 = ETY ● 5 = CIP ● 6 = (reserved) ● 7 = 140 CPU 651 x0 ● 8 = 140 CRP 312 00 ● 9 = (reserved) ● 10 = 140 NOE 771 10 	<ul style="list-style-type: none"> ● 11 = 140 NOE 771 01 ● 12 = 140 NOE 771 11 ● 13 = (reserved) ● 14 = 140 NOC 78• 00 ● 15...16 = (reserved) ● 17 = M340 CPU ● 18 = M340 NOE ● 19 = BMX NOC 0401 ● 20 = TSX ETC 101 ● 21 = 140 NOC 771 01 	
04 and 05	Number of receiver interrupts			
06 and 07	Number of transmitter interrupts			
08 and 09	Transmit_timeout detected error count			
10 and 11	Collision_detect error count			
12 and 13	Missed packets			
14 and 15	(reserved)			
16 and 17	Number of times driver has restarted			
18 and 19	Receive framing detected error			
20 and 21	Receiver overflow detected error			
22 and 23	Receive CRC detected error			
24 and 25	Receive buffer detected error			
26 and 27	Transmit buffer detected error			
28 and 29	Transmit silo underflow			
30 and 31	Late collision			

Word	Description
32 and 33	Lost carrier
34 and 35	Number of retries
36 and 37	IP address

Clear Local Statistics

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	4 = clear local statistics
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	(not used)	—
CONTROL[4]	(not used)	—
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index
CONTROL[6]	(not used)	—
CONTROL[7]		
CONTROL[8]		
CONTROL[9]		

Get Remote Statistics

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	7 = get remote statistics
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	Data buffer length	Number of addresses to be read from the statistics data field (0...37)
CONTROL[4]	Starting register	First address from which the node statistics table is read
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index

Register	Function	Description
CONTROL[6] ¹	IP address	Byte 4 of the IP address (MSB)
CONTROL[7] ¹		Byte 3 of the IP address
CONTROL[8] ¹		Byte 2 of the IP address
CONTROL[9] ¹		Byte 1 of the IP address (LSB)
1. For example, the control parameter handles the IP address 192.168.1.7 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 7.		

Clear Remote Statistics

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	8 = clear remote statistics
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	(not used)	—
CONTROL[4]	(not used)	—
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index
CONTROL[6] ¹	IP address	Byte 4 of the IP address (MSB)
CONTROL[7] ¹		Byte 3 of the IP address
CONTROL[8] ¹		Byte 2 of the IP address
CONTROL[9] ¹		Byte 1 of the IP address (LSB)
1. For example, the control parameter handles the IP address 192.168.1.7 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 7.		

Reset Module

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	10 = reset module
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	(not used)	—
CONTROL[4]	(not used)	—
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index
CONTROL[6]	(not used)	—
CONTROL[7]		
CONTROL[8]		
CONTROL[9]		

Read/Write Data

The control parameter consists of 11 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	23 = read / write data
CONTROL[2]	Detected error status	Holds the event code (<i>see Modicon M580 Standalone, System Planning Guide for, Frequently Used Architectures</i>) (read-only)
CONTROL[3]	Data buffer length	Number of addresses sent to the slave
CONTROL[4]	Starting register	Determines the %MW starting register in the slave to which the data will be written. For example: 1 = %MW1, 49 = %MW49)
CONTROL[5]	Routing register	High byte = Ethernet communication module slot Low byte = MBP on Ethernet transporter (MET) mapping index

Register	Function	Description
CONTROL[6] ¹	IP address	Byte 4 of the IP address (MSB)
CONTROL[7] ¹		Byte 3 of the IP address
CONTROL[8] ¹		Byte 2 of the IP address
CONTROL[9] ¹		Byte 1 of the IP address (LSB)
CONTROL[10]	Data buffer length	Number of addresses to be read from the slave
CONTROL[11]	Starting register	Determines the %MW starting register in the slave from which the data is read. For example: 1 = %MW1, 49 = %MW49)
1. For example, the control parameter handles the IP address 192.168.1.7 in the following order: Byte 4 = 192, Byte 3 = 168, Byte 2 = 1, Byte 1 = 7.		

Enable/Disable HTTP or FTP/TFTP Services

When HTTP or FTP/TFTP has been enabled using Unity Pro configuration tools (*see Quantum EIO, Control Network, Installation and Configuration Guide*), an MSTR block can be used to change the enabled state of the service while the application is running. The MSTR block cannot change the state of the HTTP or FTP/TFTP services if the service was disabled using one of the configuration tools.

The control parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[1]	Operation	FFF0 (hex) 65520 (dec) = enable / disable HTTP or FTP/TFTP
CONTROL[2]	Detected error status	Holds the event code (read-only). Codes returned include: 0x000 (Success): MSTR block with operational code 0xFFFF0 was called and the enabled state of HTTP or FTP/TFTP was changed. 0x5068 (Busy): MSTR block with operational code 0xFFFF0 was called within 2 seconds of the previous call (regardless of return code from previous call). 0x4001 (Same state): MSTR block with operational code 0xFFFF0 was called to change the enabled state of HTTP and FTP/TFTP to the states they were already in. 0x2004 (Invalid data): MSTR block with operational code 0xFFFF0 was called and the data in the control block did not match the specifications. 0x5069 (Disabled): If the HTTP or FTP/TFTP service was already disabled via the Unity Pro interface when the MSTR block with operational code 0xFFFF0 was called to change the state of the disabled service.
CONTROL[3]		Set this register to 1.
CONTROL[4]		
CONTROL[5]	Module slot number and destination ID	High byte = Module slot number communication module slot
		Low byte = Destination ID
CONTROL[6]	Request mode	Bit 0 (LSB) = 1: Enable FTP/TFTP Bit 0 (LSB) = 0: Disable FTP/TFTP Bit 1 = 1: Enable HTTP Bit 1 = 0: Disable HTTP
CONTROL[7]		Set this register to 0.
CONTROL[8]		
CONTROL[9]		

HTTP, FTP, and TFTP service state changes made by MSTR with operation code FFF0 (hex) are overridden by the configured value when the module is power-cycled or reset and when a new application is downloaded to the module.

Here are some examples:

State Configured By Unity Pro	Action attempted using MSTR with operation code FFF0 (hex)	Result
Disabled	Any	MSTR returns detected error code 0x5069 (service was already disabled by configuration)
Enabled	Disable	MSTR returns code 0x000 (success). <ul style="list-style-type: none">● Another MSTR block action enables the service--OR--● The module is reset or power-cycled--OR--● A new application is downloaded with the service disabled by configuration
	Enable	MSTR returns detected error code 0x4001 (same state). No change made.

Section 5.11

Implicit Messaging

Introduction

This section extends the sample Unity Pro application and contains these instructions:

- Add an STB NIC 2212 EtherNet/IP network interface module to your Unity Pro application.
- Configure the STB NIC 2212 module.
- Configure EtherNet/IP connections to link the Ethernet communications module and the STB NIC 2212 network interface module.
- Configure I/O items for the Advantys island.

NOTE: The instructions in this section describe an example of a single, specific device configuration. For other configuration choices, refer to the Unity Pro help files.

What Is in This Section?

This section contains the following topics:

Topic	Page
Setting Up Your Network	280
Adding an STB NIC 2212 Device	281
Configuring STB NIC 2212 Properties	283
Configuring EtherNet/IP Connections	285
Configuring I/O Items	290
EtherNet/IP Implicit Messaging	303

Setting Up Your Network

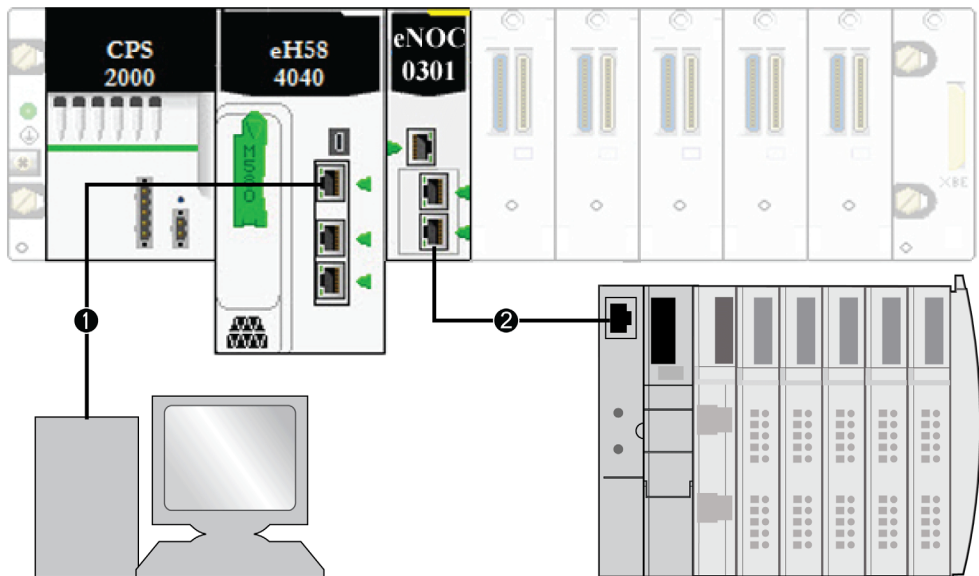
Introduction

Use this example to establish communications between the M580 rack and an Advantys STB NIC 2212 network interface module (NIM).

The STB NIC 2212 is Schneider Electric's EtherNet/IP network interface module for Advantys islands.

Network Topology

This sample network shows the Ethernet network devices used in this configuration:



- 1 The M580 CPU (with DIO scanner service) on the local rack is connected to a PC that runs the Unity Pro software.
- 2 The BMENOC0301/11 Ethernet communications module on the local rack is connected to an STB NIC 2212 NIM on an Advantys island.

To re-create this example, use the IP addresses from your own configuration for these items:

- M580 CPU
- PC
- BMENOC0301/11 Ethernet communication module
- STB NIC 2212 network interface module

Adding an STB NIC 2212 Device

Overview

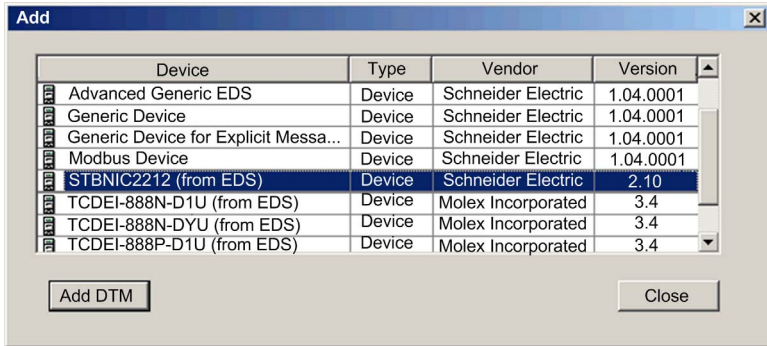
You can use the Unity Pro device library to add a remote device—in this example the STB NIC 2212 module—to your project. Only a remote device that is part of your Unity Pro device library can be added to your project.

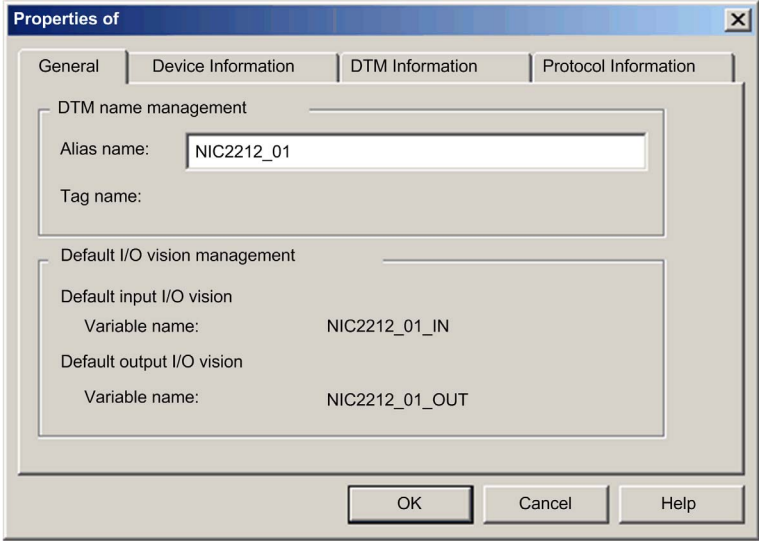
Alternatively, with a remote device already added to your device library, you can use automatic device discovery to populate your project. Perform automatic device discovery by using the **Field bus discovery** command with a communication module selected in the **DTM Browser**.

Adding an STB NIC 2212 Remote Device

NOTE: This example uses a device-specific DTM. If you do not have a device-specific DTM, Unity Pro provides a generic device DTM.

Add the STB NIC 2212 to your project:

Step	Action
1	In the DTM Browser , right-click the DTM that corresponds to the Ethernet communication module.
2	Scroll to Add .
3	<p>Select STBNIC2212 (from EDS):</p>  <p>NOTE: Click a column name to sort the list of available devices. (For example, click Device to view the items in the first column in alphabetical order.)</p>
4	Click the Add DTM button to see the association between the Ethernet communication module and the STB NIC 2212 in the DTM Browser .
5	In the DTM Browser , right-click the STB NIC 2212 node that is associated with the Ethernet communication module DTM.
6	Scroll to Properties .

Step	Action
7	<p>On the General tab, create a unique Alias name. (Using similar devices that use the same DTM can result in duplicate module names.) In this example, type in the name NIC2212_01:</p> <div></div> <p>Unity Pro uses the Alias name as the base for both structure and variable names.</p> <p>NOTE: The Alias name is the only editable parameter on this tab. The other parameters are read-only.</p>
8	<p>Click OK to add the STB NIC 2212 network interface module to the DTM Browser, beneath the communication module.</p>

The next step is to configure the device you have just added to the project.

Configuring STB NIC 2212 Properties

Introduction

Use Unity Pro to edit the settings for STB NIC 2212 device.

NOTE: To edit these settings, disconnect the DTM from a device.

Accessing the Device Properties

View the **Properties** tab:

Step	Action
1	Double-click the DTM that corresponds to the BMENOC0301/11 module to access the configuration.
2	In the navigation tree, expand the Device List (<i>see page 203</i>) to see the associated local slave instances.
3	Select the device that corresponds to the name NIC2212_01 .
4	Select the Properties tab.

These configuration tabs are available for the device:

- **Properties**
- **Address Setting**

Properties

Configure the **Properties** tab to perform these tasks:

- Add the STB NIC 2212 to the configuration.
- Remove the STB NIC 2212 from the configuration.
- Edit the base name for variables and data structures used by the STB NIC 2212.
- Indicate how input and output items are created and edited.

The descriptions for parameters (*see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*) in the **Properties** tab are described in the configuration chapter. Use these values and names from the sample configuration:

Field	Parameter	Description
Properties	Number	Accept the default.
	Active Configuration	Accept the default (Enabled).
IO Structure Name	Structure Name	Unity Pro automatically assigns a structure name based on the variable name.
	Variable Name	Variable Name: Accept the auto-generated variable name (based on the alias name).
	Default Name	Press this button to restore the default variable and structure names. For this example, custom names are used.

Field	Parameter	Description
Items Management	Import Mode	Select Manual .
	Reimport Items	Press this button to import the I/O items list from the device DTM, overwriting any manual I/O item edits. Enabled only when Import mode is set to Manual .

Click **Apply** to save your edits and leave the window open.

Address Setting

Use the **Address Setting** tab to enable the DHCP client in the STB NIC 2212 network interface module. When the DHCP client is enabled in the remote device, it obtains its IP address from the DHCP server in the Ethernet communication module.

Configure the **Address Setting** page to perform these tasks:

- Configure the IP address for a device.
- Enable or disable DHCP client software for a device.

The descriptions for parameters in the **Address Setting** tab are described in the configuration chapter. Use these values and names from the sample configuration:

Field	Parameter	Description
Change Address	IP Address	In our continuing example, type in the address 192.168.1.6 .
Address Server	DHCP for this Device	Select Enabled .
	Identified by	Select Device Name .
	Identifier	Accept the default setting of the STB NIC 2212 device (based on the Alias name).
	Mask	Accept the default value (255.255.0.0).
	Gateway	Configure the default value (192.168.10.1).

The next step is to configure the connection between the communication module and the remote device.

Configuring EtherNet/IP Connections

Overview

An EtherNet/IP connection provides a communication link between 2 or more devices. Properties for a single connection can be configured in the DTMs for the connected devices.

The following example presents settings for a connection between the CPU's DIO scanner service and a remote STB NIC 2212 network interface module. Configuration edits are made to the DTMs for each device.

When making DTM edits, disconnect the selected DTM from the actual module or device (see *Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*).

Accessing the Connection Information

View the connection information tabs:

Step	Action
1	In Unity Pro, double-click the DTM for the CPU's DIO scanner service to access the configuration.
2	In the navigation tree, expand the Device List (see <i>Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>) to see the associated local slave instances.
3	Expand (+) the device that corresponds to the STB NIC 2212 module.
4	Select Read Input/ Write Output Data to see the Connection Settings and Connection Information tabs.

Connection Settings

Unity Pro automatically creates a connection between a communication module and remote device when the remote device is added to the Unity Pro project. Thereafter, many edits to the connection can be made in the DTM for the remote device. However, some of the connection parameters can also be configured in the DTM for the communication module, as demonstrated below.

Edit these parameters on the **Connection Settings** tab. Use settings that are appropriate to your application:

Parameter	Description
Connection Bit	The (read-only) offset for both the health bit and the control bit for this connection. Offset values are auto-generated by the Unity Pro DTM.
Request Packet Interval (RPI)	The refresh period for this connection , from 2 to 65535 ms. Default = 12 ms. Type 30 ms. NOTE: This parameter can be set in the DTM for the communication module or the remote device.

Parameter	Description
Time-out Multiplier	This setting, multiplied against the RPI, produces a value that triggers an inactivity timeout. Setting selections include: x4, x8, x16, x32, x64, x128, x256 and x512. For this example, accept the default (x4).
Input Fallback Mode	This parameter describes the behavior of inputs in the application in the event communication is lost. Select Set to Zero .

Click **OK** to save your settings.

NOTE: The connection information page is read-only when the DTM is selected. This information needs to be set in the DTM for the remote device.

Configuring Connection Settings in the Remote Device DTM

Connections between the CPU's DIO scanner service and a remote device can be created and edited in the DTM for the remote device.

In this example, the following configuration edits are made to the connection that Unity Pro automatically created when the remote device was added to the project. Use settings that are appropriate for your actual application:

Step	Action
1	Open the DTM for the remote device by selecting it in the Device Editor .
2	Open the Device Editor : <ul style="list-style-type: none"> • Use the main menu (Edit → Open) ... <i>or</i> ... • Right-click and scroll to Open.
3	In the navigation pane (on the left side of the Device Editor), confirm that the remote device connection is of the type Read Input / Write Output Data . To view the connection type, select the STB NIC 2212 module in the left pane of the Device Editor . If the connection type is not of the type Read Input / Write Output Data , delete the existing connection and add a new one, as follows: <ol style="list-style-type: none"> With the connection selected in the left pane, click the Remove Connection button Result: The existing connection is removed. Click the Add Connection button. Result: The Select the connection to add dialog opens. Use the scroll buttons on the drop down list to display and select the Read Input / Write Output Data connection type. Click OK to close the Select the connection to add dialog. Result: The new connection node appears. Click Apply to save the new connection, leaving the Device Editor open for additional edits.

