**User Manual**
GFK-1503F
Jan 2020

# PACSystems™ VersaMax

## PLC USER MANUAL

EMERSON

# Contents

# Warnings, Caution Notes as Used in this Publication

Warning

**Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.**
**In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.**

Caution

**Caution notices are used where equipment might be damaged if care is not taken.**

***Notes:*** *Notes merely call attention to information that is especially significant to understanding and operating the equipment.*

These instructions do not purport to cover all details or variations in equipment, nor to provide for every possible contingency to be met during installation, operation, and maintenance. The information is supplied for informational purposes only, and Emerson makes no warranty as to the accuracy of the information included herein. Changes, modifications, and/or improvements to equipment and specifications are made periodically and these changes may or may not be reflected herein. It is understood that Emerson may make changes, modifications, or improvements to the equipment referenced herein or to the document itself at any time. This document is intended for trained personnel familiar with the Emerson products referenced herein.

Emerson may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not provide any license whatsoever to any of these patents.

Emerson provides the following document and the information included therein as-is and without warranty of any kind, expressed or implied, including but not limited to any implied statutory warranty of merchantability or fitness for particular purpose.

# Chapter 1: Introduction

### Guide to the VersaMax Document Set

This manual contains general information about CPU operation and program content. It also provides detailed descriptions of specific programming requirements.

Chapter 1: is a general introduction to the VersaMax family of products.

**CPU Modules** are described in detail in Chapter 2:and Chapter 3:.

**Installation procedures** are described in Chapter 4:.

**PLC Configuration** is described in Chapter 5:. Configuration determines certain characteristics of module operation and also establishes the program references used by each module in the system.

**Ethernet Configuration** for CPU model IC200CPUE05 is described in Chapter 6:.

**CPU Operation** is described in Chapter 7:.

**Serial Communications** are described in Chapter 12:.

**Ethernet Communications** for CPU model IC200CPUE05 is described in Chapter 13:.

The rest of the manual describes many programming features.

- Elements of an Application Program: chapter 8
- Program Data: Chapter 9:
- Instruction Set Reference: Chapter 10:
- The Service Request Function: Chapter 11:
- The PID Function: Chapter 14:
- Instruction Timing: Appendix A:

### Other VersaMax Manuals

| | |
|---|---|
| VersaMax Modules, Power Supplies, and Carriers User's Manual (catalog number GFK-1504) | Describes the many VersaMax I/O and option modules, power supplies, and carriers. This manual also provides detailed system installation instructions. |
| VersaMax PLC Ethernet Station Manager's Manual (catalog number GFK-1876) | Describes the diagnostic interface to the Ethernet functions of CPU module IC200CPUE05. |
| VersaMax Ethernet Network Interface Unit User's Manual (catalog number GFK-1860) | Describes the installation and operation of the Ethernet Network Interface Unit module. |
| VersaMax Genius NIU User's Manual (catalog number GFK-1535) | Describes the installation and operation of the Genius NIU. |

| | |
|---|---|
| VersaMax DeviceNet Communications Modules User's Manual (catalog number GFK-1533) | Describes the installation and operation of the DeviceNet Network Interface Unit module and the DeviceNet Network Slave Module. |
| VersaMax Profibus Communications Modules User's Manual (catalog number GFK-1534) | Describes the installation and operation of the Profibus Network Interface Unit module and the Profibus Network Communications Module. |

# 1.1 The VersaMax™ Family of Products

The VersaMax family of products provides universally-distributed I/O that spans PLC and PC-based architectures. Designed for industrial and commercial automation, VersaMax I/O provides a common, flexible I/O structure for local and remote-control applications. The VersaMax PLC provides big-PLC power with a full range of I/O and option modules. VersaMax I/O Stations with Network Interface Modules make it possible to add the flexibility of VersaMax I/O to other types of networks. VersaMax meets UL, CUL, CE, Class1 Zone 2 and Class I Division 2 requirements.

As a scaleable automation solution, VersaMax I/O combines compactness and modularity for greater ease of use. The 70-mm depth and small footprint of VersaMax I/O enables easy, convenient mounting as well as space- saving benefits. Modules can accommodate up to 32 points of I/O each.

The compact, modular VersaMax products feature DIN-rail mounting with up to eight I/O and option modules per "rack" and up to 8 racks per VersaMax PLC or VersaMax I/O Station system. Expansion racks can be located up to

750 meters from the main VersaMax PLC or VersaMax I/O Station rack. Expansion racks can include any VersaMax I/O, option, or communications module.

VersaMax provides automatic addressing that can eliminate traditional configuration and the need for hand-held devices. Multiple field wiring termination options provide support for two, three, and four-wire devices.

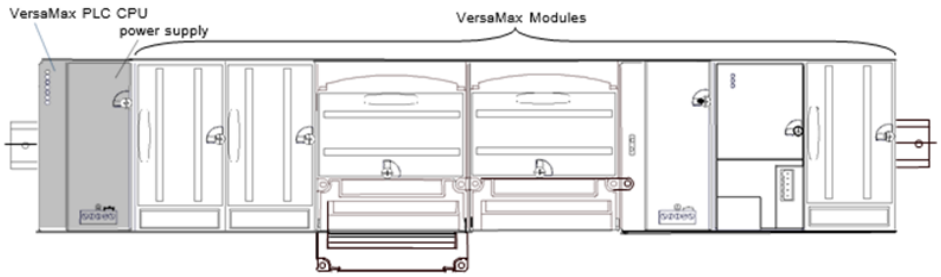
For faster equipment repair and shorter Mean-Time-To-Repair, the hot insertion feature enables addition and replacement of I/O modules while a machine or process is running and without affecting field wiring.

VersaMax I/O may be remotely located. Remote I/O interfaces for Genius, DeviceNet, Profibus, and Ethernet are available.

# 1.2 CPU Modules for VersaMax PLCs

A VersaMax PLC consists of a group of VersaMax modules with a VersaMax CPU and attached power supply in the first position.

**Figure 1**




All VersaMax CPUs provide powerful PLC functionality. They are designed to serve as the system controller for up to 64 modules with up to 2048 I/O points. Two serial ports provide RS-232 and RS-485 interfaces for SNP slave and RTU slave communications. CPU model IC200CPUE05 provides a built-in Ethernet port.

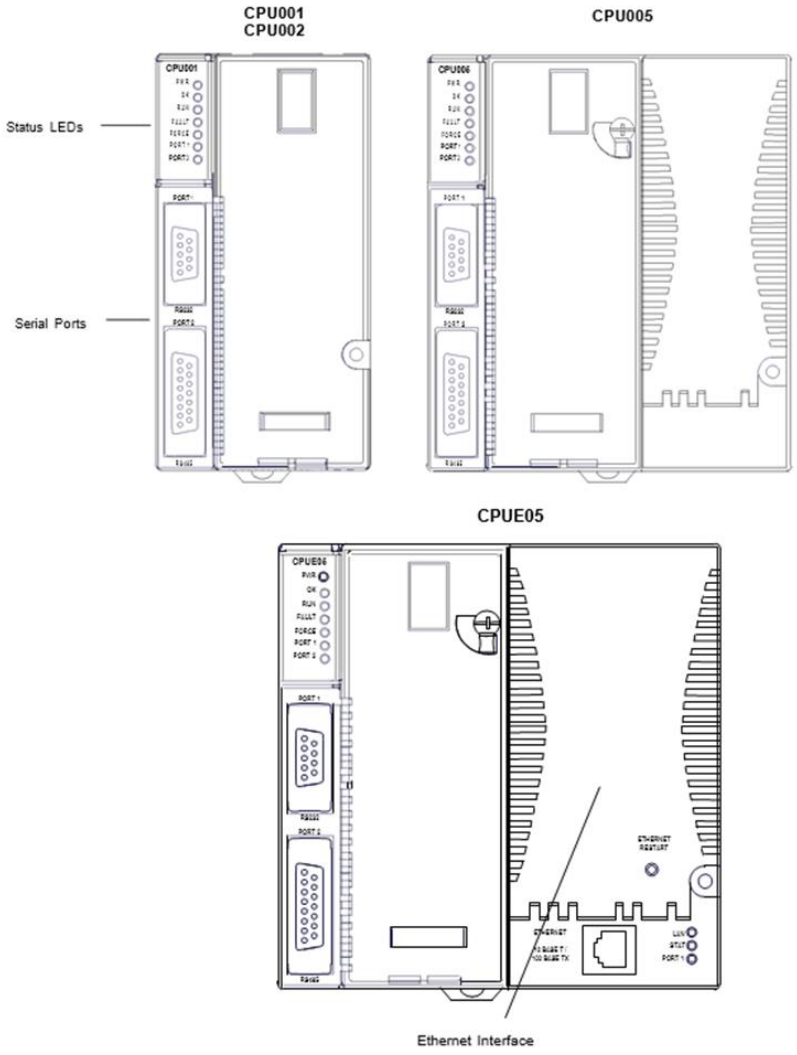
## 1.2.1 Basic CPU Features

- Programming in Ladder Diagram, Sequential Function Chart, and Instruction List
- Floating point (real) data functions
- Non-volatile flash memory for program storage
- Run/Stop switch
- Embedded RS-232 and RS-485 communications
- Compatible with EZ Program Store device

## 1.2.2 Available VersaMax CPUs

| | |
|---|---|
| CPU with Two Serial Ports, 34kB of Configurable Memory | IC200CPU00 |
| CPU with Two Serial Ports, 42kB of Configurable Memory | IC200CPU00 |
| CPU with Two Serial Ports, 128kB of Configurable Memory | IC200CPU00 |
| CPU with Two Serial Ports and Embedded Ethernet Interface, 128kB of Configurable Memory | IC200CPUE05 |

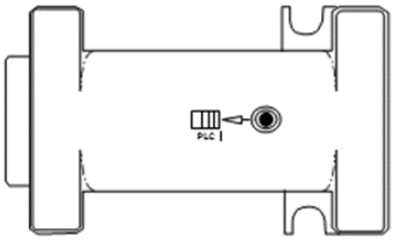
**Figure 2**




## 1.2.3 EZ Program Store

The EZ Program Store device (IC200ACC003) can be used to store and update the configuration, application program, and reference table data of a VersaMax PLC. A programmer and PLC CPU are used to initially write data to the device.

**Figure 3**

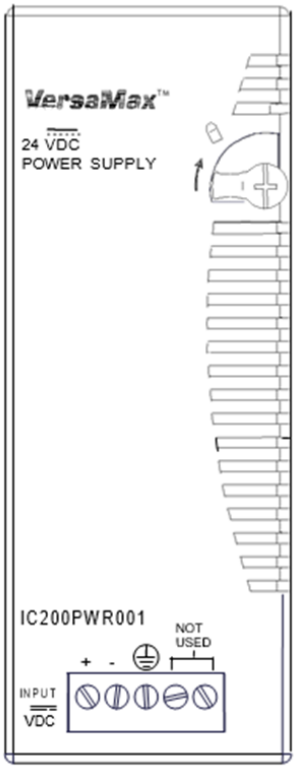


# 1.3 Power Supplies

An AC or DC Power Supply provides +5V and +3.3V power to the modules in the rack. Additional power supplies can be installed on special booster carriers if needed for systems where the number of modules creates the need for a booster. The AC or DC Power Supply on the CPU or NIU and the Power Supply that resides on the Booster Carrier must share the same external power source.

CPU models IC200CPU005 and IC200CPUE05 require the use of an "expanded" 3.3V power supply. See the table below.

**Figure 4**




## 1.3.1 Available Power Supplies and Carrier

The following VersaMax power supplies and carrier are available:

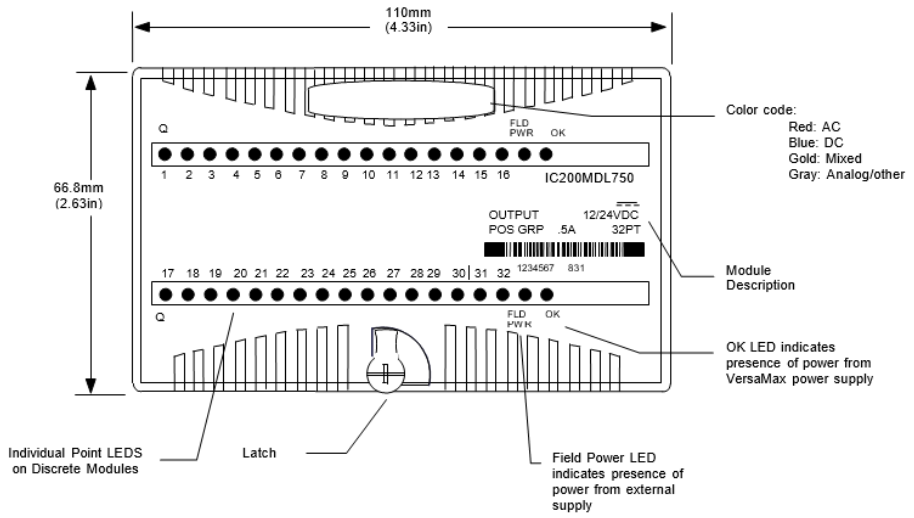
| | |
|---|---|
| 24VDC Power Supply | IC200PWR001 |
| 24VDC Expanded 3.3V Power Supply | IC200PWR002 |
| 120/240VAC Power Supply | IC200PWR101 |
| 120/240VAC Expanded 3.3V Power Supply | IC200PWR102 |
| 12VDC Power Supply | IC200PWR201 |
| 12VDC Expanded 3.3V Power Supply | IC200PWR202 |
| Power Supply Booster Carrier | IC200PWB001 |

Power supplies are described in the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504).

# 1.4 I/O Modules

VersaMax IO and option modules are approximately 110mm (4.33in) by 66.8mm (2.63in) in size. Modules can be mounted either horizontally or vertically on several types of available I/O Carriers. Modules are 50mm (1.956 in) in depth, not including the height of the carrier or the mating connectors.

**Figure 5**




VersaMax I/O modules are described in the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504).

## 1.4.1 Available I/O Modules

The following types of VersaMax I/O Modules are available:

| **Discrete Input Modules** | |
|---|---|
| Input 120VAC 8 Point Grouped Module | IC200MDL140 |
| Input 240VAC 8 Point Grouped Module | IC200MDL141 |
| Input 120VAC 8 Point Isolated Module | IC200MDL143 |
| Input 240VAC 4 Point Isolated Module | IC200MDL144 |
| Input 120VAC (2 Groups of 8) 16 Point Module | IC200MDL240 |
| Input 240VAC (2 Groups of 8) 16 Point Module | IC200MDL241 |
| Input 120VAC 16 Point Isolated Module | IC200MDL243 |
| Input 240VAC 8 Point Isolated Module | IC200MDL244 |
| Input 125VDC Positive/Negative Logic Grouped 8 Point Module | IC200MDL631 |
| Input 125VDC Positive/Negative Logic Grouped 16 Point Module | IC200MDL632 |
| Input 48VDC Positive/Negative Logic Grouped 16 Point Module | IC200MDL635 |
| Input 48VDC Positive/Negative Logic Grouped 32 Point Module | IC200MDL636 |
| Input 24VDC Positive/Negative Logic (2 Groups of 8) 16 Point Module | IC200MDL640 |
| Input 5/12VDC (TTL) Positive/Negative Logic 16 Point Module | IC200MDL643 |
| Input 5/12VDC (TTL) Positive/Negative Logic Grouped 32 Point Module | IC200MDL644 |