General Tab

This is the **General** tab of the DTM for the STB NIC 2212:

Group/Parameter	Value	Unit
RPI	30	ms
Input T -> O		
Input size	19	bytes
Input mode	Multicast	
Input type	Fixed	
Input priority	Scheduled	
Input trigger	Cyclic	
Output O -> T		
Output size	6	bytes
Output mode	Point to Point	
Output type	Fixed	
Output priority	Scheduled	

Description

OK Cancel Apply

Edit the settings in the **General** tab:

Parameter	Description
RPI	The refresh period for this connection. Accept the value of 30 ms. (This parameter can be set in the DTM for the communication module or the remote device.)
Input size	The number of bytes (0 ... 509) configured in the STB NIC 2212 module.
Input mode	Transmission type: <ul style="list-style-type: none"> ● Multicast ● Point to Point For this example, accept the default (Multicast).
Input type	Ethernet packet type (fixed or variable length) to be transmitted. (Only Fixed length packets are supported.)

Parameter	Description
Input priority	<p>The transmission priority value depends upon the device DTM. These are the available values:</p> <ul style="list-style-type: none"> • Low • High • Scheduled <p>For this example, accept the default selection (Scheduled).</p> <p>NOTE: For remote modules that support more than one priority value, you can use this setting to specify the order in which the Ethernet communication module handles packets. For more information, refer to the topic describing QoS packet prioritization (see <i>Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).</p>
Input trigger	<p>These are the available transmission trigger values:</p> <ul style="list-style-type: none"> • Cyclic • Change of state or application <p>For input I/O data, select Cyclic.</p>
Output size	The number of bytes configured in the STB NIC 2212 module in increments of 4 bytes (2 words).
Output mode	Accept the default (Point to Point).
Output type	(Read-only). Only Fixed length packets are supported.
Output priority	Accept the default (Scheduled).

Click **Apply** to save your settings and leave the window open.

Identity Check Tab

Configure the **Identity Check** page to set rules for comparing the identity of the network devices (as defined by their DTM or EDS files) against the identity of the actual network device.

This is the **Identity Check** tab:

Parameter	Value	Unit
▶ Check Identity	Disable	

Description

OK Cancel Apply

Use the **Check Identity** parameter to set the rules that the CPU's DIO scanner service uses to compare the configured versus the actual remote device:

- **Must match exactly:** The DTM or EDS file exactly matches the remote device.
- **Disable:** No checking occurs. The identity portion of the connection is filled with zero values (the default setting).
- **Must be compatible:** If the remote device is not the same as defined by the DTM/EDS, it emulates the DTM/EDS definitions.
- **None:** No checking occurs. The identity portion of the connection is omitted.
- **Custom:** Enable the following parameter settings, to be set individually.

Edit the settings in the **Identity Check** tab:

Parameter	Description
Compatibility Mode	True: For each of the following selected tests, the DTM/EDS and remote device need only be compatible.
	False: For each of the following selected tests, the DTM/EDS and remote device need to match exactly.
Compatibility Mode	Make a selection for each of these parameters: <ul style="list-style-type: none"> • Compatible: Include the parameter in the test. • Not checked: The parameter is not included in the test.
Minor Version	
Major Version	
Product Code	
Product Type	
Product Vendor	

Click **OK** to save your settings and close the window.

The next step is to configure I/O settings.

Configuring I/O Items

Overview

The final task in this example is to add I/O items to the configuration of the STB NIC 2212 and its eight I/O modules:

- Use the Advantys configuration software to identify the relative position of each I/O module's inputs and outputs.
- Use the Unity Pro **Device Editor** to create input and output items, defining each item's:
 - name
 - data type

I/O Item Types and Sizes

The goal is to create a collection of input items and output items that equal the input size and output size specified for the STB NIC 2212. In this example, items need to be created for:

- 19 bytes of inputs
- 6 bytes of outputs

The Unity Pro **Device Editor** provides great flexibility in creating input and output items. You can create input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit dwords, or 32-bit IEEE floating values. The number of items you create depends upon the data type and size of each item.

In the sample project, the following items were created:

- discrete bits for digital inputs and outputs
- 8-bit bytes or 16-bit words for analog inputs and outputs

Mapping Input and Output Items

Use the **Fieldbus Image** page of the **I/O Image Overview** window in the Advantys configuration software to identify the number and type of I/O items you need to create, as follows:

Step	Action
1	In the Advantys configuration software, select Island → I/O Image Overview . The I/O Image window opens to the Fieldbus Image page.
2	Select the first cell (word 1, cell 0) in the Input Data table to display (in the middle of the page) a description of the cell data and its source module.
3	Make a note of the word, bit(s), module and item information for that cell.
4	Repeat steps 2 and 3 for each cell containing either an S or an integer.

NOTE: The Fieldbus Image presents input and output data in the form of 16-bit words (starting with word 1). You need to rearrange this data for the Unity Pro Ethernet Configuration Tool, which presents the same data in the form of 8-bit bytes (starting with byte 0).

NOTE: When you create items, align items of data type **WORD** and **DWORD**:

- **WORD** items: align these items on a 16-bit boundary
- **DWORD** items: align these items on a 32-bit boundary.

This process yields the following tables of input and output data:

Input Data:

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	Bit(s)	Byte	Bit(s)		
1	0-15	0	0-7	NIC 2212	low byte status
		1	0-7		high byte status
2	0-1	2	0-1	DDI 3230	input data
	2-3		2-3	DDI 3230	input status
	4-5		4-5	DDO 3200	output data echo
	6-7		6-7	DDO 3200	output status
	8-11	3	0-3	DDI 3420	input data
	12-15		4-7	DDI 3420	input status
3	0-3	4	0-3	DDO 3410	output data echo
	4-7		4-7	DDO 3410	output status
	8-13	5	0-5	DDI 3610	input data
	14-15		6-7	NA	not used
4	0-5	6	0-5	DDI 3610	input status
	6-7		6-7	NA	not used
	8-13	7	0-5	DDO 3600	output data echo
	14-15		6-7	NA	not used
5	0-5	8	0-5	DDO 3600	output status
	6-15	8	6-7	NA	not used
		9	0-7		
6	0-15	10	0-7	AVI 1270	input data ch 1
		11	0-7		
7	0-7	12	0-7	AVI 1270	input status ch 1
	8-15	13	0-7	NA	not used
8	0-15	14	0-7	AVI 1270	input data ch 2
		15	0-7		
9	0-7	16	0-7	AVI 1270	input status ch 2
	8-15	17	0-7	AVO 1250	output status ch 1

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	Bit(s)	Byte	Bit(s)		
10	0-7	18	0-7	AVO 1250	output status ch 2
	8-15	NA	NA	NA	not used

Output Data:

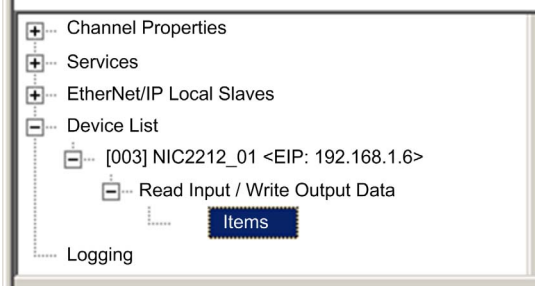
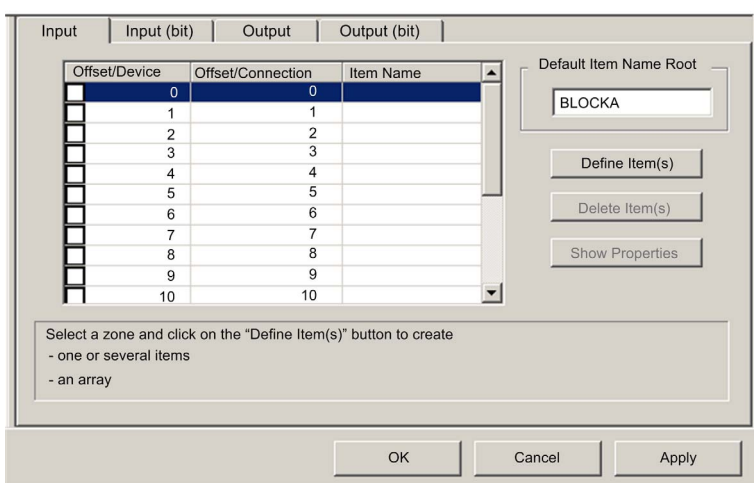
Advantys Fieldbus Image		Unity Pro EIP Items		Module	Description
Word	Bit(s)	Byte	Bit(s)		
1	0-1	0	0-1	DDO 3200	output data
	2-5		2-5	DDO 3410	output data
	6-7		6-7	NA	not used
	8-13	1	0-5	DDO 3600	output data
	14-15		6-7	NA	not used
2	0-15	2	0-7	AVO 1250	output data ch 1
		3	0-7		
3	0-15	4	0-7	AVO 1250	output data ch 2
		5	0-7		

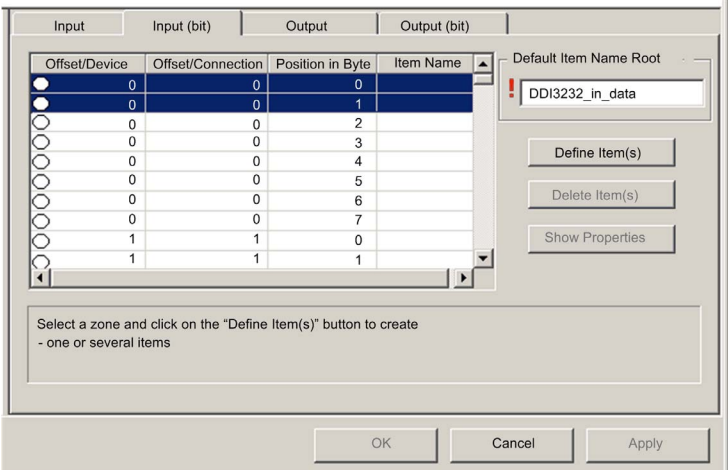
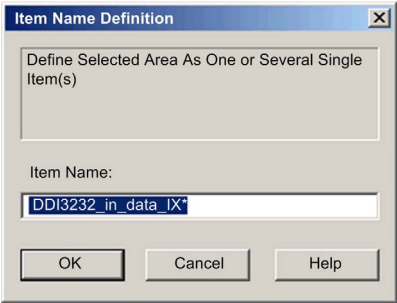
This example shows you how to create 19 bytes of inputs and 6 bytes of outputs. To efficiently use space, this example creates items in the following sequence:

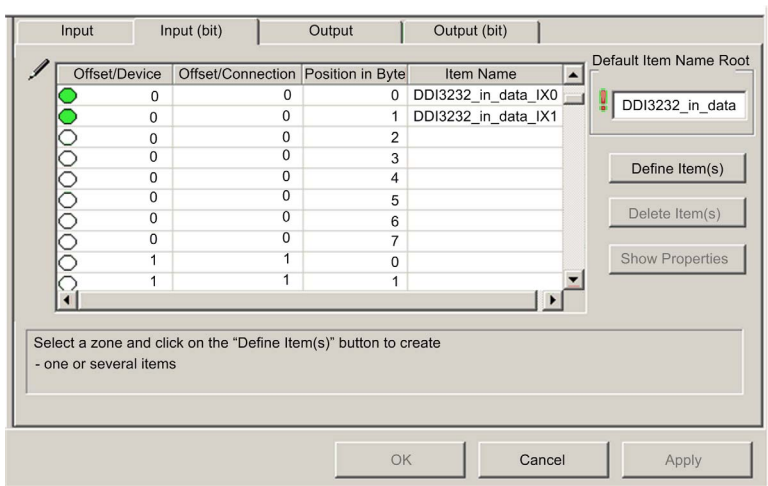
- input bit items
- input byte and word items
- output bit items
- output byte and word items

Creating Input Bit Items

To create input bit items for the STB NIC 2212 example, beginning with 16 discrete inputs for NIC 2212 status:

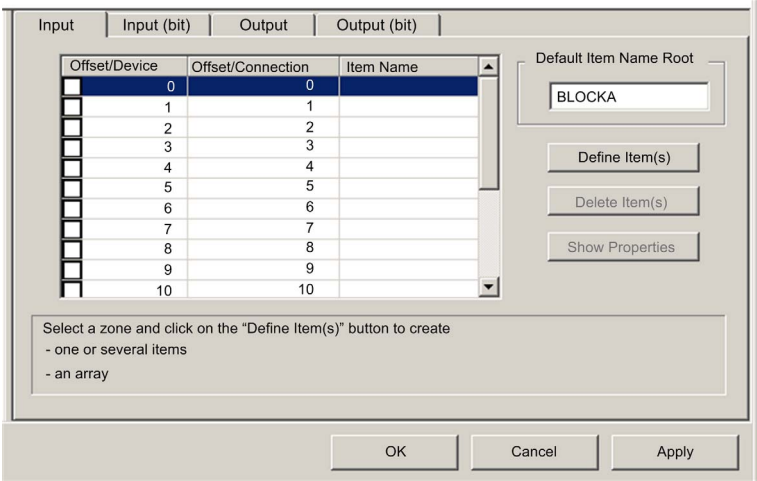
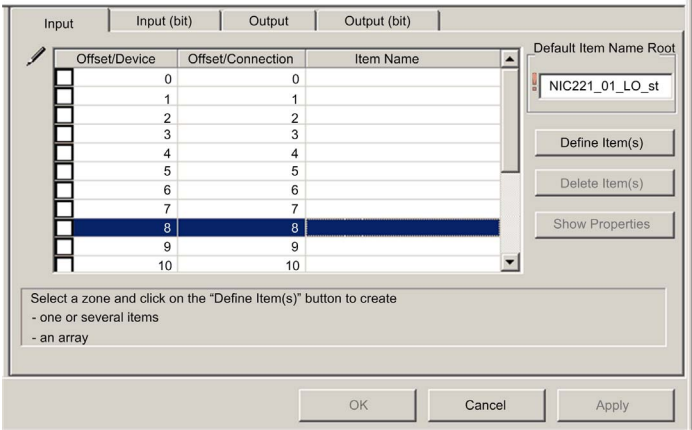
Step	Action
1	In the DTM Browser , select the DTM for the BMENOC0301/11.
2	Do one of the following: <ul style="list-style-type: none"> in the main menu, select Edit → Open. — or — Right-click and select Open in the pop-up menu. Result: The Device Editor opens, displaying the CPU DTM.
3	In the left pane of the Device Editor , navigate to and select the Items node for the STB NIC 2212 network interface module: 
4	The Items window opens: 

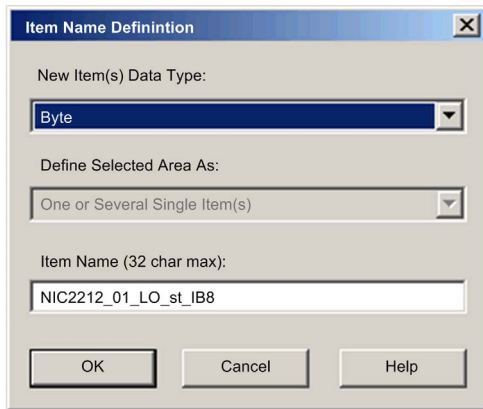
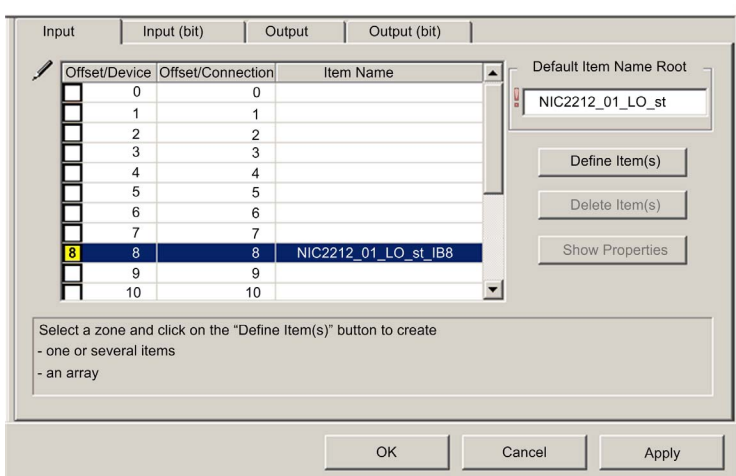
Step	Action
5	Select the Input (bit) tab to display that page.
6	In the Input (bit) page, type the following default root name (representing device status) into the Default Items Name Root input box type: DDI3232_in_data .
7	<div>In the Items List, select the first 2 rows in the table. (These rows represent bits 0-1 in byte.) </div>
8	<div>Click the Define Item(s) button. Result: The Item Name Definition dialog opens: </div> <p>NOTE: The asterisk (*) indicates that a series of discrete items with the same root name will be created.</p>

Step	Action
9	<p>Accept the default Item Name, and click OK. Result: 2 discrete input items are created:</p> 
10	Click Apply to save the items and leave the page open.
11	<p>Repeat steps 6 - 10 for each group of discrete input items you need to create. In this example, that includes items for each of the following groups:</p> <ul style="list-style-type: none"> ● Byte: 0, Bits: 2-3, Default Items Name Root: DDI3230_in_st ● Byte: 0, Bits: 4-5, Default Items Name Root: DDO3200_out_echo ● Byte: 0, Bits: 6-7, Default Items Name Root: DDO3200_out_st ● Byte: 1, Bits: 0-3, Default Items Name Root: DDI3420_in_data ● Byte: 1, Bits: 4-7, Default Items Name Root: DDI3420_in_st ● Byte: 2, Bits: 0-3, Default Items Name Root: DDO3410_out_echo ● Byte: 2, Bits: 4-7, Default Items Name Root: DDO3410_out_st ● Byte: 3, Bits: 0-5, Default Items Name Root: DDI3610_in_data ● Byte: 4, Bits: 0-5, Default Items Name Root: DDI3610_in_st ● Byte: 5, Bits: 0-5, Default Items Name Root: DDO3600_out_echo ● Byte: 6, Bits: 0-5, Default Items Name Root: DDO3600_out_st
12	The next task is to create input bytes and words.

Creating Input Items

To create input items for the STB NIC 2212 example, begin with an input data byte containing low byte status for the STB NIC 2212 module:

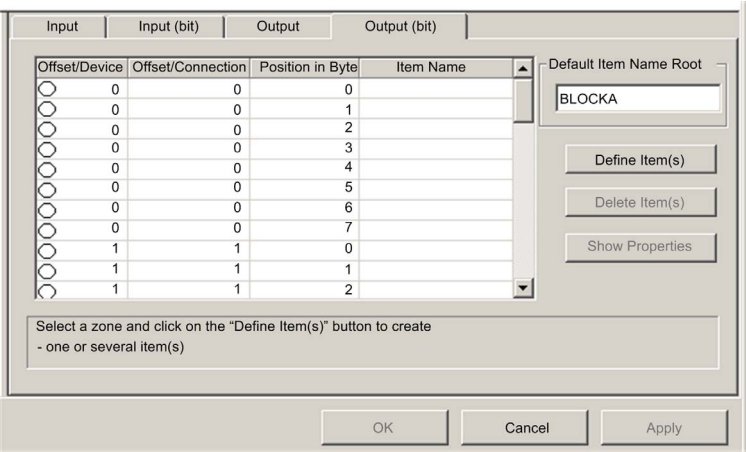
Step	Action
1	<div>Select the Input tab to return to that page:</div> <div></div> <div>NOTE: In this example, both the Offset/Device and Offset/Connection columns represent the byte address. The items you create will be either an 8-bit byte or a 16-bit word</div>
2	In the Default Item Name Root input box type: NIC2212_01_LO_st .
3	<div>Starting at the first available whole input word, select the single row at byte 8:</div> <div></div>

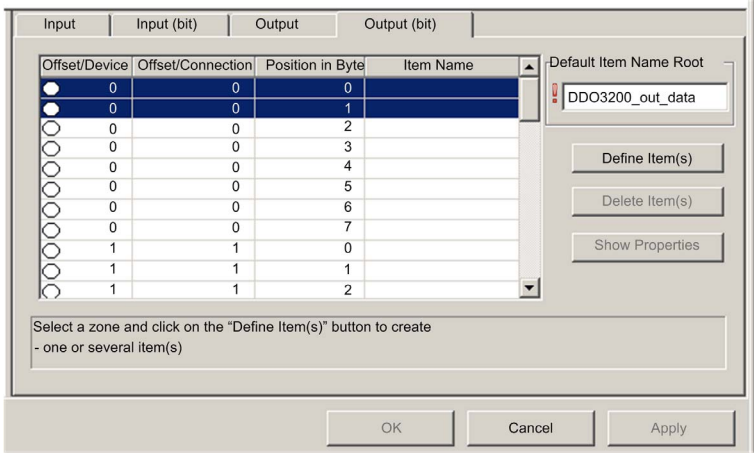
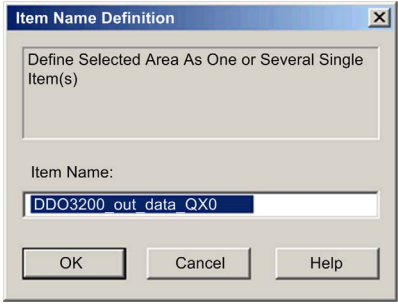
Step	Action
4	<p>Click the Define Item(s) button. Result: The Item Name Definition dialog opens:</p> 
5	<p>Select Byte as the New Item(s) Data Type, then click OK. Result: A new byte item is created:</p> 
6	Click Apply to save the new items and leave the page open.