| | |
|---|---|
| Input 24VDC Positive/Negative Logic (4 Groups of 8) 32 Point Module | IC200MDL650 |
| **Discrete Output Modules** | |
| Output 120VAC 0.5A per Point Isolated 8 Point Module | IC200MDL329 |
| Output 120VAC 0.5A per Point Isolated 16 Point Module | IC200MDL330 |
| Output 120VAC 2.0A per Point Isolated 8 Point Module | IC200MDL331 |
| Output 24VDC Positive Logic 2.0A per Point (1 Group of 8) w/ESCP 8 Point Module, | IC200MDL730 |
| Output 12/24VDC Positive Logic 0.5A per Point (1 Group of 16) 16 Point Module | IC200MDL740 |
| Output 24VDC Positive Logic 0.5A per Point (1 Group of 16) w/ESCP 16 Point Module | IC200MDL741 |
| Output 24VDC Positive Logic 0.5A per Point (2 Groups of 16) w/ESCP 32 Point Module | IC200MDL742 |
| Output 5/12/24VDC Negative Logic 0.5A per Point (1 Group of 16) 16 Point Module | IC200MDL743 |
| Output 5/12/24VDC Negative Logic 0.5A per Point (2 Groups of 16) 32 Point Module | IC200MDL744 |
| Output 12/24VDC Positive Logic 0.5A per Point (2 Groups of 16) 32 Point Module | IC200MDL750 |
| Output Relay 2.0A per Point Isolated Form A 8 Point Module | IC200MDL930 |
| Output Relay 2.0A per Point Isolated Form A 16 Point Module | IC200MDL940 |
| **Discrete Mixed I/O Modules** | |
| Mixed 24VDC Positive Logic Input Grouped 20 Point / Output Relay 2.0A per Point Grouped 12 Point Module | IC200MDD840 |
| Mixed 24VDC Positive Logic Input 20 Point / Output 12 Point / (4) High Speed Counter, PWM, or Pulse Train Configurable Points | IC200MDD841 |
| Mixed 16 Point Grouped Input 24VDC Pos/Neg Logic / 16 Pt Grouped Output 24VDC Pos. Logic 0.5A w/ESCP | IC200MDD842 |
| Mixed 24VDC Positive Logic Input Grouped 10 Point / Output Relay 2.0A per Point 6 Point Module | IC200MDD843 |
| Mixed 24 VDC Pos/Neg Logic Input Grouped 16 Point / Output 12/24VDC Pos. Logic 0.5A 16 Point Module | IC200MDD844 |
| Mixed 16 Point Grouped Input 24VDC Pos/Neg Logic / 8 Pt Relay Output 2.0A per Pt Isolated Form A | IC200MDD845 |
| Mixed 120VAC Input 8 Point / Output Relay 2.0A per Point 8 Point Module | IC200MDD846 |
| Mixed 240VAC Input 8 Point / Output Relay 2.0A per Point 8 Point Module | IC200MDD847 |
| Mixed 120VAC Input 8 Point / Output 120VAC 0.5A per Point Isolated 8 Point Module | IC200MDD848 |
| Mixed 120VAC In Isolated 8 Point / Output Relay 2.0A Isolated 8 Point Module | IC200MDD849 |
| Mixed 240VAC In Isolated 4 Point / Output Relay 2.0A Isolated 8 Point Module | IC200MDD850 |
| **Analog Input Modules** | |
| Analog Input Module, 12 Bit Voltage/Current 4 Channels | IC200ALG230 |
| Analog Input Module, 16 Bit Voltage/Current, 1500VAC Isolation, 8 Channels | IC200ALG240 |
| Analog Input Module, 12 Bit Voltage/Current 8 Channels | IC200ALG260 |

| **Analog Input Modules** | |
|---|---|
| Analog Input Module, 15 Bit Differential Voltage 8 Channels | IC200ALG261 |
| Analog Input Module, 16 Bit Differential Current 8 Channels | IC200ALG262 |
| Analog Input Module, 15 Bit Voltage 15 Channels | IC200ALG263 |
| Analog Input Module, 15 Bit Current 15 Channels | IC200ALG264 |
| Analog Input Module, 16 Bit RTD, 4 Channels | IC200ALG620 |
| Analog Input Module, 16 Bit Thermocouple, 7 Channels | IC200ALG630 |
| **Analog Output Modules** | |
| Analog Output Module, 12 Bit Current, 4 Channels | IC200ALG320 |
| Analog Output Module, 12 Bit Voltage 4 Channels. 0 to +10VDC Range | IC200ALG321 |
| Analog Output Module, 12 Bit Voltage 4 Channels. -10 to +10VDC Range | IC200ALG322 |
| Analog Output Module, 13 Bit Voltage 8 Channels | IC200ALG325 |
| Analog Output Module, 12 Bit Current 8 Channels | IC200ALG326 |
| Analog Output Module, 13 Bit Voltage 12 Channels | IC200ALG327 |
| Analog Output Module, 12 Bit Current 12 Channels | IC200ALG328 |
| Analog Output Module, 16 Bit Voltage/Current, 1500VAC Isolation, 4 Channels | IC200ALG331 |
| **Analog Mixed I/O Modules** | |
| Analog Mixed Module, Input Current 4 Channels, Output Current 2 Channels | IC200ALG430 |
| Analog Mixed Module, 0 to +10VDC Input 4 Channels, Output 0 to +10VDC 2 Channels | IC200ALG431 |
| Analog Mixed Module, 12 Bit -10 to +10VDC, Input 4 Channels / Output -10 to +10VDC 2 Channels | IC200ALG432 |

## 1.5 Carriers

Carriers provide mounting, backplane communications, and field wiring connections for all types of VersaMax modules. I/O modules can be installed on carriers or removed without disturbing field wiring.

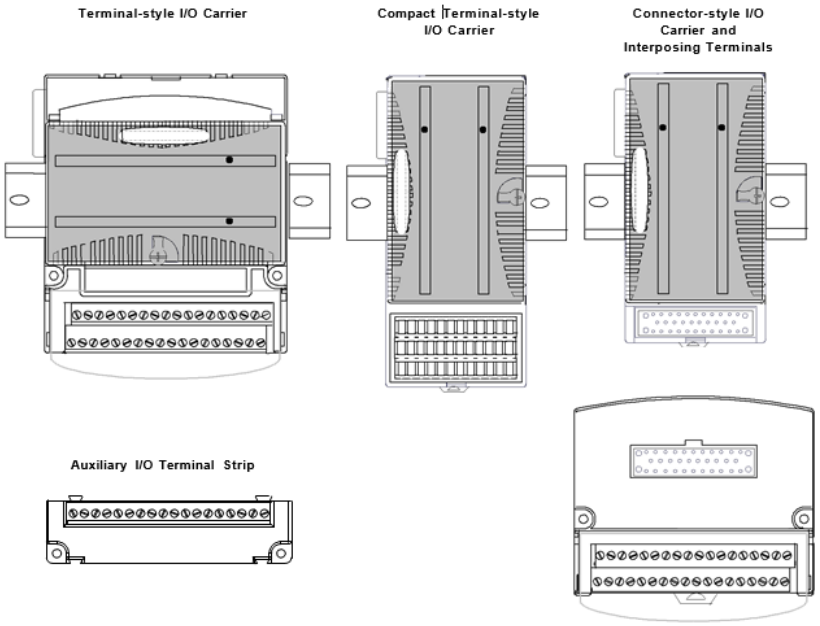
There are three basic I/O Carrier types:

- Terminal-style I/O carriers. Modules mount parallel to the DIN rail.
- Compact Terminal-style I/O Carriers. Modules mount perpendicular to the DIN rail.
- Connector-style I/O Carriers. Modules mount perpendicular to the DIN rail. These carriers are normally used with Interposing I/O Terminals as illustrated below.

See the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504) for information about VersaMax I/O Carriers.

Terminal-style I/O carriers have 36 individual terminals for direct connection of field wiring. Auxiliary I/O Terminal Strips are available for applications requiring additional wiring terminals.

**Figure 6**




## 1.5.1 Available Carriers and Terminal Strips

The following types of Carriers, terminals, and cables are available:

| | |
|---|---|
| **Terminal-Style I/O Carriers** | |
| Barrier-Style Terminal I/O Carrier | IC200CHS001 |
| Box-Style Terminal I/O Carrier | IC200CHS002 |
| Spring-Style Terminal I/O Carrier | IC200CHS005 |
| **Compact Terminal-Style I/O Carriers** | |
| Compact Box-Style I/O Carrier | IC200CHS022 |
| Compact Spring-Style I/O Carrier | IC200CHS025 |
| **Connector-Style I/O Carrier** | |
| Connector-Style I/O Carrier | IC200CHS003 |
| **Interposing Terminals for use with Connector-Style Carrier** | |
| Barrier-Style Interposing I/O Terminals | IC200CHS011 |
| Box-Style Interposing I/O Terminals | IC200CHS012 |
| Thermocouple-Style Interposing I/O Terminals | IC200CHS014 |
| Spring-Style Interposing I/O Terminals | IC200CHS015 |
| Disconnect-Style Interposing I/O Terminals, Main Base<br>Disconnect-Style Interposing I/O Terminals, Expansion Base | IC200CHS101<br>IC200CHS102 |
| Relay-Style Interposing I/O Terminals, Main Base Relay-Style Interposing I/O Terminals, Expansion Base | IC200CHS111<br>IC200CHS112 |
| Fuse-Style Interposing I/O Terminals, Main Base Fuse-Style Interposing I/O Terminals, Expansion Base | IC200CHS121<br>IC200CHS122 |
| **Cables for use with Connector-Style I/O Carriers** | |

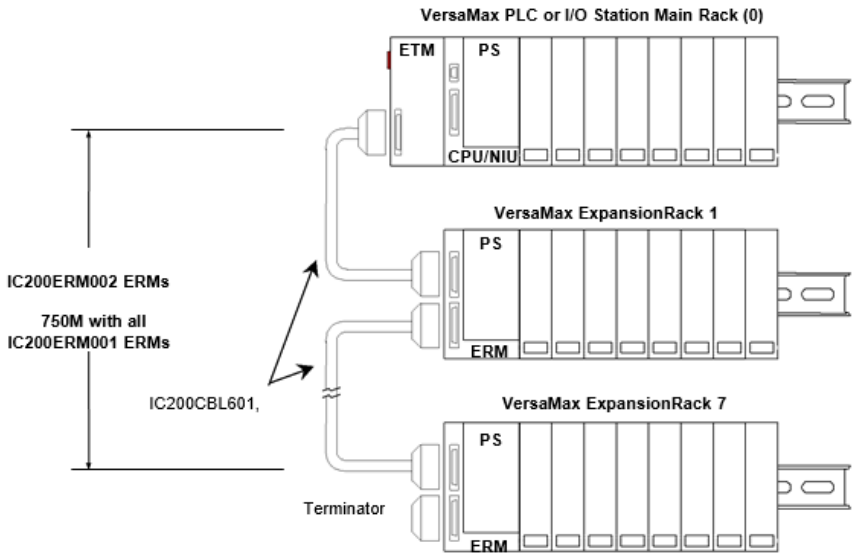
| | |
|---|---|
| 2 connectors, 0.5m, no shield | IC200CBL105 |
| 2 connectors, 1.0m, no shield | IC200CBL110 |
| 2 connectors, 2.0m, no shield | IC200CBL120 |
| 1 connector, 3.0m, no shield | IC200CBL230 |
| **Auxiliary I/O Terminal Strips for use with Terminal-style I/O Carriers and Interposing Terminals** | |
| Barrier-Style Auxiliary I/O Terminal Strip | IC200TBM001 |
| Box-Style Auxiliary I/O Terminal Strip | IC200TBM002 |
| Spring-Style Auxiliary I/O Terminal Strip | IC200TBM005 |
| **Other Carriers** | |
| Communications Carrier | IC200CHS006 |
| Power Supply Booster Carrier | IC200PWB001 |

# 1.6 Expansion Modules

There are two basic types of VersaMax I/O expansion systems, Multi-Rack and Single-ended:

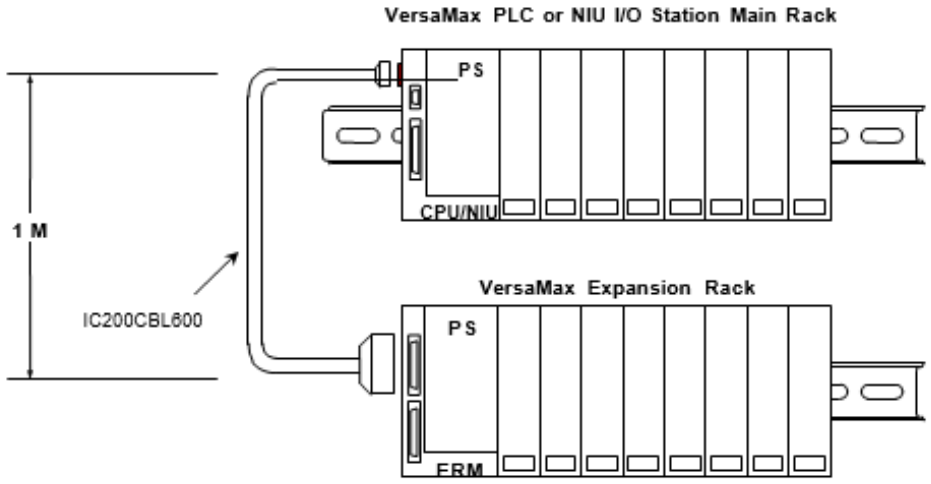
- **Multi-Rack:** A VersaMax PLC or NIU I/O Station with an Expansion Transmitter Module (IC200ETM001) and one to seven expansion "racks", each with an Expansion Receiver Module (IC200ERM001 or IC200ERM002). If all the Expansion Receivers are the Isolated type (IC200ERM001), the maximum overall cable length is 750 meters. If the expansion bus includes any non-isolated Expansion Receivers (IC200ERM002), the maximum overall cable length is 15 meters.

**Figure 7**




- **Single-ended**: A CPU or NIU I/O Station connected directly to one expansion rack with non-isolated Expansion Receiver Module (IC200ERM002). Maximum cable length is 1 meter.

**Figure 8**




## 1.6.1 VersaMax Modules for Expansion Racks

All types of VersaMax I/O and communications modules can be used in expansion racks. Some VersaMax analog modules require specific module revisions as listed below:

| Module | Module Revision |
|---|---|
| IC200ALG320 | B or later |
| IC200ALG321 | B or later |
| IC200ALG322 | B or later |
| IC200ALG430 | C or later |
| IC200ALG431 | C or later |
| IC200ALG432 | B or later |

## 1.6.2 Available Expansion Modules

The following Expansion Modules and related products are available:

| **Expansion Modules** | |
|---|---|
| Expansion Transmitter Module | IC200ETM001 |
| Expansion Receiver Module, Isolated | IC200ERM001 |
| Expansion Receiver Module, Non-isolated | IC200ERM002 |
| **Cables** | |
| Expansion Cable, Shielded, 1 meter | IC200CBL601 |
| Expansion Cable, Shielded, 2 meters | IC200CBL602 |
| Expansion Cable, Shielded, 15 meters | IC200CBL615 |
| Firmware Update Cable | IC200CBL002 |
| Terminator Plug (included with ETM) | IC200ACC201 |
| Connector Kit | IC200ACC302 |

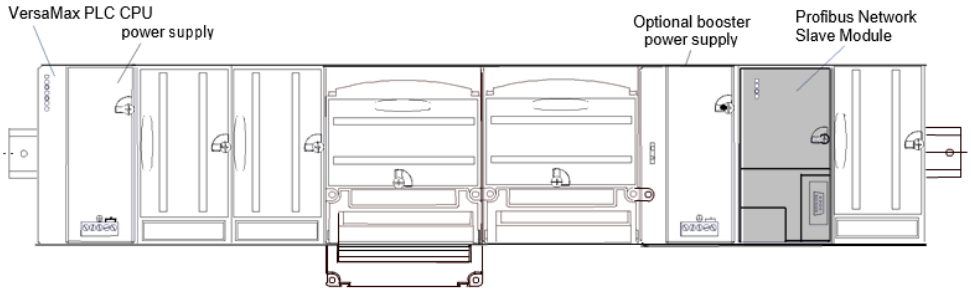
See the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504) for information about VersaMax Expansion modules.

# 1.7 Communications Modules

Communications modules provide additional flexibility for VersaMax systems.

These communications modules install on a VersaMax Communications Carrier. Power for the communications module comes from the main system power supply or from a booster supply as shown below.

**Figure 9**




## 1.7.1 Available VersaMax PLC Communications Modules

The following VersaMax PLC communications modules are available:

| Communications Modules | |
|---|---|
| Profibus-DP Network Slave Module | IC200BEM002 |
| DeviceNet Network Control Module | IC200BEM103 |
| Asi  Network Master Module | IC200BEM104 |
| Serial Communications Module | IC200CMM020 |
| Communications Carrier | IC200CHS006 |

For information about the Communications Carrier, please see the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504).

## 1.7.2 Profibus-DP Network Slave Module

The Profibus-DP Network Slave Module (IC200BEM002) is a communications module that exchanges PLC reference table data on the Profibus network. The VersaMax PLC CPU can read and write this data as though it were conventional bit- and word-type I/O data.

Multiple Profibus-DP Network Slave Modules may be used in the same VersaMax PLC. Each one can read up to 244 bytes of data from the network and send up to 244 bytes of output data. The total amount of combined inputs and outputs is 384 bytes.

For information about the Profibus-DP Network Slave Module, refer to the VersaMax System Profibus Network Modules User's Manual (GFK-1534, revision A or later).

## 1.7.3 DeviceNet Network Control Module

The DeviceNet Network Control Module (IC200BEM103) is a communications module that can be configured to operate as a master, as a slave, or as both simultaneously. It can exchange up to 512 bytes of input data and 512 bytes of output data with other devices on the DeviceNet network. The VersaMax PLC CPU can read and write this data as though it were conventional bit- and word-type I/O data.

The Network Control Module operates as a Group 2 Only Client (master) and can communicate only with Group 2 Slave devices. It can also operate as a Group 2 Only or a UCMM-capable Server (slave), or as a master and slave simultaneously.

For information about the DeviceNet Network Control Module, refer to the VersaMax System DeviceNet Network Communications User's Manual (GFK-1533).