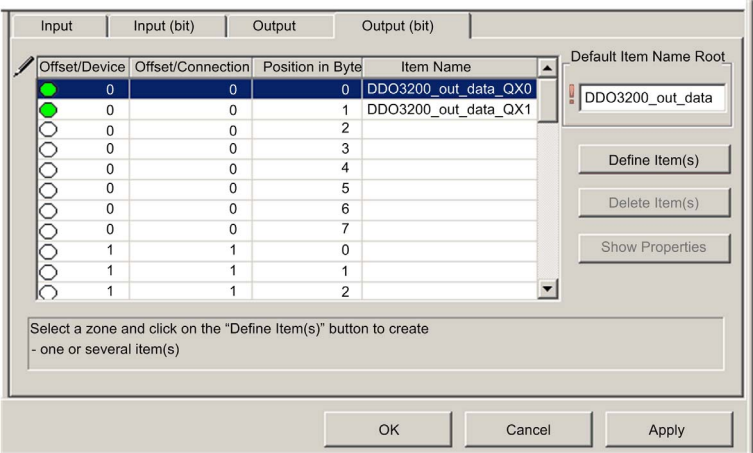
Step	Action
7	<p>Repeat steps 2 - 6 for each byte or word input item you need to create.</p> <p>NOTE: The number of rows you select for a new item depends upon the item type. If the item is a:</p> <ul style="list-style-type: none">● byte: select a single row● word: select two rows, beginning at the next available whole word <p>In this example, you will create items for each of the following:</p> <ul style="list-style-type: none">● Byte: 9, Default Items Name Root: NIC2212_01_HI_st● Word: 10, Default Items Name Root: AVI1270_CH1_in_data● Byte: 12, Default Items Name Root: AVI1270_CH1_in_st● Word: 14-15, Default Items Name Root: AVI1270_CH2_in_data● Byte: 16, Default Items Name Root: AVI1270_CH2_in_st● Byte: 17, Default Items Name Root: AVO1250_CH1_out_st● Byte: 18, Default Items Name Root: AVO1250_CH2_out_st
8	The next task is to create output bits.

Creating Output Bit Items

To create output bit items for the STB NIC 2212 example, beginning with 2 output bits for the STB DDO3200 module:

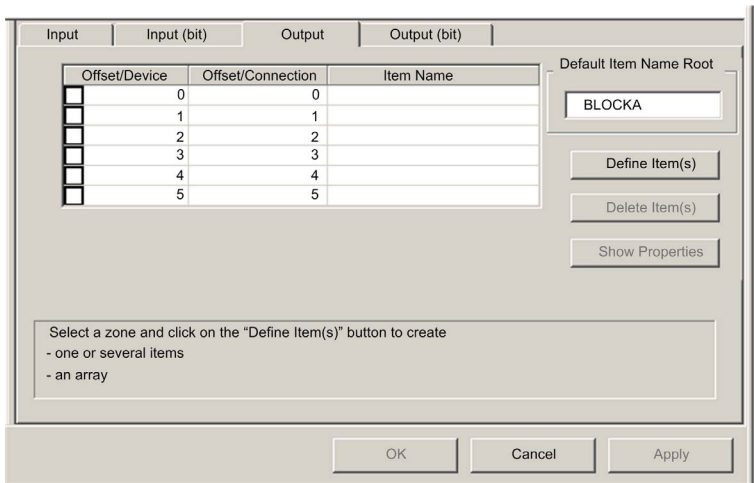
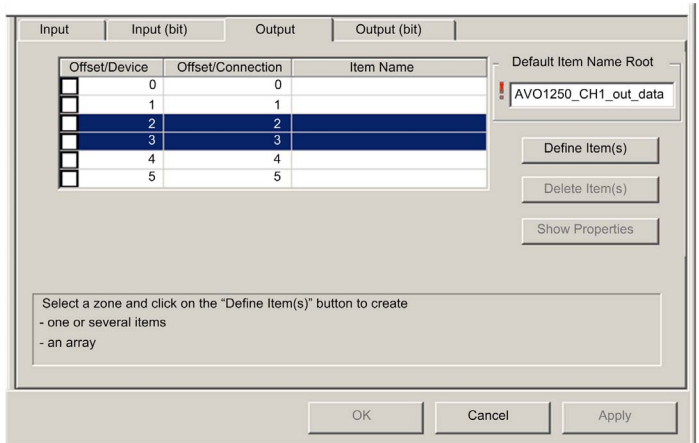
Step	Action
1	<p>Select the Output (bit) tab to open the following page:</p>  <p>NOTE: Both the Offset/Device and Offset/Connection columns represent the byte address of an output, while the Position in Byte column indicates the bit position (within the byte) of each discrete output item.</p>
2	In the Default Items Name Root input box type: DDO3200_out_data .

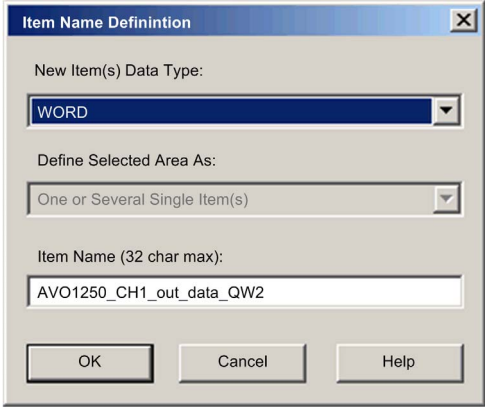
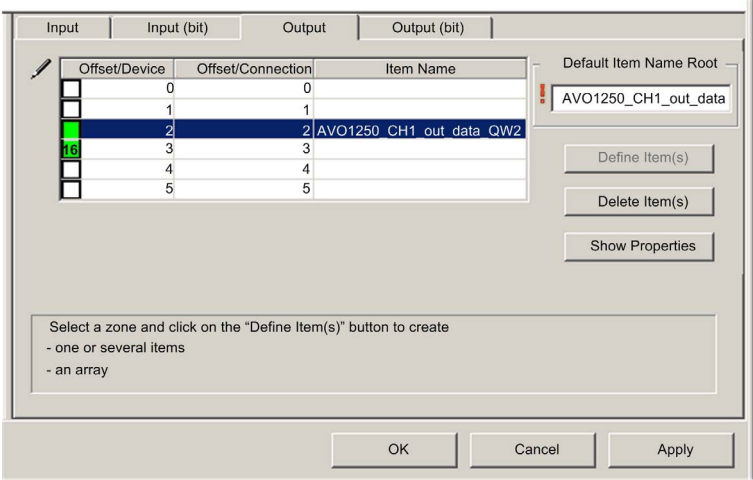
Step	Action
3	<p>In the Items List, select the rows that correspond to bits 0-1 in byte 0—i.e., the first 2 rows:</p>  <p>Select a zone and click on the "Define Item(s)" button to create - one or several item(s)</p>
4	<p>Click the Define Item(s) button.</p> <p>Result: The Item Name Definition dialog opens:</p>  <p>NOTE: The asterisk (*) indicates that a series of discrete items with the same root name will be created.</p>

Step	Action
5	<p>Accept the default output name and click OK. Result: 2 discrete output items are created:</p> 
6	Click Apply to save the new items and leave the page open.
7	<p>Repeat steps 2 - 6 for each group of discrete output items you need to create. In this example, that includes items for each of the following groups:</p> <ul style="list-style-type: none">● Byte: 0, Bits: 2-5, Default Items Name Root: DDO3410_out_data● Byte: 1, Bits: 0-5, Default Items Name Root: DDO3600_out_data
8	The next task is to create output bytes and words.

Creating Numeric Output Items

To create output items for the STB NIC 2212, example, beginning with an output data word for the STB AVO 1250 module:

Step	Action
1	<p>Click on the Output tab to open the following page:</p>  <p>NOTE: In this example, both the Offset/Device and Offset/Connection columns represent the byte address. The items you create will be 16-bit words comprising 2 bytes.</p>
2	In the Default Item Name Root input box type: AVO1250_CH1_out_data .
3	<p>Starting at the next available whole word, select 2 rows: 2 and 3:</p> 

Step	Action
4	<p>Click the Define Item(s) button.</p> <p>Result: The Item Name Definition dialog opens:</p> 
5	<p>Accept the default output name and click OK.</p> <p>Result: The following output word item is created:</p> 
6	Click Apply to save the new item and leave the page open.
7	Repeat steps 2 - 6 for the AVO 1250 channel 2 output data at bytes 4 and 5.
8	Click OK to close the Items window.
9	Select File → Save to save your edits.

EtherNet/IP Implicit Messaging

Overview

The recommended RPI for EtherNet/IP implicit message connections are 1/2 of MAST cycle time. If the resulting RPI is less than 25 ms, the implicit message connections may be adversely affected when the diagnostic features of the CPU's Ethernet I/O scanner service are accessed through explicit messages or the DTM.

In this situation, these timeout multiplier ([see page 229](#)) settings are recommended:

RPI (ms)	Recommended Timeout Multiplier	Connection Timeout (ms)
2	64	128
5	32	160
10	16	160
20	8	160
25	4	100

NOTE: If you use values that are lower than those recommended in the table, the network can consume unnecessary bandwidth, which can affect the performance of the module within the system.

Section 5.12

Configuring the M580 CPU as an EtherNet/IP Adapter

Introduction

This section describes the configuration of an M580 CPU as an EtherNet/IP adapter using *local slave* functionality.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introducing the Local Slave	305
Local Slave Configuration Example	307
Enabling Local Slaves	308
Accessing Local Slaves with a Scanner	309
Local Slave Parameters	312
Working with Device DDTs	314

Introducing the Local Slave

Introduction

The embedded Ethernet I/O scanner service in the M580 CPU scans network modules.

However, you can enable the CPU's scanner service as an EtherNet/IP adapter (or local slave). When the local slave functionality is enabled, network scanners can access CPU data that is mapped to local slave assembly objects in the CPU program.

NOTE:

- The CPU's scanner service continues to function as a scanner when it is enabled as an EtherNet/IP adapter.
- To get data from the primary CPU, make the connection to the Main IP address of the CPU (*see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures*).

The CPU's scanner service supports up to 16 instances of local slaves (Local Slave 1 ... Local Slave 3). Each enabled local slave instance supports these connections:

- one exclusive owner connection
- one listen-only connection

Process Overview

These are the steps in the local slave configuration process:

Stage	Description
1	Enable and configure the CPU's scanner service as a local slave.
2	Configure local slave instances in the scanner service. (Local slave instances correspond to each enabled local slave that is scanned.)
3	Specify the size of local slave input and output assemblies in the scanner service. (Use sizes that match the input and output sizes of the enabled local slave (<i>see page 105</i>).

Implicit and Explicit Messaging

In its role as an EtherNet/IP adapter, the CPU scanner services responds to these requests from network scanners:

- **implicit messages:** Implicit messaging requests are sent from a network scanner device to the CPU. When the local slave functionality is enabled, network scanners can perform these tasks:
 - read messages from the CPU's scanner service
 - write messages to the CPU's scanner service

Implicit messaging is especially suited to the exchange of peer-to-peer data at a repetitive rate.

- **explicit messages:** The CPU's scanner service responds to explicit messaging requests that are directed to CIP objects. When local slaves are enabled by the CPU, explicit messaging requests can access the CPU's scanner service CIP assembly instances. (This is a read-only function.)

Third-Party Devices

If the CPU's scanner service that communicates with the local slave can be configured using Unity Pro, use DTMs that correspond to the CPU to add those modules to your configuration.

Third-party EtherNet/IP scanners that access the local slave assembly instances through the CPU's scanner service do so with respect to the assembly mapping table. The CPU's scanner service is delivered with its corresponding EDS file. Third-party scanners can use the contents of the EDS file to map inputs and outputs to the appropriate assembly instances of the CPU's scanner service.

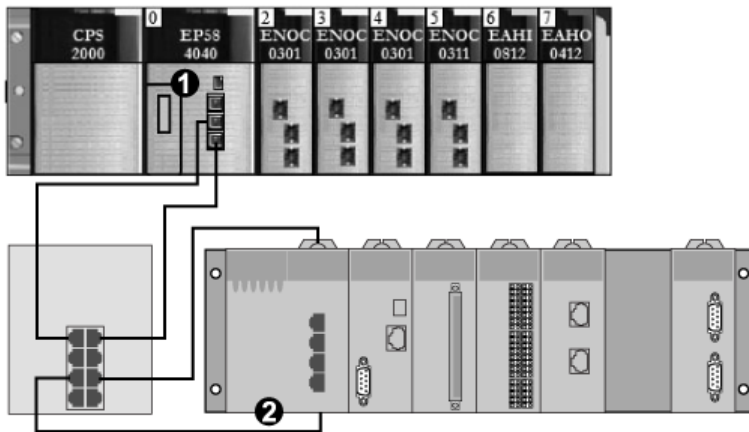
Local Slave Configuration Example

Introduction

Use these instructions to create a simple local slave configuration that includes a network scanner (originator, **O**) and an M580 CPU that is enabled as a local slave (target, **T**).

Originator and Target Devices

This figure, which is a subset of the sample network, shows the enabled local slave (1) and the master device (2):



- 1 M580 CPU: The CPU on the M580 local rack. In this example, you will enable this CPU's embedded scanner service as a local slave device (or target, **T**).
- 2 Modicon M340 rack: In this example, the scanner (or originator, **O**) on this rack scans the CPU data on the M580 rack through the enabled local slave (M580 CPU's scanner service).

Enabling Local Slaves

Introduction

In a sample configuration, you will enable **Local Slave 1** and **Local Slave 2**.

First, use these instructions to enable **Local Slave 1** in the CPU's embedded scanner service configuration. At the end of this exercise, repeat these instructions to enable **Local Slave 2**.

Enabling a Local Slave

Enable the CPU in the M580 local rack as a target device (local slave):

Step	Action
1	Open your M580 Unity Pro project.
2	On the General tab, assign this Alias name to the CPU: BMEP58_ECPU_EXT.
3	In the DTM Browser (Tools → DTM Browser) , double-click the DTM that corresponds to the alias name of the BMENOC0301.2 module to open the configuration window.
4	In the navigation pane, expand (+) EtherNet/IP Local Slaves to see the 3 available local slaves.
5	Select a local slave to see its properties. (For this example, select Local Slave 1 .)
6	In the drop-down list (Properties → Active Configuration), scroll to Enabled .
7	Click Apply to enable Local Slave 1 .
8	Click OK to apply the changes and close the configuration window.

You now have enabled **Local Slave 1** for the CPU's scanner service at IP address 192.168.20.10.

EtherNet/IP scanners that scan the network for the CPU's scanner service at that IP address can use implicit messages to read from and write to the assembly instances that are associated with the local slave instance.

Enabling Another Local Slave

This example uses two local slave connections. Make a second connection for **Local Slave 2**:

Step	Action
1	Repeat the steps above to enable a second local slave (Local Slave 2). NOTE: The appropriate IP address for this example (192.168.20.10) was already assigned to the CPU's scanner service in the assignment of Local Slave 1 .
2	Continue to the next procedure to configure the network scanner (originator, O).

Accessing Local Slaves with a Scanner

Introduction

Use these instructions to map local slave instances in a network scanner to the enabled local slaves in the CPU's embedded scanner service (**Local Slave 1**, **Local Slave 2**, **Local Slave 3**).

This example uses a BMENOC0301 Ethernet communication module as a network scanner (originator, **O**) that scans the CPU scanner service when it is enabled as a local slave (target, **T**).

Configure the BMENOC0301 module in an M580 Unity Pro project.

Adding the Device DTM

Create a local slave instance that corresponds to an enabled local slave by name:

Step	Action
1	Open your M580 Unity Pro project.
2	Right-click the BMENOC0301 module in the DTM Browser (Tools → DTM Browser) and select Add .
3	Select the DTM that corresponds to the CPU. NOTE: <ul style="list-style-type: none"> The DTM used in this example corresponds to the CPU's scanner service. For other target devices, use the DTM from the manufacturer that corresponds to your scanner device. The corresponding input I/O vision and output I/O vision variables are automatically created with the respective suffixes _IN and _OUT.
4	Press the Add DTM button to open the Properties of device dialog window.
5	Assign a context-sensitive Alias name that corresponds to Local Slave 1 for the CPU. Example: BMEP58_ECPU_from_EDS_LS1
6	Click OK to see the local slave instance in the DTM Browser .

Mapping Local Slave Numbers

In the M580 Unity Pro project, associate the local slave instances in the BMENOC0301 scanner with specific local slaves that are enabled for the CPU's scanner service:

Step	Action
1	In the DTM Browser , double-click the local slave instance that corresponds to Local Slave 1 in the CPU target device (BMEP58_ECPU_from_EDS_LS1). NOTE: The default connection is Local Slave 1 - Exclusive Owner , which is most applicable to Local Slave 1 in the target device.
2	Select Local Slave 1 - Exclusive Owner .
3	Click Remove Connection to delete the connection to Local Slave 1 .
4	Click Add Connection to open the dialog box (Select connection to add).

Step	Action
5	Select Local Slave 4 - Exclusive Owner .
6	Click Apply .

The local slave (**Local Slave 1**) is now the target of a local slave instance with a context-sensitive connection name (**Local Slave 1 - Exclusive Owner**).

Mapping IP Addresses

Associate the IP address of the local slave (target, **T**) with the local slave instances in the scanner (originator, **O**) configuration:

Step	Action
1	Double-click the BMENOC0301 module in the DTM Browser .
2	In the navigation pane, expand the Device List (<i>see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide</i>).
3	Select a local slave instance (BMEP58_ECPU_from_EDS_LS1).
4	Select the Address Setting tab.
5	In the IP Address field, enter the IP address of the local slave device (192.168.20.10).
6	Click inside the navigation pane to make the Apply button active. NOTE: You may have to select Disabled in the drop-down menu (DHCP for this device) to activate the OK and Apply buttons.
7	Configure the data size.
8	Click Apply .

Configuring an Additional Connection

You have created one local slave instance that corresponds by name and IP address to an enabled local slave. This example uses two local slave connections, so make another connection for **Local Slave 2**.

Step	Action
1	Repeat the preceding steps (<i>see page 310</i>) to create a second local slave instance that corresponds to Local Slave 2 .
2	Build the Unity Pro project.

Accessing the Device DDT Variables

Step	Action
1	In the Project Browser (Tools → Project Browser), expand Variables & FB instances .
2	Double-click Device DDT Variables to see the device DDTs that correspond with the CPU's scanner service.