## 1.7.4 Asi Network Master Module

The VersaMax AS-Interface Network Master (IC200BEM104) conforms to the AS-Interface Specification for the master AS-Interface protocol. It can be used to connect a VersaMax PLC or I/O station NIU to an Actuator-Sensor network.

The AS-Interface module supports communications with up to 31 slave devices, exchanging to exchange up to 4 bits of input data and 4 bits of output data per slave address on the Actuator-Sensor network.

For information about the AS-Interface Network Master Module, refer to the VersaMax System ASI Network Communications User's Manual (GFK-1697).

## 1.7.5 Serial Communications Module

The VersaMax Serial Communications Module, IC200CMM020, operates as a Modbus RTU Master in a VersaMax I/O Station. The Serial Communications module receives commands from a remote host such as an RX7i PLC.

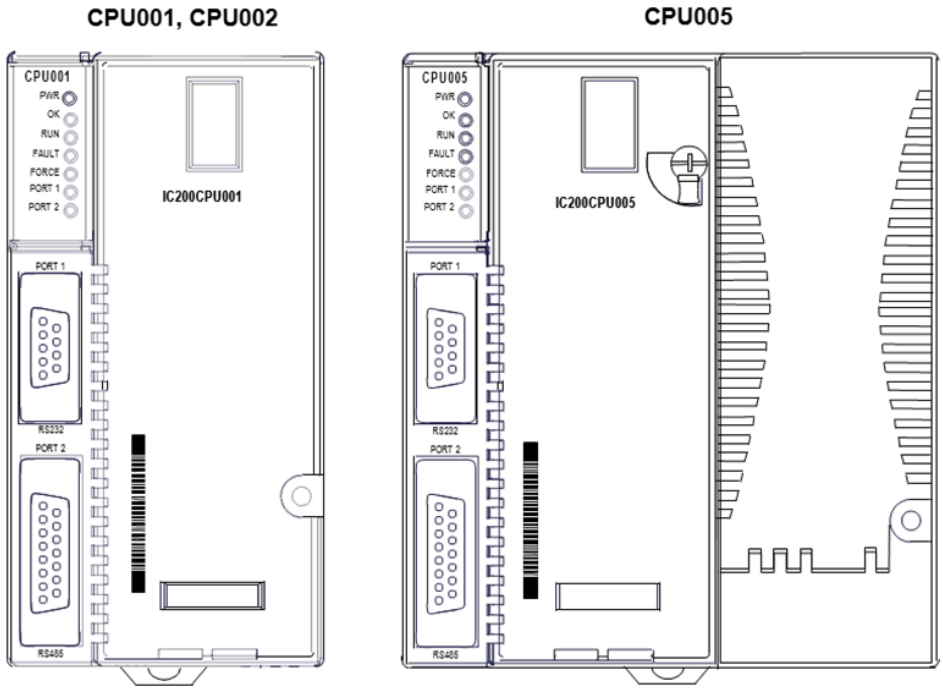
# Chapter 2: CPU Module Datasheets: CPU001, CPU002, CPU005

This chapter describes the appearance, features, and functionality of the following VersaMax PLC CPU modules:

- IC200CPU001 CPU with 34kB Configurable Memory
- IC200CPU002 CPU with 42kB Configurable Memory
- IC200CPU005 CPU with 128kB Configurable Memory

VersaMax™ PLC CPUs IC200CPU001, CPU002, and CPU005 provide powerful PLC functionality in a small, versatile system. They are designed to serve as the system controller for up to 64 modules with up to 2048 I/O points. Two serial ports provide RS-232 and RS-485 interfaces for SNP slave and RTU slave communications.

**Figure 10**



# 2.1 Features

- Non-volatile flash memory for program storage.
- Programming in Ladder Diagram, Sequential Function Chart, and Instruction List.
- Battery backup for program, data, and time of day clock.
- Run/Stop switch.
- Floating point (real) data functions.
- Embedded RS-232 and RS-485 communications.
- 70mm height when mounted on DIN rail with power supply.
- Compatible with EZ Program Store device.

# 2.2 Module Specifications

| | | | |
|---|---|---|---|
| Size | CPU001/002: 2.63" (66.8mm) x 5.04" (128mm) CPU005: 4.20" (106.7mm) x 5.04" (128mm) | | |
| Program storage | System flash, battery-backed RAM | | |
| Battery Backup for program, data, and time-of-day clock | Super capacitor provides power to memory for 1 hour.<br>Over 1 hour, backup battery protects memory contents up to 6 months.<br>Backup battery has shelf life of 5 years when not in use. | | |
| Backplane current consumption: IC200CPU001, IC200CPU002 | no serial port converter or EZ Program Store device | 5V output: 40mA | 3.3V output: 100mA |
| | with serial port converter or EZ Program Store device | 5V output: 140mA | |
| Backplane current consumption: IC200CPU005 | no serial port converter or EZ Program Store device | 5V output: 35mA | 3.3V output: 300mA* |
| | with serial port converter or EZ Program Store device | 5V output: 135mA | |
| Floating point | yes | | |
| Embedded communications | RS-232, RS-485 | | |
| Boolean execution speed | CPU001, CPU002: 1.8ms/K (typical)<br>CPU005: 0.5ms/K (typical) | | |
| Realtime clock accuracy (for timer functions) | 100ppm (0.01%) or +/- 9sec/day | | |
| Time of day clock accuracy | 23ppm (0.0023%) or +/- 2sec/day @ 30C.<br>100 ppm (0.01%) or +/- 9sec/day @ full temperature range | | |

* CPU005 requires a power supply with expanded 3.3V.

# 2.3 VersaMax General Product Specifications

VersaMax products should be installed and used in conformance with product-specific guidelines as well as the following specifications:

| | | |
|---|---|---|
| **Environmental** | | |
| Vibration | IEC68-2-6 | 1G @57-150Hz, 0.012in p--p @10-57Hz |
| Shock | IEC68-2-27 | 15G, 11ms |
| Operating Temp. | | 0 deg C to +60 deg C ambient |
| Storage Temp. | | -40 deg C to +85 deg C |
| Humidity | | 5% to 95%, noncondensing |
| Enclosure Protection | IEC529 | Steel cabinet per IP54:<br>protection from dust & splashing water |
| **EMC Emission** | | |
| Radiated, Conducted | CISPR 11/EN 55011 | Industrial Scientific & Medical Equipment (Group 1, Class A) |
| | CISPR 22/EN 55022 | Information Technology Equipment (Class A) |
| | FCC 47 CFR 15 | referred to as FCC part 15, Radio Devices (Class A) |
| **EMC Immunity** | | |
| Electrostatic Discharge | EN 61000-4-2 | 8KV Air, 4KV Contact |
| RF Susceptibility | EN 61000-4-3 | $10V_{rms}$ /m, 80Mhz to 1000Mhz, 80% AM |
| | ENV 50140/ENV 50204 | $10V_{rms}$/m, 900MHz +/-5MHZ<br>100%AM with 200Hz square wave |
| Fast Transient Burst | EN 61000-4-4 | 2KV: power supplies, 1KV: I/O, communication |
| Surge Withstand | ANSI/IEEE C37.90a | Damped Oscillatory Wave: 2.5KV power supplies, I/O [12V-240V]; 1KV communication |
| | IEC255-4 | Damped Oscillatory Wave: Class II, power supplies, I/O [12V-240V] |
| | EN 61000-4-5 | 2 kV cm(P/S); 1 kV cm (I/O and communication modules) |
| Conducted RF | EN 61000-4-6 | $10V_{rms}$, 0.15 to 80Mhz, 80%AM |
| **Isolation** | | |
| Dielectric Withstand | UL508, UL840, IEC664 | 1.5KV |

| **Power Supply** | | |
|---|---|---|
| Input Dips, Variations | EN 61000-4-11 | During Operation: Dips to 30% and 100%, Variation for AC +/-10%, Variation for DC +/-20% |

# 2.4 Serial Ports

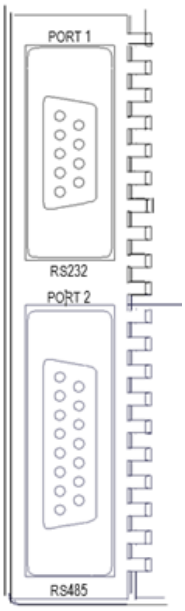
The two serial ports are software-configurable for SNP slave or RTU master or slave operation. 4-wire and 2-wire RTU are supported. If a port is being used for RTU, it automatically switches to SNP slave mode if necessary. Both ports default to SNP slave and both automatically revert to SNP slave when the CPU is in Stop mode, if configured for Serial I/O. Either port can be software-configured to set up communications between the CPU and various serial devices. An external device can obtain power from Port 2 if it requires 100mA or less at 5VDC.

**Port 1**: is an RS-232 port with a 9-pin female D-sub connector. The pinout of Port 1 allows a simple straight- through cable to connect with a standard AT-style RS-232 port. Maximum cable lengths from the CPU to the last device attached to the cable are:

| Baud Rate | Maximum Cable Length |
|---|---|
| 19.2 K and below | 15 meters (50 ft.) |
| 38.4K | 7.5 meters (25 ft.) |
| 57.6K | 5 meters (16 ft.) |
| 115.2K | 2.5 meters (8 ft.) |

**Port 2**: is an RS-485 port with a 15-pin female D-sub RS-232 adapter (IC690ACC901). Maximum cable lengths from the CPU to the last device attached to the cable is Port 2 1200 meters (4000 ft.).

**Figure 11**



The following table compares the functions of Port 1 and Port 2.

| | Port 1 | Port 2 |
|---|---|---|
| CPU Protocols (SNP slave, RTU master/slave, Serial I/O) | Defaults to SNP slave | Defaults to SNP slave |
| Firmware Upgrade | PLC in Stop/No I/O mode. | no |
| Smart module firmware upgrade | PLC in Stop/No I/O mode | PLC in Stop/No IO mode. |

### 2.4.1 Serial Port Baud Rates

| | CPU001, CPU002 | CPU005** |
|---|---|---|
| RTU protocol | 1200, 2400, 4800, 9600, 19.2K | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* |
| Serial I/O protocol | 1200, 2400, 4800, 9600, 19.2K | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* |
| SNP protocol | 4800, 9600, 19.2K, 38.4K* | 4800, 9600, 19.2K, 38.4K* |
| Firmware Upgrade via | 1200, 2400, 4800, 9600, 19.2K, 38.4K | 1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K |

* Only on one port at a time, not available on older CPU001 and CPU002 hardware revisions.

** Some versions of VersaPro software allow configuration of RTU and Serial I/O at 115.2K baud for CPU005. However, this baud rate is not supported by the CPU. If a configuration using this baud rate is stored to the PLC:

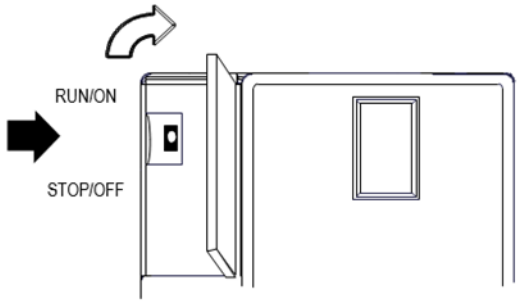
1. For RTU, an "Unsupported Feature in Configuration" fault is logged and the PLC transitions to Stop Faulted mode.
2. For Serial I/O, the same fault is logged when the transition to Run mode occurs. The PLC will immediately transition to Stop Faulted mode.

## 2.5 Mode Switch

The CPU module has a convenient switch that can be used to place the PLC in Stop or Run mode. The same switch can also be used to block accidental writing to CPU memory and forcing or overriding discrete data. Use of this feature is configurable.

The default configuration enables Run/Stop mode selection and disables memory protection.

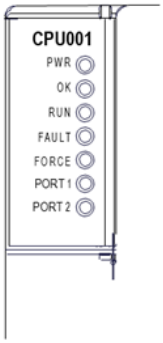
**Figure 12**




## 2.6 CPU LEDs

The seven CPU LEDs, visible through the module door, indicate the presence of power and show the operating mode and diagnostic status of the CPU. They also indicate the presence of faults, forces, and communications on the CPU's two ports.

**Figure 13**




| | |
|---|---|
| POWER | ON when the CPU is receiving 5V power from the power supply. Does not indicate the status of the 3.3V power output. |
| OK | ON indicates the CPU has passed its powerup diagnostics and is functioning properly. OFF indicates a CPU problem. Fast blinking indicates that the CPU is running its powerup diagnostics. Slow blinking indicates the CPU is configuring I/O modules. Simultaneous blinking of this LED and the green Run LED indicates that the CPU is in boot mode and is waiting for a firmware update through port 1. |
| RUN | Green when the CPU is in Run mode. Amber when the CPU is in Stop/IO Scan mode. If this LED is OFF but OK is ON, the CPU is in Stop/No IO Scan mode.<br><br>If this LED is flashing green and the Fault LED is ON, the module switch was moved from Stop to Run mode while a fatal fault existed. Toggling the switch will continue to Run mode. |

| | |
|---|---|
| FAULT | ON if the CPU is in Stop/Faulted mode because a fatal fault has occurred.<br><br>To turn off the Fault LED, clear both the I/O Fault Table and the PLC Fault Table. If this LED is blinking and the OK LED is OFF, a fatal fault was detected during PLC powerup diagnostics. Contact PLC Field Service. |
| FORCE | ON if an override is active on a bit reference. |
| PORT 1<br><br>PORT 2 | Blinking indicates activity on that port. |

# 2.7 Configurable Memory

CPU001 and CPU002 (release 2.0 or later) and CPU005 have configurable user memory. The configurable memory is the amount of memory required for the application program, hardware configuration, registers (%R), analog inputs (%AI), and analog outputs (%AQ).

The amount of memory allocated to the application program and hardware configuration are automatically determined by the actual program and configuration entered from the programmer. The rest of the configurable memory can be easily allocated to suit the application.

| | |
|---|---|
| Configurable memory | CPU001: 34K bytes maximum CPU002: 42K bytes maximum CPU005: 128K bytes maximum |
| Application program size (not configurable)<br>CPU001, for rel. 1.50 compatibility<br>CPU002, for rel. 1.50 compatibility | 128 bytes minimum<br>12K bytes<br>20K bytes |
| Hardware configuration size (not configurable) | 400 bytes minimum |
| Registers (%R)<br>CPU001/002, for rel. 1.50 compatibility | 256 bytes minimum<br>4,096 bytes |
| Analog Inputs (%AI) | 256 bytes minimum |
| Analog Outputs (%AQ) | 256 bytes minimum |

# Chapter 3: CPU Module Datasheet: CPUE05

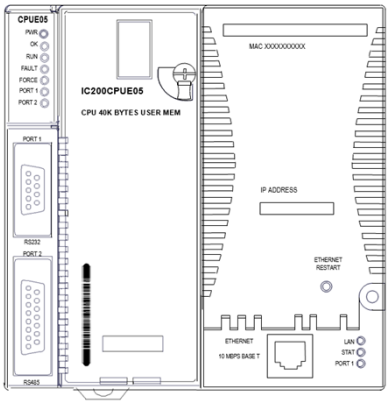
This chapter describes the appearance, features, and functionality of the following VersaMax PLC CPU module:

IC200CPUE05: CPU with Two Serial Ports, Embedded Ethernet Interface, and 128K Configurable Memory

CPU IC200CPUE05 shares the basic features of the other VersaMax PLC CPUs. It provides powerful PLC functionality in a small, versatile system. CPUE05 can serve as the system controller for up to 64 modules with up to 2048 I/O points. Two serial ports provide RS-232 and RS-485 interfaces for serial communications. CPUE05 also provides a built-in Ethernet Interface. The RS-232 serial port can be configured for Local Station manager operation to provide access to diagnostic information about the Ethernet interface. CPUE05 has 128kB of configurable memory.

In addition, CPUE05 is compatible with the EZ Program Store device, which can be used to write, read, update, and verify programs, configuration, and reference tables data without a programmer or programming software.