Local Slave Parameters

Accessing the Configuration

Open the **EtherNet/IP Local Slaves** configuration page:

Step	Action
1	Open the Unity Pro project.
2	Open the DTM Browser (Tools → DTM Browser).
3	In the DTM Browser , double-click the CPU DTM to open the configuration window. NOTE: You can also right-click the CPU DTM and select Open .
4	Expand (+) Device List in the navigation tree to see the local slave instances.
5	Select the local slave instance to view the Properties and Assembly configuration tabs.

Properties

Identify and enable (or disable) the local slave on the **Properties** tab:

Parameter	Description	
Number	The Unity Pro DTM assigns a unique identifier (number) to the device. These are the default values: <ul style="list-style-type: none">● <i>local slave 1</i>: 129● <i>local slave 2</i>: 130● <i>local slave 3</i>: 131	
Active Configuration	Enabled	Enable the local slave with the configuration information in the Assembly fields when the CPU scanner service is an adapter for the local slave node.
	Disabled	Disable and deactivate the local slave. Retain the current local slave settings.
Comment	Enter an optional comment (maximum: 80 characters).	
Connection Bit	The connection bit is represented by an integer (769 ... 896). NOTE: <ul style="list-style-type: none">● This setting is auto-generated after the local slave settings are input and the network configuration is saved.● The connection bit is represented by an integer:<ul style="list-style-type: none">○ 385...387 (firmware v1.0)○ 769...896 (firmware v.2.10)	

Assembly

Use the **Assembly** area of the **Local Slave** page to configure the size of the local slave inputs and outputs. Each device is associated with these assembly instances:

- Outputs
- Inputs
- Configuration
- Heartbeat (The heartbeat assembly instance is for listen-only connections only.)

The Unity Pro assembly numbers are fixed according to this table, where **O** indicates the originator (scanner) device and **T** indicates the target device:

Local Slave	Number		Connection
	Device	Assembly	
1	129	101	Outputs (T->O)
		102	Inputs (O->T)
		103	Configuration
		199	Heartbeat
2	130	111	Outputs (T->O)
		112	Inputs (O->T)
		113	Configuration
		200	Heartbeat
3	131	121	Outputs (T->O)
		122	Inputs (O->T)
		123	Configuration
		201	Heartbeat

NOTE: When using explicit messaging to read the CPU's scanner service assembly instance, allocate sufficient room for the response. The size of the response equals the sum of: assembly size + 1 byte (Reply service) + 1 byte (General Status).

Limitations (from the perspective of the local slave):

- *maximum RPI value:* 65535 ms
- *maximum timeout value:* 512 * RPI
- *outputs (T->O):* 509 bytes maximum
- *inputs (O->T):* 505 bytes maximum
- *configuration for the CPU scanner service:* 0 (fixed)

Working with Device DDTs

Introduction

Use Unity Pro to create a collection of device derived data types (DDDTs) and variables that support communications and the transfer of data between the PAC and the various local slaves, distributed devices, and corresponding I/O modules.

You can create DDDTs and corresponding variables in the Unity Pro DTM. Those program objects support your network design.

NOTE: The default device name depends on the firmware version installed in the selected CPU, and may be one of the following:

- T_BMEP58_ECPU
- T_BMEP58_ECPU_EXT
- T_M_ECPU_HSBY

Use the DDDTs for these tasks:

- Read status information from the Ethernet communication module.
- Write control instructions to the Ethernet communication module.

You can double-click the name of the DDDT in the **Project Browser** at any time to view its properties and open the corresponding EDS file.

NOTE: For applications that require multiple DDDTs, create an **Alias name** that logically identifies the DDDT with the configuration (module, slot, local slave number, etc.).

DDDT Variables

You can access the DDDTs and the corresponding variables in Unity Pro and add them to a user-defined **Animation Table**. Use that table to monitor read-only variables and edit read-write variables.

Use these data types and variables to perform these tasks:

- Read the status of connections and communications between the Ethernet communication module and distributed EtherNet/IP and Modbus TCP devices:
 - The status is displayed in the form of a HEALTH_BITS array consisting of 32 bytes.
 - A bit value of 0 indicates the connection is lost or the communication module can no longer communicate with the distributed device.
- Toggle a connection ON (1) or OFF (0) by writing to a selected bit in a 16-word DIO_CTRL array
- Monitor the value of local slave and distributed device input and output items that you created in Unity Pro.

NOTE: The HEALTH_BITS array is not copied to the standby CPU in a Hot Standby switchover. The DIO_CTRL array is copied to the standby CPU in a Hot Standby switchover.

Displaying the Order of Input and Output Items

View the DDDTs in Unity Pro (**Project Browser** → **Variables & FB instances** → **Device DDT Variables**). The **Data Editor** is now open. Click the **DDT Types** tab.

The **Data Editor** displays each input and output variable. When you open the first input and output variables, you can see both the connection health bits ([see page 218](#)) and the connection control bits ([see page 217](#)).

This table shows the rule assignment for connection numbers:

Input Variables	Order	Output Variables
Modbus TCP input variables (note 1)	1	Modbus TCP output variables (note 1)
ERIO drop input variables	2	
local slave input variables (note 2)	3	local slave output variables (note 3)
EtherNet/IP input variables(note 1)	4	EtherNet/IP output variables (note 1)
<p>NOTE 1: DDDTs are in this format:</p> <ul style="list-style-type: none"> i. by device number ii. within a device (by connection number) iii. within a connection (by item offset) <p>NOTE 2: Local slave variables are in this format:</p> <ul style="list-style-type: none"> i. by local slave number ii. within each local slave (by item offset) 		

Section 5.13

Hardware Catalog

Introduction

The Unity Pro **Hardware Catalog** displays the modules and devices that you can add to a Unity Pro project. Each module or device in the catalog is represented by a DTM that defines its parameters.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction to the Hardware Catalog	317
Adding a DTM to the Unity Pro Hardware Catalog	318
Adding an EDS File to the Hardware Catalog	319
Removing an EDS File from the Hardware Catalog	322

Introduction to the Hardware Catalog

Introduction

The Unity Pro **Hardware Catalog** contains a list of modules and devices that you can add to a Unity Pro project. EtherNet/IP and Modbus TCP devices are located in the **DTM Catalog** tab at the bottom of the **Hardware Catalog**. Each module or device in the catalog is represented by a DTM that defines its parameters.

EDS Files

Not all devices in today's market offer device-specific DTMs. Some devices are defined by device-specific EDS files. Unity Pro displays EDS files in the form of a DTM. In this way, you can use Unity Pro to configure devices that are defined by an EDS file in the same way you would configure a device defined by its DTM.

Other devices lack both a DTM and an EDS file. Configure those devices by using the generic DTM on the **DTM Catalog** page.

View the Hardware Catalog

Open the Unity Pro **Hardware Catalog**:

Step	Action
1	Open Unity Pro.
2	Find the PLC bus in the Project Browser .
3	Use one method to open the catalog: <ul style="list-style-type: none">● Use the pull-down menu (Tools → Hardware Catalog).● Double-click an empty slot in the PLC bus.

Adding a DTM to the Unity Pro Hardware Catalog

A Manufacturer-Defined Process

Before a DTM can be used by the Unity Pro **Hardware Catalog**, install the DTM on the host PC (the PC that is running Unity Pro).

The installation process for the DTM is defined by the device manufacturer. Consult the documentation from the device manufacturer to install a device DTM on your PC.

NOTE: After a device DTM is successfully installed on your PC, update the Unity Pro Hardware Catalog to see the new DTM in the catalog. The DTM can then be added to a Unity Pro project.

Adding an EDS File to the Hardware Catalog

Introduction

You may want to use an EtherNet/IP device for which no DTM is in the catalog. In that case, use these instructions to import the EDS files into the catalog to create a corresponding DTM.

Unity Pro includes a wizard you can use to add one or more EDS files to the Unity Pro **Hardware Catalog**. The wizard presents instruction screens to execute these commands:

- Simplify the addition of EDS files to the **Hardware Catalog**.
- Provide a redundancy check when you add duplicate EDS files to the **Hardware Catalog**.

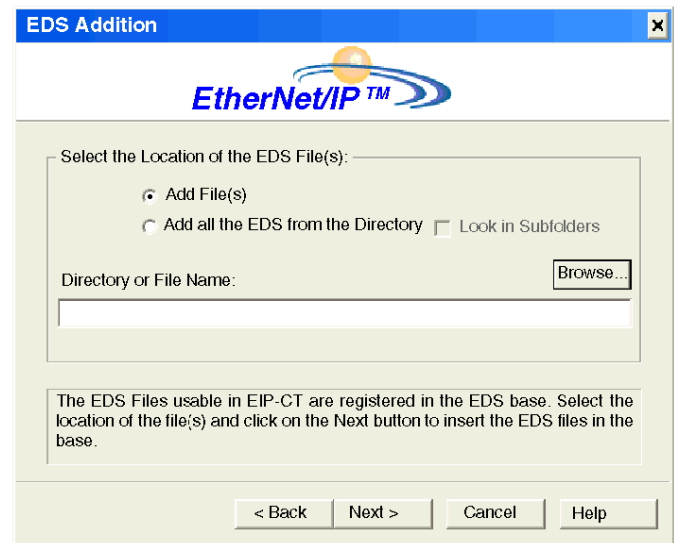
NOTE: The Unity Pro **Hardware Catalog** displays a partial collection of DTMs and EDS files that are registered with the ODVA. This library includes DTMs and EDS files for products that are not manufactured or sold by Schneider Electric. The non-Schneider Electric EDS files are identified by vendor in the catalog. Please contact the identified device's manufacturer for inquiries regarding the corresponding non-Schneider Electric EDS files.

Adding EDS Files

Open the **EDS Addition** dialog box:




Step	Action
1	Open a Unity Pro project that includes an Ethernet communication module.
2	Open the DTM Browser (Tools → DTM Browser).
3	In the DTM Browser , select a communication module.
4	Right-click on the communication module and scroll to Device menu → Additional functions → Add EDS to library .
5	In the EDS Addition window, click Next .

You can now see this page:



Add one or more EDS files to the library:

Step	Action
1	Use these commands in the Select the Location of the EDS File(s) area of the EDS Addition dialog box to identify the location of the EDS files: <ul style="list-style-type: none">● Add File(s): Add one or more EDS files that are individually selected.● Add all the EDS from the Directory: Add all files from a selected folder. (Check Look in Subfolders to add EDS files from the folders within the selected folder.)
2	Click Browse to open a navigation dialog box.
3	Select the location of the EDS file(s): <ul style="list-style-type: none">● Navigate to at least one EDS file.● Navigate to a folder that contains EDS files. NOTE: Keep the location selected (highlighted).
4	Click Select to close the navigation window. NOTE: Your selection appears in the Directory or File Name field.
5	Click Next to compare the selected EDS files to the files in the library. NOTE: If one or more selected EDS files is a duplicate, a File Already Exists message appears. Click Close to hide the message.

Step	Action
6	<p>The next page of the EDS Addition wizard opens. It indicates the status of each device you attempted to add:</p> <ul style="list-style-type: none">● check mark  (green): The EDS file can be added.● informational icon  (blue): There is a redundant file.● exclamation point  (red): There is an invalid EDS file. <p>NOTE: You can click View Selected File to open and view the selected file.</p>
7	<p>Click Next to add the non-duplicate files.</p> <p>Result: The next page of the EDS Addition wizard opens to indicate that the action is complete.</p>
8	<p>Click Finish to close the wizard.</p> <p>Result: The hardware catalog automatically updates.</p>

Removing an EDS File from the Hardware Catalog

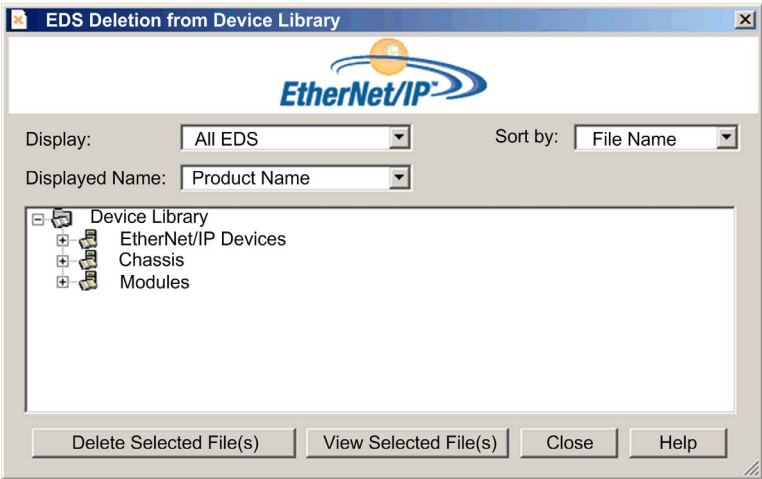
Introduction

You can remove a module or device from the list of available devices in the Unity Pro **Hardware Catalog** by removing its **EDS** file from the library.

When you remove an EDS file from the library, the device or module disappears from the **DTM Catalog**. However, removing the file from the library does not delete the file from its stored location, so you can import the file again later.

Removing an EDS File from the Catalog

Use these steps to remove an EDS file from the catalog:

Step	Action
1	Open the Unity Pro DTM Browser (Tools → DTM Browser).
2	In the DTM Browser , select an Ethernet communication module.
3	Right-click the module and scroll to Device menu → Additional functions → Remove EDS from library to open the EDS Deletion from Device Library window: 

Step	Action	
4	Use the selection lists in the heading of this window to specify how EDS files are displayed:	
	Display	Choose criteria to filter the list of EDS files: <ul style="list-style-type: none"> ● All EDS (no filtering) ● Only Devices ● Only Chassis ● Only Modules
	Sort by	Choose criteria to sort the list of displayed EDS files: <ul style="list-style-type: none"> ● File Name ● Manufacturer ● Category ● Device Name
	Displayed Name	Choose the identifier for each device: <ul style="list-style-type: none"> ● Catalog Name ● Product Name
5	Expand (+) the Device Library navigation tree and select the EDS file you want to remove. NOTE: Click View Selected File to see the read-only contents of the selected EDS file.	
6	Click the Delete Selected File(s) button to open the DeleteEDS dialog box.	
7	Click Yes to remove the selected EDS file from the list.	
8	Repeat these steps for each EDS file you want to delete.	
9	Click Finish to close the wizard. Result: The hardware catalog automatically updates.	

Section 5.14

M580 CPU Embedded Web Pages

Introduction

The M580 CPU includes a Hypertext Transfer Protocol (HTTP) server. The server transmits web pages for the purpose of monitoring, diagnosing, and controlling remote access to the communication module. The server provides easy access to the CPU from standard internet browsers.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introducing the Standalone Embedded Web Pages	325
Status Summary (Standalone CPUs)	326
Performance	328
Port Statistics	329
I/O Scanner	331
Messaging	333
QoS	334
NTP	336
Redundancy	338
Alarm Viewer	339
Rack Viewer	340

Introducing the Standalone Embedded Web Pages

Introduction

Use the embedded web server pages to perform these tasks:

- Display real-time diagnostic data for both the M580 CPU and other networked devices.
- Read the values from and write values to Unity Pro application variables.
- Manage and control access to the embedded web pages by assigning separate passwords for these functions:
 - View the diagnostic web pages.
 - Use the Data Editor to write values to Unity Pro application variables.

Browser Requirements

The embedded web server in the M580 CPU displays data in standard HTML web pages. Access the embedded web pages on a PC, iPad, or Android tablet with these browsers:

- Internet Explorer (v8 or later) (v10 or later for Windows Phone OS)
- Google Chrome (v11 or later) (v35 or later for Android OS v4 mini)
- Mozilla Firefox (v4 or later)
- Safari (v6.0 for Apple Mac. No support for Windows.)

Access the Web Pages

Open the **Home** page:

Step	Action
1	Open an Internet browser.
2	In the address bar, enter the IP address of the M580 CPU (<i>see page 119</i>).
3	Press Enter and wait for the Home page to open.

Access these pages by expanding the **Menu** on the **Home** page:

- **Status Summary** (*see page 326*)
- **Performance** (*see page 328*)
- **Port Statistics** (*see page 329*)
- **I/O Scanner** (*see page 331*)
- **Messaging** (*see page 333*)
- **QoS** (*see page 334*)
- **Network Time Service** (*see page 336*)
- **Redundancy** (*see page 338*)
- **Alarm Viewer** (*see page 339*)
- **Rack Viewer** (*see page 340*)

Status Summary (Standalone CPUs)

Open the Page

Access the **Status Summary** page from the **Diagnostics** tab (**Menu** → **Module** → **Summary**):

Status Summary

RUN

MOD STATUS

ERR

NETWORK STATUS

Service Status

✓

DHCP Server

Enabled

✓

FDR Server

Enabled

⊗

Access Control

Disabled

✗

Scanner Status

One Connection Is Bad

✓

NTP Status

Enabled

CPU Summary

Model

BME P58 3040

State

RUN

Scan Time

2 ms

Logged In

No

CPU Exec. Version

1.13

Unity Program

Project

Version Info.

Exec. Version

2.01

Web Server Version

1.0

Web Site Version

V2.01 IR02

CIP Version

1.0

Network Info.

IP Address

192.168.20.40

Sunbnet Address

255.255.0.0

Gateway Address

192.168.0.120

MAC Address

0 00 54 00 10 20

Host Name

BMENOC0311

NOTE:

- This page is updated every 5 seconds.
- Refer to the **Status Summary** page for Hot Standby CPUs (*see page 346*).

326

EIO0000001578 09/2017

Diagnostic Information

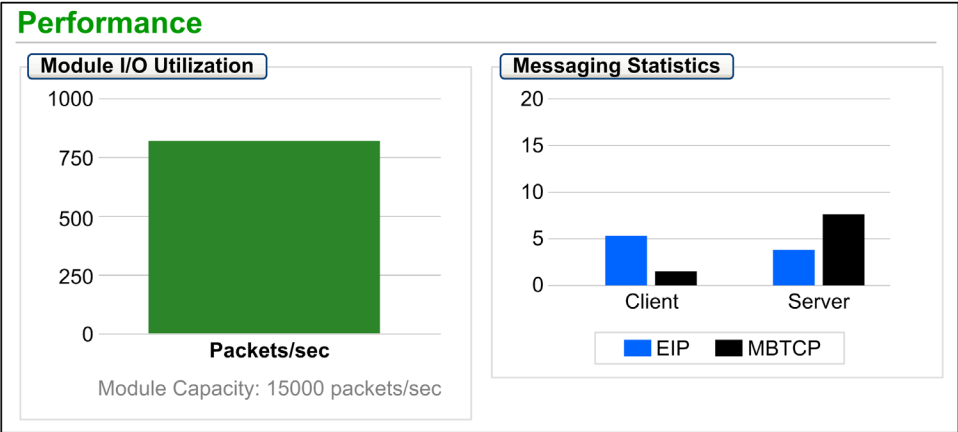
The objects on this page provide status information:

Parameters	Description	
LEDs	The black field contains LED indicators (RUN , ERR , etc.). NOTE: The diagnostics information is explained in the description of LED activity and indications (see page 47).	
Service Status	green	The available service is operational and running.
	red	An error is detected in an available service.
	black	The available service is not present or not configured.
Version Info.	This field describes the software versions that are running on the CPU.	
CPU Summary	This field describes the CPU hardware and the applications that are running on the CPU.	
Network Info.	This field contains network and hardware address information and connectivity that corresponds to the CPU.	

Performance

Open the Page

Access the **Performance** page from the **Diagnostics** tab (**Menu → Module → Performance**):



NOTE:

- Move the mouse over the dynamic graphs to see the current numeric values.
- This page is updated every 5 seconds.

Diagnostic Information

This table describes the performance statistics:

Field	Description
Module I/O Utilization	This graph shows the total number of packets (per second) the CPU can handle at once.
Messaging Statistics	This graph shows the number of Modbus/TCP or EtherNet/IP messages per second for the client or server.