**Figure 14**




## 3.1 Features

- 128kB of configurable memory
- Programming in Ladder Diagram, Sequential Function Chart, and Instruction List
- Non-volatile flash memory for program storage
- Battery backup for program, data, and time of day clock
- Run/Stop switch
- Floating point (real) data functions
- Embedded RS-232 and RS-485 communications
- Embedded Ethernet interface
- 70mm height when mounted on DIN rail with power supply

# 3.2 Module Specifications

| Item | Description | | |
|---|---|---|---|
| Size | 4.95” (126mm) x 5.04” (128mm) | | |
| Program storage | System flash, battery-backed RAM | | |
| Battery backup for program, data, and time-of-day clock | Super capacitor provides power to memory for 1 hour<br>Over 1 hour, backup battery protects memory contents up to 6 months.<br>Backup battery has shelf life of 5 years when not in use. | | |
| Backplane current consumption: IC200CPUE05 | no serial port converter or EZ Program Store device | 5V output: 100mA | 3.3V output: 820mA* |
| | with serial port converter or EZ Program Store device | 5V output: 200mA | |
| Floating point | yes | | |
| Boolean execution speed | 0.5ms/K (typical) | | |
| Realtime clock accuracy (for timer functions) | 100ppm (0.01%) or +/- 9sec/day | | |
| Time of day clock accuracy | 23ppm (0.0023%) or +/- 2sec/day @ 30C.<br>100 ppm (0.01%) or +/- 9sec/day @ full temperature range | | |
| Embedded communications | RS-232, RS-485, Ethernet interface | | |
| Configurable memory | 128K bytes maximum | | |
| **Ethernet Interface Specifications** | | | |
| Number of SRTP server connections | 8 | | |
| Ethernet data rate | 10Mbps | | |
| Physical interface | 10BaseT RJ45 Shielded | | |
| WinLoader support | via CPU port | | |
| Number of Ethernet Global Data configuration-based exchanges | 32 | | |
| EGD Exchange limits | 100 data ranges and 1400 bytes of data per exchange; 1200 total data ranges across all exchanges. | | |
| Time Synchronization | NTP - client only (Supported IC200 CPUE05-HK or before) | | |
| Selective Consumption of EGD | yes | | |
| Load EGD configuration from PLC to programmer | yes | | |
| Remote Station Manager over UDP | yes | | |
| Local Station Manager (RS-232) | via CPU port | | |

| Item | Description |
|---|---|
| Configurable Advanced User Parameters | yes |
| * CPUE05 requires a power supply with expanded 3.3V. | |

# 3.3 VersaMax General Product Specifications

VersaMax products should be installed and used in conformance with product-specific guidelines as well as the following specifications

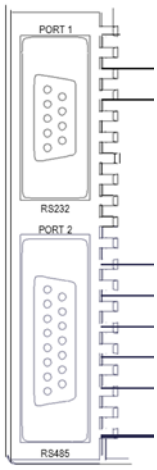
| | | |
|---|---|---|
| **Environmental** | | |
| Vibration | IEC60068-2-6 | 1G @57-150Hz, 0.012in p--p @10-57Hz |
| Shock | IEC60068-2-27 | 15G, 11ms |
| Operating Temp. | | 0 deg C to +60 deg C ambient |
| Storage Temp. | | -40 deg C to +85 deg C |
| Humidity | | 5% to 95%, noncondensing |
| Enclosure Protection | IEC60529 | Steel cabinet per IP54:<br>protection from dust & splashing water |
| **EMC Emission** | | |
| Radiated, Conducted | CISPR 11/EN 55011 | Industrial Scientific & Medical Equipment (Group 1, Class A) |
| | CISPR 22/EN 55022 | Information Technology Equipment (Class A) |
| | FCC 47 CFR 15 | referred to as FCC part 15, Radio Devices (Class A) |
| **EMC Immunity** | | |
| Electrostatic Discharge | EN 61000-4-2 | 8KV Air, 4KV Contact |
| RF Susceptibility | EN 61000-4-3 | 10Vrms /m, 80Mhz to 1000Mhz, 80% AM |
| | | |
| Fast Transient Burst | EN 61000-4-4 | 2KV: power supplies, 1KV: I/O, communication |
| Surge Withstand | ANSI/IEEE C37.90a<br>EN 61000-4-18 | Damped Oscillatory Wave: 2.5KV power supplies, I/O [12V-240V]; 1KV communication |
| | IEC60255-4 | Damped Oscillatory Wave: Class II, power supplies, I/O [12V-240V] |
| | EN 61000-4-5 | 2 kV cm(P/S); 1 kV cm (I/O and communication modules) |
| Conducted RF | EN 61000-4-6 | 10Vrms, 0.15 to 80Mhz, 80%AM |

| | | |
|---|---|---|
| **Isolation** | | |
| Dielectric Withstand | UL508, UL840, IEC664 | 1.5KV |
| **Power Supply** | | |
| Input Dips, Variations | EN 61000-4-11 | During Operation: Dips to 30% and 100%, Variation for AC +/-10%, Variation for DC +/-20% |

# 3.4 Serial Ports

The two serial ports are software-configurable for SNP slave or RTU master or slave operation. 4-wire and 2-wire RTU are supported. If a port is being used for RTU, it automatically switches to SNP slave mode if necessary. Port 1 can also be configured for Local Station Manager operation to provide access to diagnostic information about the Ethernet interface. Both ports default to SNP slave and both automatically revert to SNP slave when the CPU is in Stop mode, if configured for Serial I/O. Either port can be software-configured to set up communications between the CPU and various serial devices. An external device can obtain power from Port 2 if it requires 100mA or less at 5VDC.

**Figure 15**




**Port 1:** is an RS-232 port with a 9-pin female D-sub connector. The pinout of Port 1 allows a simple straight-through cable to connect with a standard AT-style RS-232 port.

Port 1 can be configured for either CPU serial communications (SNP, RTU, Serial I/O), or local Station Manager use. If Port 1 has been operation using the Restart pushbutton. Once forced, Port 1 remains available for station manager use until the PLC is power cycled, or the Restart pushbutton is pressed. If Port 1 is configured as a local Station Manager, it cannot be used for CPU serial communications or for firmware upgrades using Winloader.The Restart pushbutton will NOT toggle it to the CPU serial protocols.

**Port 2:** is an RS-485 port with a 15-pin female D-sub connector. This can be attached directly to an RS-485 to RS-232 adapter (IC690ACC901). Port 2 can be used for program, configuration, and table updates with the EZ Program Store module.

The following table compares the functions of Port 1 and Port 2.

| | Port 1 | Port 2 |
|---|---|---|
| CPU Protocols (SNP slave, RTU master/ slave, Serial I/O) | Defaults to SNP slave | Defaults to SNP slave |
| Local Station Manager | Yes (see above) | no |
| Firmware Upgrade | PLC in Stop/No I/O mode, Port 1 not disabled or in Local Station Manager mode. | no |
| Smart module firmware upgrade | PLC in Stop/No I/O mode, Port 1 configured for CPU protocol | PLC must be in Stop/No IO mode. |
| EZ Program Store device | No | Read, Write, Verify, and Update. PLC must be in Stop/No IO mode. |

## 3.4.1 Cable Lengths

Maximum cable lengths from the CPU to the last device attached to the cable are:

**Port 1 (RS-232):**

| Baud Rate | Maximum Cable Length |
|---|---|
| 19.2 K and below | 15 meters (50 ft.) |
| 38.4K | 7.5 meters (25 ft.) |
| 57.6K | 5 meters (16 ft.) |
| 115.2K | 2.5 meters (8 ft.) |

**Port 2 (RS-485):** 1200 meters (4000 ft.)

## 3.4.2 Serial Port Baud Rates

| | Port 1 | Port 2 |
|---|---|---|
| RTU protocol | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* |
| Serial I/O protocol | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* | 1200, 2400, 4800, 9600, 19.2K, 38.4K*, 57.6K* |
| SNP protocol | 4800, 9600, 19.2K, 38.4K* | 4800, 9600, 19.2K, 38.4K* |
| Local Station Manager (this is independent of serial protocol baud rate) | 1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K | na |
| Firmware Upgrade via WinLoader | 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K | na |

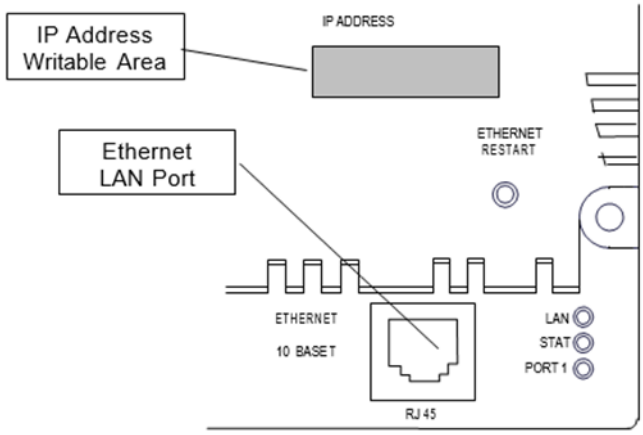
| | Port 1 | Port 2 |
|---|---|---|
| * Only on one port at a time.<br>Some versions of VersaPro software allow configuration of RTU and Serial I/O at 115.2K baud. However, this baud rate is not supported by the CPU. If a configuration using this baud rate is stored to the PLC:<br>1. For RTU, an "Unsupported Feature in Configuration" fault is logged and the PLC transitions to Stop Faulted mode.<br>2. For Serial I/O, the same fault is logged when the transition to Run mode occurs. The PLC will immediately transition to Stop Faulted mode. | | |

# 3.5 Ethernet LAN Port

The Ethernet LAN port supports SRTP Server and Ethernet Global Data. This port connects directly to a 10BaseT (twisted pair shielded) network without an external transceiver. The 10BaseT twisted pair shielded cables must meet applicable IEEE 802 standards. CPUE05 automatically selects either half-duplex or full- duplex operation, as sensed from the network connection.