Port Statistics

Open the Page

Access the **Port Statistics** page from the **Diagnostics** tab (**Menu → Module → Port Statistics**):

Port Statistics					
	Internal Port	ETH1	ETH2	ETH3	Eth Backplane Port
Speed	1000 Mbps	100 Mbps	100 Mbps	100 Mbps	100 Mbps
Duplex	TP-Full	TP-Full Link	TP-Full Link	TP-Full	TP-Full Link
Redundancy Status	Disabled	Disabled	Forwarding	Forwarding	Disabled
Success Rate	100.00%	100.00%	100.00%	100.00%	100.00%
Total Errors	0	0	0	0	0
<div> Reset Counters Detail View </div>					

NOTE: This page is updated every 5 seconds. Click **Reset Counters** to reset all dynamic counters to 0.

Diagnostic Information

This page shows the statistics for each port on the CPU. This information is associated with the configuration of the Ethernet ports ([see page 55](#)) and the configuration of the service/extended port ([see page 130](#)).

The frame color indicates the port activity:

- *green*: active
- *gray*: inactive
- *yellow*: error detection
- *red*: error detection

Expanded View

Click **Detail View** to see more statistics:

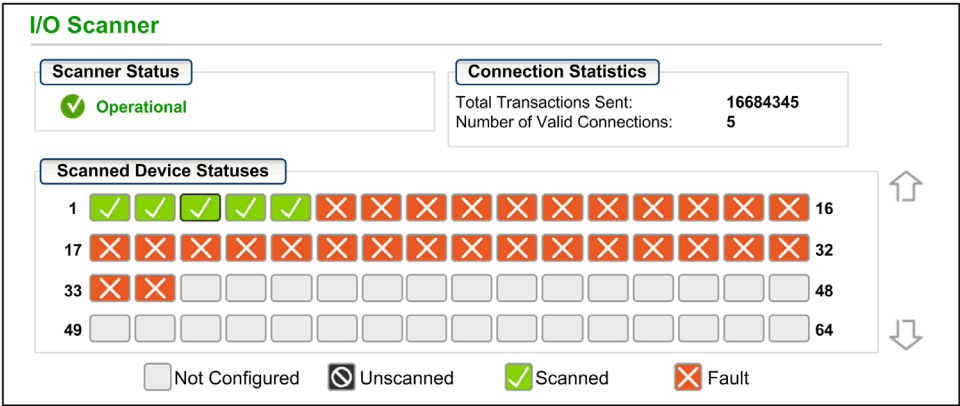
Statistic	Description
Frames Transmitted	number of frames successfully transmitted
Frames Received	number of frames received
Excessive Collisions	number of excessive Ethernet collisions
Late Collisions	number of late Ethernet collisions
CRC Errors	number of detected cyclic redundancy check errors
Bytes Received	number of bytes received
Inbound Packet Errors	number of detected inbound packet errors

Statistic	Description
Inbound Packets Discarded	number of inbound packets discarded
Bytes Transmitted	number of bytes transmitted
Outbound Packet Errors	number of detected outbound packet errors
Outbound Packets Discarded	number of outbound packets discarded

I/O Scanner

Open the Page

Access the I/O Scanner page from the **Diagnostics** tab (**Menu** → **Connected Devices** → **Scanner Status**):



NOTE: This page is updated every 5 seconds.

Diagnostic Information

This table describes the scanner status and connection statistics:

Scanner Status	Operational	The I/O scanner is enabled.
	Stopped	The I/O scanner is disabled.
	Idle	The I/O scanner is enabled but not running.
	Unknown	The I/O scanner returns unexpected values from the device.
Connection Statistics	Transactions per Second	
	Number of Connections	

In the **Scanned Device Status** display, the colors that appear in each block indicate these states for specific remote devices:

Color	Indication	Status
gray	Not Configured	There is an unconfigured device.
black	Unscanned	The scanning of the specific device has been intentionally disabled.
green	Scanned	A device is being scanned successfully.
red	Fault	A device that is being scanned is returning detected errors.

Hold the cursor over any block to get information for a specific device:

1

✓

✓

✓

✓

✓

✗

✗

✗

✗

✗

✗

✗

✗

✗

✗

16

17

✗

✗

Health: OK

IP: 192.168.110.1

Type: Modbus TCP

Device Number: 3

32

33

✗

✗

48

49

64

Not Configured

⊘

Unscanned

✓

Scanned

✗

Fault

Messaging

Open the Page

Access the **Messaging** page from the **Diagnostics** tab (Menu → Connected Devices → **Messaging**):

Messaging

Messaging Statistics

Messages Sent: 6513

Messages Received: 6516

Success Rate: 100.00%

Active Connections

Remote Address	Remote Port	Local Port	Type	Msgs. Sent	Msgs. Received	Errors
127.0.0.1	65359	502	0	2173	2172	0

NOTE: This page is updated every 5 seconds.

Diagnostic Information

This page shows current information for open Modbus TCP connections on port 502:

Field	Description
Messaging Statistics	This field contains the total number of sent and received messages on port 502. These values are not reset when the port 502 connection is closed. Therefore, the values indicate the number of messages that have been sent or received since the module was started.
Active Connections	This field shows the connections that are active when the Messaging page is refreshed.

QoS

Open the Page

Access the **QoS** (quality of service) page from the **Diagnostics** tab (**Menu** → **Services** → **QoS**):

QoS

Service Status

Running

Precision Time Protocol

DSCP PTP Event Priority59

DSCP PTP General47

EtherNet/IP Traffic

DSCP Value for I/O Data Schedule Priority Messages47

DSCP Value for Explicit Messages27

Detail View

Modbus/TCP Traffic

DSCP Value for I/O Messages43

DSCP Value for Explicit Messages27

Network Time Protocol Traffic

DSCP Value for Network Time59

- NOTE:**
- Configure the QoS in Unity Pro (*see page 129*).
 - Click **Detail View** to expand the list of parameters.
 - This page is updated every 5 seconds.

Service Status

This table shows the possible states for the **Service Status**:

Status	Description
Running	The service is correctly configured and running.
Disabled	The service is disabled.
Unknown	The status of the service is not known.

Diagnostic Information

This page displays information about the QoS service that you configure in Unity Pro (*see page 129*).

When you enable QoS, the module adds a differentiated services code point (DSCP) tag to each Ethernet packet it transmits, thereby indicating the priority of that packet:

Field	Parameter	Description
Precision Time Protocol	DSCP PTP Event Priority	Point-to-point time synchronization.
	DSCP PTP General	Point-to-point general.
EtherNet/IP Traffic	DSCP Value for I/O Data Scheduled Priority Messages	Configure the priority levels to prioritize the management of data packets.
	DSCP Value for Explicit Messages	
Modbus/TCP Traffic	DSCP Value for I/O Messages	NOTE: We recommend that you use a larger timeout value for explicit messaging connections and a smaller timeout value for implicit messaging connections. The specific values that you employ depend on your application requirements.
	DSCP Value for Explicit Messages	
Network Time Protocol Traffic	DSCP Value for Network Time	—

Considerations

Take measures to effectively implement QoS settings in your Ethernet network:

- Use only network switches that support QoS.
- Apply the same DSCP values to all network devices and switches.
- Use switches that apply a consistent set of rules for handling the different DSCP values when transmitting and receiving Ethernet packets.

NTP




Introduction


The **NTP** page displays information about the network time service. Configure this service in Unity Pro (*see page 125*).

Open the Page

Access the **NTP** page from the **Diagnostics** tab (**Menu → Services → NTP**):

NTP

Service Status  Running	Server Status  192.168.0.121	Server Type Secondary
DST Status  On	Current Date Wed Jan 02 2015	Current Time 02:00:18
Time Zone UTC +01:00		
NTP Service Statistics Number of Requests: 6546 Success Rate: 100%		
Number of Responses: 6546 Last Error: 0		
Number of Errors: 0		

Reset Counters

NOTE:

- Click **Reset Counters** to reset all dynamic counters to 0.
- This page is updated every 5 seconds.

Diagnostic Information

The Network Time Service synchronizes computer clocks over the Internet for the purposes of event recording (sequence events), event synchronization (trigger simultaneous events), or alarm and I/O synchronization (time stamp alarms):

Field	Description	
Service Status	Running	The NTP service is correctly configured and running.
	Disabled	The NTP service is disabled.
	Unknown	The NTP service status is unknown.
Server Status	green	The server is connected and running.
	red	A bad server connection is detected.
	gray	The server status is unknown.
Server Type	Primary	A primary server polls a master time server for the current time.
	Secondary	A secondary server requests the current time only from a primary server.
DST Status	Running	DST (daylight saving time) is configured and running.
	Disabled	DST is disabled.
	Unknown	The DST status is unknown.
Current Date	This is the current date in the selected time zone.	
Current Time	This is the current time in the selected time zone.	
Time Zone	This field shows the time zone in terms of plus or minus Universal Time, Coordinated (UTC).	
NTP Service Statistics	These fields show the current values for service statistics.	
	Number of Requests	This field shows the total number of requests sent to the NTP server.
	Success Rate	This field shows the percentage of successful requests out of the total number of requests.
	Number of Responses	This field shows the total number of responses received from the NTP server.
	Last Error	This field contains the error code of the last error that was detected during the transmission of an email message to the network.
	Number of Errors	This field contains the total number of email messages that could not be sent to the network or that have been sent but not acknowledged by the server.


Redundancy

Open the Page

Access the **Redundancy** page on the **Diagnostic** tab (**Menu** → **Services** → **Redundancy**):

Redundancy

Service Status

 **Running**

Last Topology Change

6/17/2015 4:26:35 PM

Router Bridge Statistics

Bridge ID: 00 00 00 80 F4 01 F5 BB
Bridge Priority: 0

Internal Interface

ETH1

ETH2

ETH3

Eth Backplane...

RSTP Disabled
Non-STP Port
Priority: 0

RSTP Disabled
Non-STP Port
Priority: 0

RSTP Forwarding
Designated Port
Priority: 0

RSTP Forwarding
Designated Port
Priority: 0

RSTP Disabled
Non-STP Port
Priority: 0

NOTE: This page is updated every 5 seconds.

Diagnostic Information

This page displays values from the RSTP configuration in Unity Pro (*see page 121*):

Field	Description	
Service Status	Running	The RSTP bridge on the corresponding CPU is properly configured and running.
	Disabled	The RSTP bridge on the corresponding CPU is disabled.
	Unknown	The status of the RSTP bridge on the corresponding CPU is not known.
Last Topology Change	These values represent the date and time that the last topology change was received for the corresponding Bridge ID .	
Redundancy Status	green	The designated Ethernet port is learning or formatting information.
	yellow	The designated Ethernet port is discarding information.
	gray	RSTP is disabled for the designated Ethernet port.
Router Bridge Statistics	Bridge ID	This unique bridge identifier is the concatenation of the bridge RSTP priority and the MAC address.
	Bridge Priority	In Unity Pro, configure the RSTP operating state (<i>see page 121</i>) of the Bridge ID .

Alarm Viewer

Open the Page

Access the **Alarm Viewer** page from the **Diagnostics** tab (**Menu → System → Alarm Viewer**):

Type	Status	Message	Occurance	Acknowledged	Zone
	OK		Invalid Date		0
		Generic system error	5/28/2015 10:47:34 AM	No	0
		Arithmetic error	5/28/2015 10:52:07 AM	No	0

NOTE: This page is updated every 5 seconds.

Diagnostic Information

The **Alarm Viewer** page reports detected application errors. You can read, filter, and sort information about alarm objects on this page. Adjust the type of information displayed by the **Alarm Viewer** in the **Filter Alarms** box.

Each alarm has a timestamp, a description, and an acknowledgement status:

- critical (red)
- acknowledged (green)
- information (blue) (These alarms do not require acknowledgement.)

This table describes the components of the page:

Column	Description	
Type	This column describes the alarm type.	
Status	STOP	You need to acknowledge the alarm.
	ACK	An alarm has been acknowledged.
	OK	An alarm does not require acknowledgment.
Message	This column contains the text of the alarm message.	
Occurance	This column contains the date and time that the alarm occurred.	
Acknowledged	This column reports the acknowledged status of the alarm.	
Zone	This column contains the area or geographical zone from which the alarm comes (0: common area).	

Rack Viewer

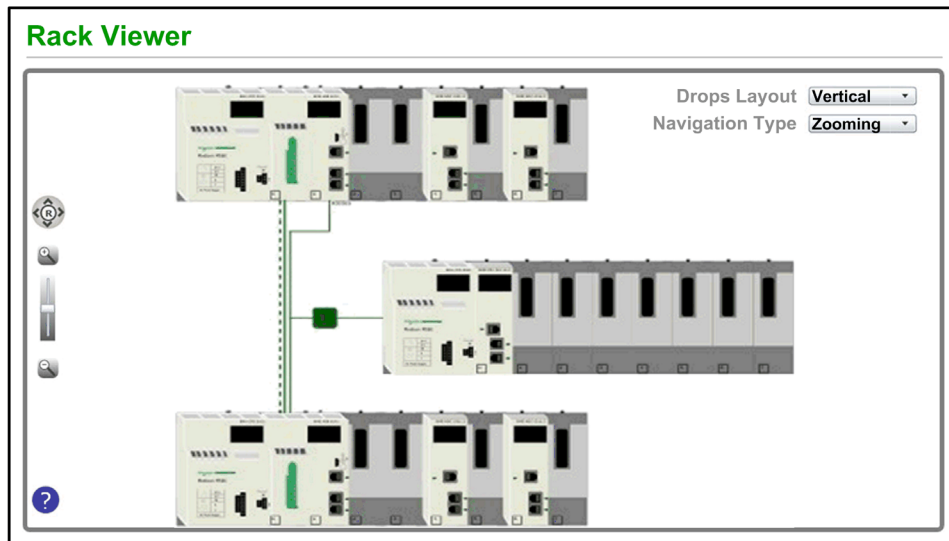
Open the Page

The BMEP584040, BMEP585040, and BMEP586040 standalone CPUs include a **Rack Viewer** web page. Access this page from the **Diagnostics** tab (**Menu → System → Rack Viewer**).

NOTE: You may have to wait a few seconds for the **Rack Viewer** to replicate your configuration.

Example

This example of a **Rack Viewer** page shows the Hot Standby connection between a primary CPU rack and a standby CPU rack. Both racks contain a power supply, a CPU, and a BMECRA312•0 communications module (in slot 7):



The Hot Standby connection (dashed line) is green when the Hot Standby link is healthy.

Information from This Page

The rack that appears in the top left of the **Rack Viewer** represents the local rack that contains the CPU.

Select navigation and view options in the **Rack Viewer** page:

Control	Selection	Description
Drops Layout (menu)	Horizontal	Each RIO drop is shown in a top-to-bottom order beneath the primary bus. The lowest number RIO drop is at the top.
	Vertical	Each RIO drop is shown in a left-to-right order beneath the primary bus. The lowest number RIO drop is at the left.
Navigation Type (menu)	Zooming	Zoom in (+) an out (-) with the zoom control (magnifying glass).
	ScrollBar	View different parts of the page by sliding the scroll bars.
R (button)	R	Click the R (reset) button to reset the page.
	Navigate Up	Press the up arrow to scroll up.
	Navigate Down	Press the down arrow to scroll down.
	Navigate Right	Press the right arrow to scroll to the right.
	Navigate Left	Press the right arrow to scroll to the left.

NOTE: Click the help button (blue question mark) at any time to get information about navigating on the **Rack Viewer** page.

Click on any CPU in the **Rack Viewer** to see this information:

BME H58 : Bus o Drop 0 Rack 0 Slot 0

RUN

ERR

I/O

Processor

RAM Size(kb):

131072 KB

CID:

190984392

Processor Version:

2.01 - 8

MID:

513287308

Hardware Id:

2330B0E

AID:

0

State:

Run

LID:

513287308

Error:

0X0C82

DID:

513287308

Calendar:

June 02 2015 15:56:26

Name:

"Project"

RAM Size(kb):

FALSE

Version:

3

RAM Size(kb):

FALSE

Creation Product:

Unity Pro XLV11.01.01.150422-May 29, Friday...

Creation Product:

TRUE

You can read this CPU data:

- CPU reference name
- rack and slot location
- CPU state (**RUN**, **ERR**, and **I/O**)
- processor and network card information
- application name (on the CPU)

Click the **X** to close this window.

Section 5.15

M580 Hot Standby CPU Web Pages

Overview

This section describes the diagnostic web pages for the M580 BMEH58•040 Hot Standby CPU modules.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introducing the M580 Hot Standby CPU Web Pages	344
Status Summary (Hot Standby CPUs)	346
HSBY Status	348
Rack Viewer	351

Introducing the M580 Hot Standby CPU Web Pages

Introduction

The M580 BMEH58•040 Hot Standby CPUs includes an embedded web server that provide monitoring and diagnostic functions. All web pages are read-only.

These web pages are included:

- Module:
 - Status Summary (Hot Standby) (*see page 346*)
 - HSBY Status (*see page 348*)
 - Performance (*see page 328*)
 - Port Statistics (*see page 329*)
- Connected Devices:
 - I/O Scanner (*see page 331*)
 - Messaging (*see page 333*)
- Services:
 - QoS (*see page 334*)
 - NTP (*see page 336*)
 - Redundancy (*see page 338*)
- System:
 - Alarm Viewer (*see page 339*)

In addition, for the BMEH584040 and BMEH586040 Hot Standby CPUs, a Rack Viewer (*see page 351*) page is also included.

This section describes the web pages that are unique to the M580 BMEH58•040 Hot Standby CPUs: the *Status Summary* and *HSBY Status* web pages. For all other web pages, refer to the M580 CPU Embedded Web Pages (*see page 324*) topics of the *Modicon M580 Hardware Reference Manual*.

Browser Access Requirements

The embedded web pages are accessible using the following operating system and browser combinations:

Operating system	Browser
Android OS v4 mini	Chrome mobile minimum version 35.0.1916.141
iOS6	Safari v6
iOS7	
Windows 7	Internet Explorer v8.0.7601.17514
Windows 8	
Windows 8.1	
Windows 8.1 RT	Internet Explorer minimum v8
Windows Phone OS	Internet Explorer Mobile v10

The embedded web site is accessible via WiFi, using a smartphone or tablet equipped with a:

- Schneider Electric WiFi dongle, called the *wifer*, part number TCSEGWB13FA0.
- PMXNOW0300 wireless module.

Status Summary (Hot Standby CPUs)

Introduction

The **Status Summary** web page provides this information about the CPU:









- Ethernet service diagnostic information
- Version descriptions for installed firmware and software
- CPU description and operating state
- IP addressing settings

NOTE: The **Status Summary** web page is refreshed every 5 seconds.






Open the Page

Access the **Status Summary** page on the **Diagnostics** tab (**Menu → Module → Status Summary**):

Status Summary

 RUN	ERR	 I/O	DL
 REMOTE RUN		 BACKUP	
 ETH MS		 ETH NS	
 A		 PRIM	
FORCED_IO	B		STBY

Service Status

-  DHCP Server **Enabled**
-  FDR Server **Enabled**
-  Access Control **Disabled**
-  [Scanner Status](#) **One Connection Is Bad**
-  [NTP Status](#) **Disabled**

FDR Server0.04%

CPU Summary

Model	M580 CPU
State	RUN
Scan Time	2ms

Version Info.

Exec. Version	2.01
Web Server Version	1.0
Web Site Version	V2.01 IR02
CIP Version	1.0

Network Info.

IP Address	192.168.10.1
Subnet Address	255.255.0.0
Gateway Address	0.0.0.0

346

EIO0000001578 09/2017

Diagnostic and Status Information

The **Status Summary** web page provides this information:

Parameters	Description
LEDs	The web page displays the state of these LEDs:
	<ul style="list-style-type: none"> ● RUN ● ERR ● I/O ● DL ● REMOTE RUN ● BACKUP ● ETH MS ● ETH NS ● A ● B ● PRIM ● STBY ● FORCED_IO
	NOTE: The LEDs on the web page behave the same as the LEDs on the CPU (<i>see page 50</i>).
Service Status	This area presents information describing the status of CPU Ethernet services. The colored icons appearing to the left of some items indicate the following status:
	green The available service is operational and running.
	red An error is detected in an available service.
	black The available service is not present or not configured.
	The status of these Ethernet services is included:
	<ul style="list-style-type: none"> ● DHCP Server ● FDR Server ● Access Control ● Scanner Status ● NTP Status ● FDR Usage
Version Info.	This area describes the software versions that are running on the CPU, including:
	<ul style="list-style-type: none"> ● Executable Version ● Web Server Version ● Web Site Version ● CIP Version
CPU Summary	This area describes the CPU hardware and the applications that are running on the CPU, including: <ul style="list-style-type: none"> ● Model ● State ● Scan Time
Network Info.	This field contains IP addressing settings for the CPU, including: <ul style="list-style-type: none"> ● IP Address ● Subnet Address ● Gateway Address

HSBY Status

Introduction

The **HSBY Status** web page provides this information about the Hot Standby system:

- Hot Standby role and status of the **Local** CPU
- Hot Standby role and status of the **Remote** CPU
- General errors detected for the Hot Standby system

NOTE:

- The local CPU is the CPU configured with the **Main IP Address** (primary) or **Main IP Address + 1** (standby) used to access this web page.
- The **HSBY Status** web page is refreshed every 5 seconds.

Open the Page

Access the **HSBY Status** page from the **Diagnostics** tab (**Menu → Module → HSBY Status**):

HSBY Status

Local

Primary	Run
A	Online
IP Address	192.168.10.1
OS Firmware Level	3
Sync Link Validity	OK
Supplementary Link Validity	OK
Detected Errors:	
None	

Remote

Standby	Run
B	Online
IP Address	192.168.10.2
OS Firmware Level	3
Sync Link Validity	OK
Supplementary Link Validity	OK
Detected Errors:	
None	

General Errors

None

Diagnostic and Status Information

The **HSBY Status** web page provides this information:

Area	Description
Local/Remote	This area displays the state of Hot Standby settings for the local and remote CPUs:
	<div> <div><Hot Standby Role></div> <div> <p>The Hot Standby system role of the CPU. Valid values include:</p> <ul style="list-style-type: none"> ● Primary ● Standby ● Wait </div> </div>
	<div> <div><Operating State></div> <div> <p>The operating state of the CPU. Valid values include:</p> <ul style="list-style-type: none"> ● RUN ● STOP ● NoConf ● HALT </div> </div>
	<div> <div>A/B switch setting</div> <div> <p>The designation of the CPU, defined by the rotary switch (<i>see page 44</i>) on the back of the CPU. Valid values include:</p> <ul style="list-style-type: none"> ● A ● B </div> </div>
	<div> <div><Run Mode></div> <div> <p>The designation of the CPU, defined by the rotary switch on the back of the CPU. Valid values include:</p> <ul style="list-style-type: none"> ● Online ● Wait </div> </div>
	<div> <div>IP Address</div> <div> <p>The IP address used to communicate with the CPU for web page access:</p> <ul style="list-style-type: none"> ● For the primary Hot Standby CPU, this is the Main IP Address setting. ● For the standby Hot Standby CPU, this is the Main IP Address setting + 1. </div> </div>
	<div> <div>OS Firmware Level</div> <div> <p>Firmware version of the CPU operating system.</p> </div> </div>
	<div> <div>Sync Link Validity</div> <div> <p>The status of the Hot Standby link (<i>see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures</i>):</p> <ul style="list-style-type: none"> ● OK: the link is operational. ● NOK: the link is not operational. </div> </div>
	<div> <div>Supplementary Link Validity</div> <div> <p>The status of the Ethernet RIO link (<i>see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures</i>):</p> <ul style="list-style-type: none"> ● OK: the link is operational. ● NOK: the link is not operational. </div> </div>
	<div> <div>Detected Errors</div> <div> <p>Detected errors for the CPU, including:</p> <ul style="list-style-type: none"> ● HSBY link error detected ● RIO link error detected (the connection between PAC A and PAC B over the Ethernet RIO network) ● RIO error detected (the connection between a PAC and (e)X80 EIO adapter modules over the Ethernet RIO network. </div> </div>

Area	Description
General Errors	Detected errors for the Hot Standby system, including: <ul style="list-style-type: none">● Application mismatch● Logic mismatch● Firmware mismatch● Data layout mismatch● Backup application mismatch

Rack Viewer

Introducing the CPU Status Page

The BMEH584040 and BMEH586040 Hot Standby CPUs include a **Rack Viewer** web page. Use this page to view CPU information, including:

- LEDs status
- processor identification
- application signature identification
- select application configuration settings

Accessing the Rack Viewer Page

Access the **Rack Viewer** page from the **Diagnostics** menu. In the navigation menu at the left side of the page, select **Menu → System → Rack Viewer**:

BME H58 6040 : Bus 0 Drop 0 Rack 0 Slot 0

● RUN
● ERR
● I/O

Processor	
RAM Size(kb):	131072 KB
Processor Version:	2.01 - 2
Hardware Id:	2330B0E
State:	Run
Error:	0X0C8A
Calendar:	June 02 2015 15:56:26

CID:	208032960
MID:	19649345
AID:	0
LID:	19649345
DID:	19649345

Application	
Name:	"Project"
Version:	2
Creation Product:	Unity Pro XLV11.01.01.150422-May 29, Friday...
Modification Product:	Unity Pro XLV11.01.01.150422-May 29, Friday...
Forced bit:	0
Analog channel forced:	FALSE

Events Disabled:	FALSE
Section Protected:	FALSE
Automatic Start in Run:	FALSE
RAZ %MW on cold start:	FALSE
Cold Start only:	FALSE
Diagnostic:	TRUE

Rack Viewer Data

The **Rack Viewer** page for M580 Hot Standby CPUs displays the following data:

Data Field	Description
Processor	
RAM size (kb)	The size of processor RAM in KB
Processor Version	Firmware version
Hardware ID	An identifier for the module hardware. OS Loader checks this value to determine compatibility between the hardware and the operating system.
State	<p>The operating state of the processor:</p> <ul style="list-style-type: none"> • NO CONFIGURATION • IDLE • STOP • RUN • HALT • INITIALIZING • ERROR • OS LOADER
Error	The identity of the last detected error
Calendar	Date and time of last detected error
Signature	
CID	<i>Creation ID</i> : Random number generated when an application is created. The number remains a constant.
MID	<i>Modification ID</i> : Random number generated on each application modification and rebuild, either partial or global. When an application is created, MID = CID.
AID	<p><i>AutoModification ID</i>: A new random value is generated for AID by the PAC after one of the following minor modifications to the application:</p> <ul style="list-style-type: none"> • a Unity request to modify %KW • a P_Unit request that performs a save_param request or replaces init value <p>When an application is created or built in local mode, AID = 0.</p>
LID	<p><i>Layout ID</i>: Random number generated after a modification of the variable layout. LID does not change as a result of a runtime change either adding or deleting a data block. LID changes only on when the global rebuild of the application.</p> <p>LID addresses the needs of Hot Standby. It permits the transfer of a memory block from the primary PAC to the standby so that application variables (except for deleted or new ones) exist at the same location.</p> <p>LID = CID = MID when the application is created.</p>
DID	<i>Data ID</i> : Indicates that a block of data has been freed. Also used for the special case of remapping a symbol from unlocated to located.

Data Field	Description
Application	
Name	Name of the Unity Pro project
Version	Project version
Creation Product	Includes both: <ul style="list-style-type: none"> ● Version and build of Unity Pro used to create the project. ● Date and time the project was created.
Modification Product	Includes both: <ul style="list-style-type: none"> ● Version and build of Unity Pro used to edit the project. ● Date and time the project was last edited.
Events Disabled	Indicates if all event processing has been disabled: <ul style="list-style-type: none"> ● True indicates all event processing has been disabled. ● False indicates event processing has not been disabled. <p>NOTE: Events can be enabled/disabled by using:</p> <ul style="list-style-type: none"> ● The Enable or Disable all command (<i>see Unity Pro, Operating Modes</i>) in the Task tab of the CPU. ● The MASKEVT and UNMASKEVT functions. ● System bit %S38.
Forced bit	The number of forced bits in the application.
Analog channel forced:	Indicates if one or more inputs or outputs for an analog channel have been forced: <ul style="list-style-type: none"> ● True indicates the an analog input or output has been forced. ● False indicates no analog input or output has been forced.
Last Stop	The event that last caused the application to stop. Values include: <ul style="list-style-type: none"> ● Changeover from RUN to STOP by the terminal or dedicated input ● Stop on software detected fault (task overrun or SFC overrun) ● Power loss detected ● Stop on hardware detected fault ● Stop on HALT instruction
Last Stop Date	Date an event last caused the application to stop.
Section protected	Indicates if password access is required to edit one or more sections of the application: <ul style="list-style-type: none"> ● True indicates that a password is required to edit specified sections of the application. ● False indicates that no password is required for application editing.
Automatic Start in Run	Indicates if the application is automatically set to start when the PAC goes into RUN operational mode: <ul style="list-style-type: none"> ● True indicates the application automatically starts. ● False indicates the application does not automatically start.
RAZ %MW on cold start	Indicates if %MW registers are reset to their initial values on a cold start: <ul style="list-style-type: none"> ● True indicates that values are reset. ● False indicates that values are not reset.

Data Field	Description
Cold Start only	Indicates if a cold start is forced on a system re-start: <ul style="list-style-type: none">● True indicates that a reset forces a cold start of the application.● False indicates that a warm start will occur on application reset.
Diagnostic	Indicates if the diagnostic buffer has been activated for the project: <ul style="list-style-type: none">● True indicates that Application diagnostics and/or System diagnostics has been selected in the General → PAC Diagnostics tab of the Project Settings dialog for the application.● False indicates Application diagnostics and System diagnostics have not been selected.

Chapter 6

M580 CPU Programming and Operating Modes

Overview

This chapter provides information on M580 CPU I/O exchanges, tasks, memory structure, and operating modes.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	I/O and Task Management	356
6.2	BMEP58xxxx CPU Memory Structure	361
6.3	BMEP58xxxx CPU Operating Modes	362

Section 6.1

I/O and Task Management

Overview

This section presents information on M580 I/O addressing and management, tasks allowed, and I/O scanning capabilities.

What Is in This Section?

This section contains the following topics:

Topic	Page
I/O Exchanges	357
CPU Tasks	359

I/O Exchanges

I/O Vision

Each module uses a structure that represents inputs, outputs, control, and diagnostic data. The structures can be represented using:

- topological addressing / IODDT
- Device DDT

I/O Module Location	I/O Family	Topological Addressing / IODDT	Device DDT
local rack	(e)X80	X	X
	Premium	X	–
RIO	(e)X80	–	X
	Quantum	–	X
distributed equipment	Schneider Electric or third party	–	X
X Supported. When both visions are supported, select one of the exchange types when adding the equipment. – Not supported.			

Adding an I/O Module in Unity Pro

When you insert an I/O module on a rack in Unity Pro, the type of addressing appears in the bottom of the **New Device** dialog box. Choose between the following:

- **I/O data type: Topological** (default)
- **I/O data type: Device DDT**

NOTE: If you want to change the type of addressing you selected when you added an I/O module to your application, delete the module from your application and then insert the module again selecting the appropriate addressing type.

Exchange Types

I/O modules in an M580 system can be controlled, read, or written with 2 types of exchanges:

- implicit exchanges
Implicit exchanges are performed automatically on each cycle of the task (MAST, FAST, AUX0, AUX1) associated with the I/O modules. They are used to read inputs from and write outputs of the modules.
- explicit exchanges
Explicit exchanges are performed on application request. They are typically for detailed diagnostics and to set/read command and adjust parameters. They use specific function blocks. An acknowledgment or reply is sent once the requested action is performed. This reply may be received a few cycles after the request was sent.

NOTE: Explicit exchanges are performed in the MAST task.

Explicit Exchanges

Function block usage depends on the module location and I/O vision selected for the module:

I/O Module Location	I/O Vision	Function Block
Local rack	Topological addressing/IODDT	READ_PARAM
		READ_STS
		READ_TOPO_ADDR
		RESTORE_PARAM
		SAVE_PARAM
		WRITE_CMD
		WRITE_PARAM
		READ_VAR
		WRITE_VAR
		DATA_EXCH
	Device DDT	READ_PARAM_MX
		READ_STS_MX
		NOTE: MOD_FAULT parameter is not automatically updated; perform a READ_STS_MX.
		RESTORE_PARAM_MX
		SAVE_PARAM_MX
		WRITE_CMD_MX
		WRITE_PARAM_MX
RIO and local rack	Device DDT	READ_STS_MX
		WRITE_CMD_MX

The function blocks mentioned in previous table are detailed in the *Explicit Exchange* part of *Unity Pro, I/O Management, Block Library manual*, and in the *Extended* part of *Unity Pro, Communication, Block Library manual*.

CPU Tasks

Introduction

An M580 CPU can execute single-task and multi-task applications. Unlike a single-task application which only executes the MAST task, a multi-task application defines the priorities of each task.

There are four tasks available (see *Application Program Structure* chapter in *Unity Pro Program Languages and Structure Reference Manual*) and two types of event tasks:

- MAST
- FAST
- AUX0
- AUX1
- I/O event in a local rack only
- timer event in a local rack only

NOTE: The time to perform an *update init values with current values* operation is not taken into account in the watchdog calculation.

Task Characteristics

The time model, task period, and maximum number of tasks per CPU are defined according to the standalone or Hot Standby CPU reference.

Standalone CPUs:

Task	Time Model	Task Period (ms)		BM58 References					
		Range	Default Value	1020 (H)	20•0 (H)	30•0	40•0	5040	6040
MAST ^(1.)	cyclic ^(2.) or periodic	1...255	20	X	X	X	X	X	X
FAST	periodic	1...255	5	X	X	X	X	X	X
AUX0	periodic	10...2550 by 10	100	X	X	X	X	X	X
AUX1	periodic	10...2550 by 10	200	X	X	X	X	X	X
<ol style="list-style-type: none"> 1. MAST task is mandatory. 2. When set to cyclic mode, the minimum cycle time is 8 ms if there is a RIO network and 1 ms if there is no RIO network in the system. <p>X This task is supported.</p>									

Hot Standby CPUs:

Task	Time Model	Task Period (ms)		CPU Reference (BMEH58 ...)		
		Range	Default Value	2040	4040	6040
MAST ^(1.)	periodic ^(2.)	1...255	20	X	X	X
FAST ^(3.)	periodic	1...255	5	X	X	X
AUX0 ^(4.)	—	—	—	—	—	—
AUX1 ^(4.)	—	—	—	—	—	—
<div><div>1. MAST task is mandatory.</div><div>2. Only periodic is supported; cyclic is not supported.</div><div>3. Supported for (e)X80 ERIO drops.</div><div>4. Not supported.</div><div>X This task is supported.</div></div>						

Section 6.2

BMEP58_{XXXX} CPU Memory Structure

Memory Structure

CPU Memory

3 types of memories are available in a BMEP58_{XXXX} CPU:

- non-persistent application RAM: run the application program and store temporary data
- flash memory: back up the application program and a copy of %MW values
- optional SD memory card: store application and data in parallel to the CPU flash memory, allowing a fast CPU hardware replacement

Application Download to the CPU Memory

CPU memory involved during an application download from a programming terminal:

- Application is transferred into the non-persistent application RAM.
- If a memory card is inserted, working and not write protected, then an internal backup is performed in the memory card.
- The application backup is performed in the the flash memory.

NOTE: A write protected memory card inserted disables the application download.

Application Upload from the CPU Memory

The application upload reads and copies non-persistent application content from RAM to your selected location.

Application Online Modification Backup

An application program modification is performed in the CPU non-persistent memory with an automatic backup performed as follows:

- If a memory card is inserted, working and not write protected, then the backup is performed in the memory card.
- The application backup is performed in the flash memory.

NOTE: The online modification is disabled when a write protected memory card is inserted.

Application Memory Self Modification

The user code may modify the application content (for example to save I/O parameters or replace variables initial value by the current value).

In such a case, only the non-persistent application RAM content is modified.

To back up the application in the memory card and to the flash memory, use the system bit %S66.

Section 6.3

BMEP58~~xxxx~~ CPU Operating Modes

Overview

This section provides information on the CPU operating modes.

What Is in This Section?

This section contains the following topics:

Topic	Page
Managing Run/Stop Input	363
Power Cut and Restore	364
Cold Start	366
Warm Restart	369

Managing Run/Stop Input

Input Run/Stop

The `%lr.m.c` input can be parameterized to switch the PAC to **Run/Stop** mode as follows:

- Set `%lr.m.c` to 1: The PAC switches to **Run** mode (executing the program).
- Set `%lr.m.c` to 0: The PAC switches to **Stop** mode (stopping program execution).

NOTE: A Stop command takes priority over a Run command. A Stop command sent from a terminal or via the network has priority over the `%lr.m.c` input.

An error detected on the Run/Stop input causes the PAC to switch to **Stop** mode.

Do not enable this option if the associated discrete input is mapped in state RAM because this inhibits the start-up of the PAC.

Memory Protect

The input `%lr.m.c` can be parameterized to protect the internal application RAM and the memory card as follows:

- `%lr.m.c` to 0: The internal application and the memory card **are not** protected.
- `%lr.m.c` to 1: The internal application and the memory card **are** protected.

NOTE: If the input is in error, `%lr.m.c` is considered at 1 (memory is protected). To remove this protection in the configuration screen, the input should not be in error.

Managing Run/Stop Remote Access

When configuring the M580 CPU, you can help prevent remote commands/requests from accessing the CPU **Run/Stop** modes. Select the respective **Run/Stop input** and **Run/Stop by input only** check boxes according to the following table parameters to determine the type of remote access for your system.

Run/Stop Input	Run/Stop by Input Only	Description
–	–	Allows remote access to run/stop the CPU by request.
X	–	<ul style="list-style-type: none"> • Allows remote access to stop the CPU by request • You can run the CPU by input only.
X	X	Denies remote access to run/stop the CPU by request.
X: check box selected –: check box deselected		

Power Cut and Restore

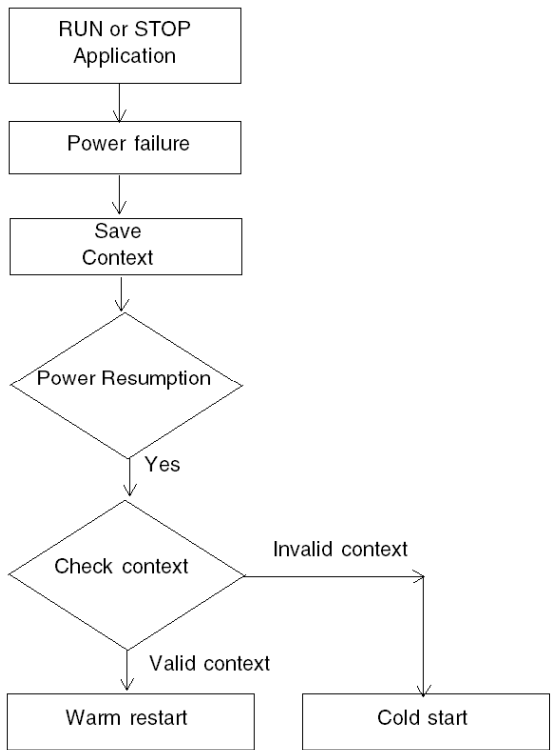
Introduction

If the duration of the outage is shorter than the power supply filtering time, it has no effect on the program which continues to run normally.

If the duration of the outage is longer than the power supply filtering time, the program is interrupted and power restoration processing is activated. The CPU then restarts in warm restart or cold start as described in the following diagram.

Illustration

Power cycle phases:



Power Supply Filtering Times

The BMX CPS 2000, BMX CPS 3500, and BMX CPS 3540T power supplies, which provide Vac power, have a filtering time of 10 ms.

The BMX CPS 2010 and BMX CPS 3020 power supplies, which provide Vdc power, have a filtering time of 1 ms.

Power Outage Processing Phases

When power to the system is lost, it recovers in 3 phases:

Phase	Description
1	On power outage, the system saves the application context, the values of application variables, and the state of the system on internal flash memory.
2	The system sets all the outputs into fallback state (state defined in configuration).
3	<p>On power restoral, some actions and checks are done to verify if warm restart is available:</p> <ul style="list-style-type: none">● restore internal flash memory application context● verify application and context validity <p>If all checks are correct a warm restart (see page 369) is performed, otherwise a cold start (see page 366) is carried out.</p>

Cold Start

CPU Cold Start Causes and States

Cold start causes and resulting CPU states:

Cause	Resulting CPU State
End of the application download.	STOP
Application restored from flash memory is different than the one in the non-persistent application RAM. Use case: <ul style="list-style-type: none"> ● application restored from a memory card if a compatible memory card is in the card slot ● application restored from the CPU flash memory 	STOP ^(1.)
Application restored from persistent memory with Unity Pro command PLC → Project backup → is different than the one in the non-persistent application RAM: <ul style="list-style-type: none"> ● application restored from a memory card if a compatible memory card is in the card slot ● application restored from the CPU flash memory 	STOP ^(1.)
Power supply RESET button pressed.	STOP ^(1.)
Power supply RESET button pressed less than 500 ms after a power down.	STOP ^(1.)
Power supply RESET button pressed after a CPU detected error, except in the case of a watchdog detected error (halt state).	STOP ^(2.)
Init requested with one of the 3 following means: <ul style="list-style-type: none"> ● %S0 system bit set to 0 ● INIT request ● Cold Start command in Unity Pro 	The CPU does not change its state. It only initializes the application. It is a simulation of cold start.
Restoral after power down with a loss of context.	STOP ^(1.)
1. CPU state is set to RUN if Automatic start in Run option is selected. 2. Automatic start in Run option does not set the CPU to RUN state.	

Loading or transferring an application to the CPU involves initialization of unlocated variables.

You need to assign a topological address to the data if the process requires keeping the current values of the data when transferring the application.

To save the located variables, avoid the initialization of the %MWi by unchecking **Initialize %MWi on cold start** parameter in the CPU configuration screen.

NOTE: Pressing the **RESET** button on the power supply resets %MWi and initial values are loaded.

NOTE: Do not press the **RESET** button on the power supply if you do not want %MWi to be reset and loaded with initial values.

Executing a Cold Start

Use these steps to perform a cold start:

Phase	Description
1	<p>The startup is performed in RUN or in STOP state depending on one of the 2 following conditions:</p> <ul style="list-style-type: none"> ● The status of the Automatic start in Run parameter defined in the CPU configuration. If the parameter is selected, the start will be performed in RUN. ● The state of the I/O defined in the Run/Stop input parameter in the CPU configuration. <p>Program execution is resumed at the start of the cycle.</p>
2	<p>The system carries out the following:</p> <ul style="list-style-type: none"> ● Disable FAST, AUX, and event tasks. ● MAST task is executed until the end of data initialization. ● Initialize data (bits, I/O image, words, and so on) with the initial values defined in the data editor (value set to 0 if no other initial value has been defined). For %MW words, the values can be retrieved on a cold start when these conditions are met: <ul style="list-style-type: none"> ○ The Initialize %MWi on cold start parameter is not checked in the CPU configuration screen, ○ The internal flash memory has a valid backup (see %SW96). <p>NOTE: If the number of %MW words exceeds the backup size during the save operation the remaining words are set to 0.</p> <ul style="list-style-type: none"> ● Initialize elementary function blocks (initial data). ● Initialize data declared in the DFBs: either to 0 or to the initial value declared in the DFB type. ● Initialize system bits and words. ● Position charts to initial steps. ● Cancel any forcing action. ● Initialize message and event queues. ● Send configuration parameters to all I/O and application-specific modules.
3	<p>To start a cycle, the system performs these tasks:</p> <ul style="list-style-type: none"> ● Relaunch the MAST task with the %S0 (cold start) and %S13 (first cycle in RUN) system bits set to 1. %SW10 (first cycle after cold start) system word is set to 0. ● Reset the %S0 and %S13 system bits to 0 and set each bit of %SW10 system word to 1 at the end of this first cycle of the MAST task. ● Activate the FAST and AUX tasks and event processing at the end of the first cycle of the MAST task.

Processing a Cold Start by Program

Test %SW10.0 system bit to detect a cold start and adapt the program consequently.

NOTE: It is possible to test the %S0 system bit on the first execution cycle if the **Automatic start in RUN** parameter is selected. If it is not selected, the CPU starts in STOP state and the bit %S0 switches to 1 on the first cycle after start (not visible for the program).

Output Changes

As soon as a power outage is detected the outputs are set in the fallback position configured (programmed fallback value or current value).

On power down, the outputs are not driven and remain at 0.

After power restoral, the outputs remain at 0 until they are updated by the task.

Warm Restart

Introduction

A warm restart occurs after a power cycle.

Executing a Warm Restart

Phase	Description
1	Program execution does not resume from the element where the power outage occurred. The remaining program is discarded during the warm restart. Each task restarts from the beginning.
2	<p>The system carries out the following:</p> <ul style="list-style-type: none"> ● Restore the application variables value, ● Set %S1 system bit to 1. ● Initialize message and event queues, ● Send configuration parameters to all I/O and application-specific modules, ● If the application was reserved, the CPU removes the reservation. ● Reset communication. ● If needed, the CPU configures the I/O modules with the current adjustment parameters. ● Disable FAST, AUX, and event tasks.
3	<p>The system performs a restart cycle during which it:</p> <ul style="list-style-type: none"> ● Restarts the MAST task from beginning of cycle, ● Sets %S1 system bit to 0 when the MAST task is completed. ● Enable FAST, AUX, and event tasks at the end of the first MAST task cycle. ● CPU state set to the value before power down. <p>If the CPU was in HALT state, it is set to STOP state.</p>

Processing a Warm Restart by Program

On warm restart, if the application needs to be processed in a particular way, the program needs to test that %S1 system bit is set to 1 at the start of the MAST task program.

SFC Warm Restart Specific Features

The warm start on Modicon M580 CPU is not considered as a real warm start by the CPU. SFC interpreter does not depend on tasks.

SFC publishes a `ws_data` memory area to the OS that contains SFC section-specific data to be saved on power down.

At the beginning of chart processing the active steps are saved to `ws_data` and processing is marked to be in a section that is essential to the applicatoin. At the end of chart processing the essential section is unmarked.

If a power down hits into the essential section, it could be detected if this state is active at the beginning (as the scan is aborted and MAST task is restarted from the beginning). In this case, the workspace may be inconsistent and is restored from the saved data.

Additional information from `SFCSTEP_STATE` variable in located data area is used to reconstruct the state machine.

When a power down occurs, the following is performed:

- During first scan, `%S1 = 1`, MAST task is executed but FAST and event tasks are not executed.

On power restoral, the following is performed:

- clear chart, deregister diagnostics, keep set actions
- set steps from saved area
- set step times from `SFCSTEP_STATE`
- suppress execution of the P / P1 actions
- restores elapsed time for timed actions

NOTE: SFC interpreter is independent, if the transition is valid, the SFC chart evolves while `%S1 = 1`.

Output Changes

As soon as a power outage is detected the outputs are set in the fallback position configured: either programmed fallback value or current value.

After power restoral, the outputs remain at 0 until they are updated by the task.

Appendices



Appendix A

Function Blocks

ETH_PORT_CTRL: Executing a Security Command in an Application

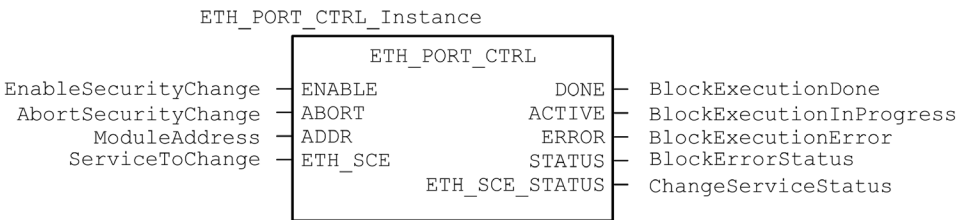
Function Description

Use the ETH_PORT_CTRL function block to control the FTP TFTP, HTTP, and DHCP / BOOTP protocols when they are enabled in the Unity Pro **Security** screen (*see Modicon M580, BMENOC0301/0311 Ethernet Communications Module, Installation and Configuration Guide*). (By default, these protocols are disabled.) For cyber security reasons (to help protect data against requests to modify in the monitoring mode), map the inputs on variables and on unlocated variables in which the HMI property is disabled (the variable is not in the data dictionary).

The additional parameters EN and ENO may also be configured.

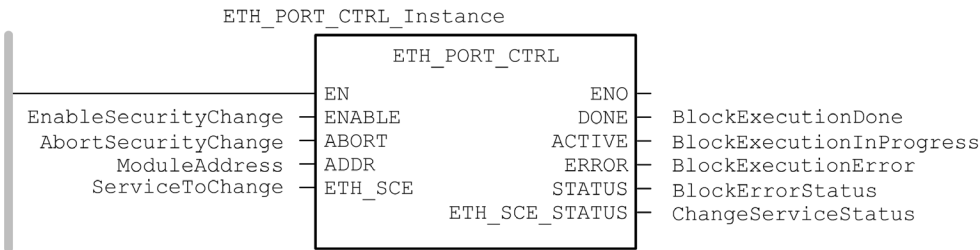
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

```
CAL ETH_PORT_CTRL_Instance (ENABLE := EnableSecurityChange, ABORT :=
AbortSecurityChange, ADDR := ModuleAddress, ETH_SCE := ServiceToChange,
DONE => BlockExecutionDone, ACTIVE => BlockExecutionInProgress, ERROR
=> BlockExecutionError, STATUS => BlockErrorStatus, ETH_SCE_STATUS =>
ChangeServiceStatus)
```

ST Representation

```
ETH_PORT_CTRL_Instance (ENABLE := EnableSecurityChange, ABORT :=
AbortSecurityChange, ADDR := ModuleAddress, ETH_SCE := ServiceToChange,
DONE => BlockExecutionDone, ACTIVE => BlockExecutionInProgress, ERROR
=> BlockExecutionError, STATUS => BlockErrorStatus, ETH_SCE_STATUS =>
ChangeServiceStatus);
```

Description of Parameters

This table describes the input parameters:

Parameter	Type	Comment
ENABLE	BOOL	Set to 1 to enable the operation.
ABORT	BOOL	Set to 1 to abort the currently active operation.
ADDR	ANY_ARRAY_INT	<p>This array contains the address of the entity for which you want to change the security state, which is the result of the ADDMX (<i>see Unity Pro, Communication, Block Library</i>) or ADDMX or ADDM function (<i>see Unity Pro, Communication, Block Library</i>). For example:</p> <ul style="list-style-type: none"> ● ADDM('0.0.10') for a M580 CPU ● ADDM('0.3.0') for a BMENOC0301/11 plugged in slot 3 of main rack
ETH_SCE	WORD	<p>For each protocol, use these binary values to control the protocol:</p> <ul style="list-style-type: none"> ● 00: The protocol is unchanged. ● 01: Enable the protocol. ● 10: Disable the protocol. ● 11: reserved <p>NOTE: A value of 11 reports a detected error in ETH_SCE_STATUS.</p> <p>These bits are used for the different protocols:</p> <ul style="list-style-type: none"> ● 0, 1: FTP ● 2, 3: TFTP (Only available for Modicon M580) ● 4, 5: HTTP ● 6, 7: DHCP / BOOTP ● 8...15: reserved (value = 0)
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

This table describes the output parameters:

Parameter	Type	Comment
DONE	BOOL	Operation completed indication. Set to 1 when the execution of the operation is completed successfully.
ACTIVE	BOOL	Operation in progress indication. Set to 1 when the execution of the operation is in progress.
ERROR	BOOL	Set to 1 if an error is detected by the function block.
STATUS	WORD	Code providing the detected error identification (<i>see Unity Pro, I/O Management, Block Library</i>).
ETH_SCE_STATUS	WORD	<p>For each protocol, these values contain the response to any attempt to enable or disable the FTP, TFTP, HTTP, or DHCP / BOOTP protocols:</p> <ul style="list-style-type: none"> ● 0: command executed ● 1: command not executed <p>Reasons for not executing the command can be:</p> <ul style="list-style-type: none"> ● The communication service has been disabled by the configuration. ● The communication service is already in the state requested by the command (Enabled or Disabled). ● The communication service (x) is not supported by the module or is a non-existing service. <p>These bits are used for the different protocols:</p> <ul style="list-style-type: none"> ● 0: FTP ● 1: TFTP ● 2: HTTP ● 3: DHCP / BOOTP ● 4 ... 15: reserved (value = 0)

Execution Type

Synchronous:

When used on the following M580 CPU modules, the ETH_PORT_CTRL function block is executed **synchronously**. As a result, the DONE output turns **ON** as soon as the ENABLE input is set to **ON**. In this case, the ACTIVE output remains **OFF**.

- BM581020
- BM582020
- BM582040
- BM583020
- BM583040
- BM584020
- BM584040
- BM585040
- BM586040

- BMEH582040*
- BMEH584040*
- BMEH586040*

* In BMEH58*040 Hot Standby CPUs, verify that the ETH_PORT_CTRL function block is executed equally on both primary and standby CPUs.

Asynchronous:

When used on the following modules, the ETH_PORT_CTRL function block is executed *asynchronously* and may take several cycles until the DONE output turns **ON**. Therefore, the ACTIVE output is set to **ON** until the completion of the ETH_PORT_CTRL function block.

- **M340 modules:**
 - BMXNOC0401
 - BMXNOE0100
 - BMXNOE0110
- **M580 modules:**
 - BMENOC0301/11

How to Use the ETH_PORT_CTRL EFB

Use the ETH_PORT_CTRL EFB:

Step	Action
1	Set the bits of the services you want to activate in ETH_SCE.
2	Set ENABLE input to activate the EFB.
3	ENABLE input should be an OR between a pulse command and the ACTIVE output of the EFB.
4	Check STATUS output value: <ul style="list-style-type: none">● STATUS<>0: There is a communication issue.● STATUS = 0: Check ETH_SCE_STATUS. The services for which the bits are set haven't been modified as they should be.



!

%MW

According to the CEI standard, %MW indicates a language object of type memory word.

A

adapter

An adapter is the target of real-time I/O data connection requests from scanners. It cannot send or receive real-time I/O data unless it is configured to do so by a scanner, and it does not store or originate the data communications parameters necessary to establish the connection. An adapter accepts explicit message requests (connected and unconnected) from other devices.

B

BCD

(*binary-coded decimal*) Binary encoding of decimal numbers.

BOOTP

(*bootstrap protocol*) A UDP network protocol that can be used by a network client to automatically obtain an IP address from a server. The client identifies itself to the server using its MAC address. The server, which maintains a pre-configured table of client device MAC addresses and associated IP addresses, sends the client its defined IP address. The BOOTP service utilizes UDP ports 67 and 68.

C

CCOTF

(*change configuration on the fly*) A feature of Unity Pro that allows a module hardware change in the system configuration while the system is operating. This change does not impact active operations.

CIP™

(*common industrial protocol*) A comprehensive suite of messages and services for the collection of manufacturing automation applications (control, safety, synchronization, motion, configuration and information). CIP allows users to integrate these manufacturing applications with enterprise-level Ethernet networks and the internet. CIP is the core protocol of EtherNet/IP.

CPU

(*central processing unit*) The CPU, also known as the processor or controller, is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. CPUs are computers suited to survive the harsh conditions of an industrial environment.

D**determinism**

For a defined application and architecture, you can predict that the delay between an event (change of value of an input) and the corresponding change of a controller output is a finite time t , smaller than the deadline required by your process.

Device DDT (DDDT)

A Device DDT is a DDT predefined by the manufacturer and not modifiable by user. It contains the I/O language elements of an I/O module.

device network

An Ethernet-based network within an RIO network that contains both RIO and distributed equipment. Devices connected on this network follow specific rules to allow RIO determinism.

DFB

(*derived function block*) DFB types are function blocks that can be defined by the user in ST, IL, LD or FBD language.

Using these DFB types in an application makes it possible to:

- simplify the design and entry of the program
- make the program easier to read
- make it easier to debug
- reduce the amount of code generated

DHCP

(*dynamic host configuration protocol*) An extension of the BOOTP communications protocol that provides for the automatic assignment of IP addressing settings, including IP address, subnet mask, gateway IP address, and DNS server names. DHCP does not require the maintenance of a table identifying each network device. The client identifies itself to the DHCP server using either its MAC address, or a uniquely assigned device identifier. The DHCP service utilizes UDP ports 67 and 68.

DIO

(*distributed I/O*) Also known as distributed equipment. DRSS use DIO ports to connect distributed equipment.

DIO cloud

A group of distributed equipment that is not required to support RSTP. DIO clouds require only a single (non-ring) copper wire connection. They can be connected to some of the copper ports on DRSSs, or they can be connected directly to the CPU or Ethernet communications modules in the *local rack*. DIO clouds **cannot** be connected to *sub-rings*.

DNS

(*domain name server/service*) A service that translates an alpha-numeric domain name into an IP address, the unique identifier of a device on the network.

DRS

(*dual-ring switch*) A ConneXium extended managed switch that has been configured to operate on an Ethernet network. Predefined configuration files are provided by Schneider Electric to downloaded to a DRS to support the special features of the main ring / sub-ring architecture.

DSCP

(*differentiated service code points*) This 6-bit field is in the header of an IP packet to classify and prioritize traffic.

DTM

(*device type manager*) A DTM is a device driver running on the host PC. It provides a unified structure for accessing device parameters, configuring and operating the devices, and troubleshooting devices. DTMs can range from a simple graphical user interface (GUI) for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes. In the context of a DTM, a device can be a communications module or a remote device on the network.

See FDT.

E**EDS**

(*electronic data sheet*) EDS are simple text files that describe the configuration capabilities of a device. EDS files are generated and maintained by the manufacturer of the device.

EF

(*elementary function*) This is a block used in a program which performs a predefined logical function.

A function does not have any information on the internal state. Several calls to the same function using the same input parameters will return the same output values. You will find information on the graphic form of the function call in the [*functional block (instance)*]. Unlike a call to a function block, function calls include only an output which is not named and whose name is identical to that of the function. In FBD, each call is indicated by a unique [number] via the graphic block. This number is managed automatically and cannot be modified.

Position and configure these functions in your program in order to execute your application.

You can also develop other functions using the SDKC development kit.

EFB

(*elementary function block*) This is a block used in a program which performs a predefined logical function.

EFBs have states and internal parameters. Even if the inputs are identical, the output values may differ. For example, a counter has an output indicating that the preselection value has been reached. This output is set to 1 when the current value is equal to the preselection value.

EIO network

(*Ethernet I/O*) An Ethernet-based network that contains three types of devices:

- local rack
- X80 EIO drop, or a Quantum EIO drop (using a BM•CRA312•0 adapter module), or a BMENOS0300 network option switch module
- ConneXium extended dual-ring switch (DRS)

NOTE: Distributed equipment may also participate in an EIO network via connection to DRSs or the service port of X80 EIO adapter modules.

Ethernet

A 10 Mb/s, 100 Mb/s, or 1 Gb/s, CSMA/CD, frame-based LAN that can run over copper twisted pair or fiber optic cable, or wireless. The IEEE standard 802.3 defines the rules for configuring a wired Ethernet network; the IEEE standard 802.11 defines the rules for configuring a wireless Ethernet network. Common forms include 10BASE-T, 100BASE-TX, and 1000BASE-T, which can utilize category 5e copper twisted pair cables and RJ45 modular connectors.

EtherNet/IP™

A network communication protocol for industrial automation applications that combines the standard internet transmission protocols of TCP/IP and UDP with the application layer common industrial protocol (CIP) to support both high speed data exchange and industrial control. EtherNet/IP employs electronic data sheets (EDS) to classify each network device and its functionality.

explicit messaging

TCP/IP-based messaging for Modbus TCP and EtherNet/IP. It is used for point-to-point, client/server messages that include both data, typically unscheduled information between a client and a server, and routing information. In EtherNet/IP, explicit messaging is considered class 3 type messaging, and can be connection-based or connectionless.

F**FDR**

(*fast device replacement*) A service that uses configuration software to replace an inoperable product.

FDT

(*field device tool*) The technology that harmonizes communication between field devices and the system host.

FTP

(*file transfer protocol*) A protocol that copies a file from one host to another over a TCP/IP-based network, such as the internet. FTP uses a client-server architecture as well as separate control and data connections between the client and server.

G**gateway**

A gateway device interconnects two different networks, sometimes through different network protocols. When it connects networks based on different protocols, a gateway converts a datagram from one protocol stack into the other. When used to connect two IP-based networks, a gateway (also called a router) has two separate IP addresses, one on each network.

H**HMI**

(*human machine interface*) System that allows interaction between a human and a machine.

Hot Standby

A Hot Standby system uses a primary PAC (PLC) and a standby PAC. The two PAC racks have identical hardware and software configurations. The standby PAC monitors the current system status of the primary PAC. If the primary PAC becomes inoperable, high-availability control is maintained when the standby PAC takes control of the system.

HTTP

(*hypertext transfer protocol*) A networking protocol for distributed and collaborative information systems. HTTP is the basis of data communication for the web.

I**implicit messaging**

UDP/IP-based class 1 connected messaging for EtherNet/IP. Implicit messaging maintains an open connection for the scheduled transfer of control data between a producer and consumer. Because an open connection is maintained, each message contains primarily data, without the overhead of object information, plus a connection identifier.

IP address

The 32-bit identifier, consisting of both a network address and a host address assigned to a device connected to a TCP/IP network.

L

local rack

An M580 rack containing the CPU and a power supply. A local rack consists of one or two racks: the main rack and the extended rack, which belongs to the same family as the main rack. The extended rack is optional.

local slave

The functionality offered by Schneider Electric EtherNet/IP communication modules that allows a scanner to take the role of an adapter. The local slave enables the module to publish data via implicit messaging connections. Local slave is typically used in peer-to-peer exchanges between PACs.

M

MAST

A master (MAST) task is a deterministic processor task that is run through its programming software. The MAST task schedules the RIO module logic to be solved in every I/O scan. The MAST task has two sections:

- IN: Inputs are copied to the IN section before execution of the MAST task.
- OUT: Outputs are copied to the OUT section after execution of the MAST task.

MB/TCP

(*Modbus over TCP protocol*) This is a Modbus variant used for communications over TCP/IP networks.

Modbus

Modbus is an application layer messaging protocol. Modbus provides client and server communications between devices connected on different types of buses or networks. Modbus offers many services specified by function codes.

N

NIM

(*network interface module*) A NIM resides in the first position on an STB island (leftmost on the physical setup). The NIM provides the interface between the I/O modules and the fieldbus master. It is the only module on the island that is fieldbus-dependent — a different NIM is available for each fieldbus.

NTP

(*network time protocol*) Protocol for synchronizing computer system clocks. The protocol uses a jitter buffer to resist the effects of variable latency.

P

PAC

programmable automation controller. The PAC is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. PACs are computers suited to survive the harsh conditions of an industrial environment.

port 502

Port 502 of the TCP/IP stack is the well-known port that is reserved for Modbus TCP communications.

R

RIO drop

One of the three types of RIO modules in an Ethernet RIO network. A RIO drop is an M580 rack of I/O modules that are connected to an Ethernet RIO network and managed by an Ethernet RIO adapter module. A drop can be a single rack or a main rack with an extended rack.

RIO network

An Ethernet-based network that contains 3 types of RIO devices: a local rack, an RIO drop, and a ConneXium extended dual-ring switch (DRS). Distributed equipment may also participate in an RIO network via connection to DRSs or BMENOS0300 network option switch modules.

RPI

(requested packet interval) The time period between cyclic data transmissions requested by the scanner. EtherNet/IP devices publish data at the rate specified by the RPI assigned to them by the scanner, and they receive message requests from the scanner at each RPI.

RSTP

(rapid spanning tree protocol) Allows a network design to include spare (redundant) links to provide automatic backup paths if an active link stops working, without the need for loops or manual enabling/disabling of backup links.

S

SFP

(small form-factor pluggable). The SFP transceiver acts as an interface between a module and fiber optic cables.

SNMP

(simple network management protocol) Protocol used in network management systems to monitor network-attached devices. The protocol is part of the internet protocol suite (IP) as defined by the internet engineering task force (IETF), which consists of network management guidelines, including an application layer protocol, a database schema, and a set of data objects.

SNTP

(simple network time protocol) See NTP.

sub-ring

An Ethernet-based network with a loop attached to the main ring, via a dual-ring switch (DRS) or BMENOS0300 network option switch module on the main ring. This network contains RIO or distributed equipment.

T

TCP

(*transmission control protocol*) A key protocol of the internet protocol suite that supports connection-oriented communications, by establishing the connection necessary to transmit an ordered sequence of data over the same communication path.

TFTP

(*trivial file transfer protocol*) A simplified version of *file transfer protocol* (FTP), TFTP uses a client-server architecture to make connections between two devices. From a TFTP client, individual files can be uploaded to or downloaded from the server, using the user datagram protocol (UDP) for transporting data.

trap

A trap is an event directed by an SNMP agent that indicates one of these events:

- A change has occurred in the status of an agent.
- An unauthorized SNMP manager device has attempted to get data from (or change data on) an SNMP agent.

U

UDP

(*user datagram protocol*) A transport layer protocol that supports connectionless communications. Applications running on networked nodes can use UDP to send datagrams to one another. Unlike TCP, UDP does not include preliminary communication to establish data paths or provide data ordering and checking. However, by avoiding the overhead required to provide these features, UDP is faster than TCP. UDP may be the preferred protocol for time-sensitive applications, where dropped datagrams are preferable to delayed datagrams. UDP is the primary transport for implicit messaging in EtherNet/IP.

UMAS

(*Unified Messaging Application Services*) UMAS is a proprietary system protocol that manages communications between Unity Pro and a controller.

UTC

(*coordinated universal time*) Primary time standard used to regulate clocks and time worldwide (close to former GMT time standard).



A

- access control
 - security, 115
- add
 - I/O module, 357
- add remote device, 281
- address
 - field bus, 39
- advanced settings, 131
 - tab, 113
- alarm viewer web page
 - CPU, 339
- altitude, 70
- application
 - legacy, 108
 - password, 103
- assembly object, 169, 173
- asynchronous execution
 - ETH_PORT_CTRL, 373
- authorized address
 - security, 115
- AUTOTEST
 - state, 31
- AUX0 task
 - CPU, 359
- AUX1 task
 - CPU, 359

B

- backup, 108
- blocking condition, 92
- BMEP581020
 - CPU, 19
- BMEP582020
 - CPU, 19
- BMEP582040
 - CPU, 19
- BMEP583020
 - CPU, 19

- BMEP583040
 - CPU, 19
- BMEP584020
 - CPU, 19
- BMEP584040
 - CPU, 19
- BMEP585040
 - CPU, 19
- BMEP586040
 - CPU, 19
- BMXRMS004GPF, 59
- BMXXCAUSB018 USB cables, 53
- BMXXCAUSB045 USB cables, 53
- BOOTP
 - security, 115

C

- certifications, 67
- channel properties, 134
- characteristics
 - current consumption, 35
 - power consumption, 35
- CIP objects, 166
- clear
 - application, 44
- clear local statistics, 273
- clear remote statistics, 274
- cold
 - start, 366
- compatibility
 - CPU, 96
- CONF_SIG
 - device DDT, 211
- configuration
 - CPU, 113
 - Unity Pro, 97
- conformity
 - tests, 67

conformity test

- climatic variations, *72*
- electromagnetic emissions, *72*
- equipment and personnel safety, *72*
- immunity to high frequency interference, *72*
- immunity to low frequency interference, *72*
- mechanical constraints, *72*
- protective enclosure, *72*

connection

- diagnostics, *148*
- I/O, *152*

connection manager object, *171*connection summary, *203*convert, *108*

CPU

- alarm viewer web page, *339*
 - BMEP581020, *19*
 - BMEP582020, *19*
 - BMEP582040, *19*
 - BMEP583020, *19*
 - BMEP583040, *19*
 - BMEP584020, *19*
 - BMEP584040, *19*
 - BMEP585040, *19*
 - BMEP586040, *19*
 - clear, *44*
 - compatibility, *96*
 - configuration, *113*
 - diagnostics, *91*
 - DIO scanner service, *110*
 - front panel, *42*
 - I/O scanner web page, *331*
 - install, *83*
 - LED, *91*
 - memory, *361*
 - memory protect, *103*
 - messaging web page, *333*
 - MTBF, *35*
 - NTP web page, *336*
 - performance web page, *328*
 - physical description, *40*
 - port statistics web page, *329*
 - QoS web page, *334*
 - redundancy web page, *338*
 - role in M580 system, *21*
 - state, *31*
 - status summary web page, *326*
 - task, *359*
 - web page, *325*
- CPU dimensions, *41*
- CPU Ethernet I/O scanner service
- RIO, DIO, *23*
- CPU LEDs, *47*
- CPU scanner service
- RSTP, *121*
- CPU service port, *130*
- CRA_OBJ_CTRL
- device DDT, *211*

CRA_OBJ_HEALTH
 device DDT, 211
 current consumption, 35
 cyber security
 access control, 115
 authorized address, 115
 DHCP/BOOTP, 115
 EIP, 115
 enforce in Unity Pro, 115
 FTP, 115
 HTTP, 115
 memory protect, 103
 password, 103
 SNMP, 115
 TFTP, 115
 unlock in Unity Pro, 115
 cycle
 power, 364

D

DATA_EXCH, 235, 238, 242, 249
 explicit message, 227
 DDT
 LOCAL_HSBY_STS, 219
 REMOTE_HSBY_STS, 219
 T_M_ECPU_HSBY, 219
 default IP address, 42, 82, 83, 113
 device DDT, 314
 Device DDT, 357
 device DDT
 T_BMEP58_ECPU, 211
 T_BMEP58_ECPU_EXT, 211
 device list configuration, 203
 DEVICE_OBJ_CTRL
 device DDT, 211
 DEVICE_OBJ_HEALTH
 device DDT, 211
 DHCP, 136
 security, 115

diagnostics, 140
 bandwidth, 142
 blocking condition, 92
 connection, 148
 CPU, 91
 CPU LEDs, 47
 CPU/system error, 95
 Hot Standby LEDs, 50
 local slave, 148
 memory card, 60
 Modbus codes, 162
 non-blocking condition, 94
 NTP, 146
 RSTP, 144
 web pages, 344
 dimension
 CPU, 41
 DIO scanner service, 110
 RSTP, 121
 selecting CPU, 23
 download, 108
 DTM
 add, 318
 DTM events
 logging to syslog server, 154

E

ECPU_HSBY_1
 device DDT, 211
 EDS file
 add, 319
 remove, 322
 EIO scanner service
 RSTP, 121
 EIP
 security, 115
 elementary functions, 62
 embedded DIO scanner service, 110
 ERROR
 state, 31
 error
 system, 95
 ETH_PORT_1_2_STATUS
 device DDT, 211

- ETH_PORT_3_BKP_STATUS
 - device DDT, *211*
- ETH_PORT_CTRL, *373*
- ETH_STATUS
 - device DDT, *211*
- Ethernet
 - port, *55*
- Ethernet I/O scanner service
 - CPU, *23*
- Ethernet link object, *179*
- EtherNet/IP device
 - explicit message, *252*
- EtherNet/IP explicit connection diagnostics object, *192, 194*
- EtherNet/IP interface diagnostics object, *183*
- EtherNet/IP IO Scanner Diagnostics object, *186*
- events
 - logging to syslog server, *154*
- execution type
 - ETH_PORT_CTRL, *373*
- explicit
 - I/O, *357*
- explicit message, *227*
 - Get_Attribute_Single, *235*
 - Quantum RIO drops in M580, *256*
 - Read Modbus Object, *238*
 - read register, *249*
 - to EtherNet/IP device, *252*
 - to Modbus device, *254*
 - Write Modbus Object, *242*
- explicit messaging
 - EtherNet/IP, *261*
 - EtherNet/IP services, *259*
 - Get_Attributes_Single, *263*
 - MBP_MSTR, *257*
 - Modbus TCP, *269*
 - Modbus TCP function codes, *246, 268*

F

- FAST task
 - CPU, *359*
- FDR, *136*
- field bus address, *39*

- firmware
 - update, *64*
 - upgrade, *64*
- front panel
 - CPU, *42*
- FTP
 - device DDT, *211*
 - SD memory card, *59*
 - security, *115*
- FTP/TFTP services
 - enable/disable, *277*
- function block
 - ETH_PORT_CTRL, *373*

G

- get local statistics, *271*
- get remote statistics, *273*

H

- HALT
 - state, *31*
- hardened, *65*
- HSBY status web page
 - CPU, *348*
- HTTP services
 - enable/disable, *277*
- HTTP)
 - security, *115*
- humidity, *70*

I

- I/O
 - connection, *152*
 - explicit, *357*
 - implicit, *357*
 - local slave, *152*
 - management, *356*
- I/O module
 - add, *357*
- I/O scanner web page
 - CPU, *331*
- identity object, *167*

- IDLE
 - state, *31*
- implicit
 - I/O, *357*
- IN_ERRORS
 - device DDT, *211*
- IN_PACKETS
 - device DDT, *211*
- install
 - CPU, *83*
 - memory card, *88*
 - modules, *81*
- IO connection diagnostics object, *188*
- IODDT, *357*
- IP address
 - default, *42, 83, 113*
 - IP, *82*
- IP address configuration, *119*
- IPConfig
 - tab, *113*
- L**
- LED
 - CPU, *91*
- LEDs
 - CPU, *47*
 - Hot Standby, *50*
- legacy
 - application, *108*
- local slave
 - diagnostics, *148*
 - I/O, *152*
- logging
 - syslog server, *154*
 - to Unity Pro, *153*
- M**
- M580 performance, *23*
- management
 - I/O, *356*
 - task, *356*
- MAST task
 - CPU, *359*
- MBP_MSTR, *257, 261, 263, 269*
 - Quantum RIO drops in M580, *256*
- memory
 - CPU, *361*
- memory card
 - diagnostics, *60*
 - FTP, *59*
 - install, *88*
- memory protect
 - for CPU, *103*
- messaging web page
 - CPU, *333*
- Modbus
 - explicit message, *254*
- module events
 - logging to syslog server, *154*
- modules
 - install, *81*
- MTBF
 - CPU, *35*
- N**
- NOCONF
 - state, *31*
- non-blocking condition, *94*
- NTP
 - diagnostics, *146*
 - RIO scanner service, *126*
 - tab, *113*
- NTP web page
 - CPU, *336*
- O**
- online action, *156*
 - CIP object, *158*
 - ping, *160*
 - port configuration, *159*
- OS DOWNLOAD
 - state, *31*
- OUT_ERRORS
 - device DDT, *211*
- OUT_PACKETS
 - device DDT, *211*

P

- panel
 - CPU, front, *42*
- password
 - for Unity Pro application, *103*
- performance web page
 - CPU, *328*
- physical description
 - CPU, *40, 43*
- ping, *160*
- port
 - Ethernet, *55*
- port function
 - device DDT, *211*
- port statistics web page
 - CPU, *329*
- power
 - cycle, *364*
- power consumption, *35*
- project
 - password, *103*

Q

- QoS, *129*
 - tab, *113*
- QoS object, *175*
- QoS web page
 - CPU, *334*
- Quantum RIO drops in M580
 - MBP_MSTR explicit message, *256*

R

- read data, *270*
- read/write data, *275*
- real-time clock, *36*
- redundancy web page
 - CPU, *338*
- reset module, *275*
- restart
 - warm, *369*
- restore, *108*
- RIO drops, Quantum
 - MBP_MSTR explicit message, *256*

- RIO scanner service
 - RSTP, *121*
 - selecting CPU, *23*
- RSTP
 - device DDT, *211*
 - DIO scanner service, *121*
 - EIO scanner service, *121*
 - RIO scanner service, *121*
 - tab, *113*
- RSTP diagnostics, *144*
- RSTP diagnostics object, *196*
- ruggedized, *65*
- RUN
 - state, *31*

S

- scanner service
 - RSTP, *121*
- SD memory card, *361*
 - FTP, *59*
- security
 - access control, *115*
 - authorized address, *115*
 - DHCP/BOOTP, *115*
 - EIP, *115*
 - enforce in Unity Pro, *115*
 - ETH_PORT_CTRL, *373*
 - FTP, *115*
 - HTTP, *115*
 - memory protect, *103*
 - password, *103*
 - SNMP, *115*
- Security
 - tab, *113*
- security
 - TFTP, *115*
 - unlock in Unity Pro, *115*
- service port
 - CPU, *130*
 - tab, *113*
- SERVICE_STATUS
 - device DDT, *211*
- SERVICE_STATUS2
 - device DDT, *211*

SNMP

- security, 115
- tab, 113

standards, 67

start

- cold, 366
- warm, 369

state

- AUTOTEST, 31
- CPU, 31
- ERROR, 31
- HALT, 31
- IDLE, 31
- NOCONF, 31
- OS DOWNLOAD, 31
- RUN, 31
- STOP, 31
- WAIT, 31

State RAM

- LL984, 106
- Quantum ERIO drops, 106

status summary web page

- CPU, 326, 346

STB NIC 2212

- configuring I/O items, 290

STOP

- state, 31

summary

- configuration, 317
- connections, 317

supply voltage, 70

switch, 128

Switch

- tab, 113

synchronous execution

- ETH_PORT_CTRL, 373

syslog server

- logging, 154

system error, 95

system states

- Hot Standby, 32

T

T_BMEP58_ECPU, 211

- device DDT, 211

T_BMEP58_ECPU_EXT, 211

- device DDT, 211

tab

- advanced settings, 113
- IPConfig, 113
- NTP, 113
- QoS, 113
- RSTP, 113
- Security, 113
- Service Port, 113
- SNMP, 113
- Switch, 113

task

- CPU, 359
- management, 356

TCP/IP interface object, 177

temperature, 70

tests

- conformity, 67

TFTP

- security, 115

U

Unity Pro

- configuration, 97

Unity Pro

- logging, 153

update

- firmware, 64

upgrade

- firmware, 64

USB

- cables, 53
- pin assignments, 53
- transparency, 53

V

voltage

- supply, 70

W

WAIT

- state, *31*

warm

- restart, *369*

- start, *369*

web page

- CPU alarm viewer, *339*

- CPU I/O scanner, *331*

- CPU messaging, *333*

- CPU NTP, *336*

- CPU performance, *328*

- CPU port statistics, *329*

- CPU QoS web page, *334*

- CPU redundancy, *338*

- CPU status summary, *326*

web pages, *344*

- rack viewer, *351*

write data, *270*