A space is provided on the front of the CPUE05 module where the configured IP Address can be written.

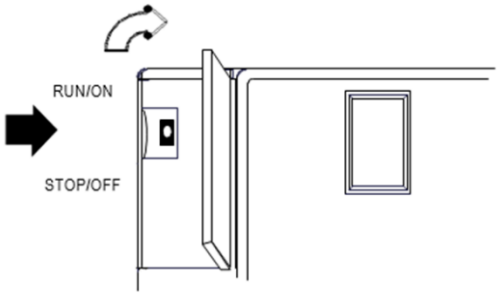
**Figure 16**




# 3.6 Mode Switch

The Mode switch is located behind the module door. It can be used to place the PLC in Stop or Run mode. It can also be used to block accidental writing to CPU memory and forcing or overriding discrete data. Use of this feature is configurable. The default configuration enables Run/Stop mode selection and disables memory protection.

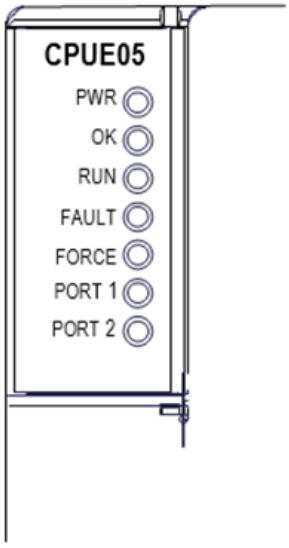
**Figure 17**




# 3.7 CPU LEDs

The seven CPU LEDs, visible through the module door, indicate the presence of power and show the operating mode and diagnostic status of the CPU. They also indicate the presence of faults, forces, and communications on the CPU's two ports indicate the status of the 3.3V power output.

**Figure 18**



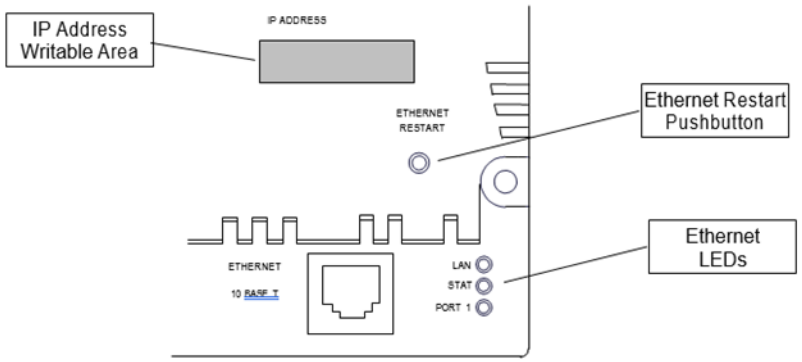

| | |
|---|---|
| OK | ON indicates the CPU has passed its powerup diagnostics and is functioning properly. OFF indicates a CPU problem. Fast blinking indicates that the CPU is running its powerup diagnostics. Slow blinking indicates the CPU is configuring I/O modules. Simultaneous blinking of this LED and the green Run LED indicates that the CPU is in boot mode and is waiting for a firmware update through port 1. |
| RUN | Green when the CPU is in Run mode. Amber when the CPU is in Stop/IO Scan mode. If this LED is OFF but OK is ON, the CPU is in Stop/No IO Scan mode. |

| | |
|---|---|
| | If this LED is flashing green and the Fault LED is ON, the module switch was moved from Stop to Run mode while a fatal fault existed. Toggling the switch will continue to Run mode. |
| FAULT | ON if the CPU is in Stop/Faulted mode because a fatal fault has occurred. To turn off the Fault LED, clear both the I/O Fault Table and the PLC Fault Table. If this LED is blinking and the OK LED is OFF a fatal fault was detected during PLC powerup diagnostics. Contact PLC Field Service. |
| FORCE | ON if an override is active on a bit reference. |
| PORT 1<br>PORT 2 | Blinking indicates activity on that port when controlled by the CPU. |

# 3.8 Ethernet Restart Pushbutton

The Ethernet Restart pushbutton is located on the right side of the module.

**Figure 19**




The Ethernet Restart pushbutton has two functions:

- When pressed for less than 5 seconds, it resets the Ethernet hardware, tests the Ethernet LEDs, and restarts the Ethernet firmware. This disrupts any Ethernet communications that are presently underway.
- When pressed for at least 5 seconds, it toggles the function of Port 1 between its configured operation and forced local Station Manager operation. Note that if Port 1 is available for Local Station Manager operation, Winloader cannot be used for a firmware upgrade.

# 3.9 Ethernet LEDs

Three LEDs indicate the status and activity of the Ethernet interface.

LAN indicates the status and activity of the Ethernet network connection. ON/flickering green indicates Ethernet interface is online.

**STAT** indicates the general status of the Ethernet interface. ON green indicates no "exception" detected. ON amber indicates an exception. Blinking amber indicates error code. Blinking green indicates waiting for configuration or waiting for IP address.

**PORT1** indicates when the Ethernet interface is controlling the RS-232 serial port. It also indicates when the Ethernet Restart pushbutton has been used to override configured RS-232 port usage for Local Station Manager operation. ON amber indicates Port 1 is available for Local Station Manager use (either by configuration or forced). OFF indicates PLC CPU is controlling Port 1. (Does not blink to indicate traffic).

The Ethernet LEDs turn ON briefly green, when a restart is performed in the Operational state by pressing and releasing the Restart pushbutton. This verifies that the Ethernet LEDs are operational. All three LEDs blink green in unison when a software load is in progress.

## 3.10 Configurable Memory

CPUE05 provides a total of 128K bytes of configurable user memory. This 64K of memory is use for the application program, hardware configuration, registers (%R), analog inputs (%AI), and analog outputs (%AQ). The amount of memory allocated to the application program and hardware configuration are automatically determined by the actual program and configuration entered from the programmer. The rest of the 64K bytes can be easily configured to suit the application.

| | |
|---|---|
| Configurable memory | 128K bytes maximum |
| Application program size (not configurable) | 128 bytes minimum |
| Hardware configuration size (not configurable) | 528 bytes minimum |
| Registers (%R) | 256 bytes minimum |
| Analog Inputs (%AI) | 256 bytes minimum |
| Analog Outputs (%AQ) | 256 bytes minimum |

# 3.11 Ethernet Interface Overview

CPUE05 has a built-in Ethernet interface that makes it possible to communicate on a 10BaseT network. Both half-duplex and full-duplex operation are supported. Using 10/100 hubs allows CPUE05 to communicate on a network containing 100Mb devices.-

## 3.11.1 SRTP Server

CPUE05 supports up to eight simultaneous SRTP Server connections for use by other devices on the Ethernet network, such as the PLC programmer, CIMPLICITY HMI, SRTP channels for Series 90 PLCs, and Host Communications Toolkit applications. No PLC programming is required for server operation.

## 3.11.2 Ethernet Global Data

CPUE05 supports up to 32 simultaneous Ethernet Global Data exchanges. Global Data exchanges are configured using the PLC programming software, then stored to the PLC. Both Produced and Consumed exchanges may be configured. CPUE05 supports up to 1200 variables across all Ethernet Global Data exchanges and supports selective consumption of Ethernet Global Data exchanges. See chapter 13 for information about Ethernet Global Data.

## 3.11.3 Station Manager Functionality

CPUE05 has built-in Station Manager functionality. This permits on-line diagnostic and supervisory access through either the Station Manager port or via the Ethernet network. Station Manager services include:

- An interactive set of commands for interrogating and controlling the station.
- Unrestricted access to observe internal statistics, an exception log, and configuration parameters.
- Password security for commands that change station parameters or operation.

Use of the Station Manager function requires a separate computer terminal or terminal emulator.

See GFK-1876 for information about Station Manager operation.

# Chapter 4: Installation

This chapter describes:

- Installing the CPU
- Installing the power supply
- Installing additional modules
- Activating or replacing the backup battery
- Serial port connections
- Installing expansion modules
- Ethernet connection for CPUE05
- CE Mark installation requirements

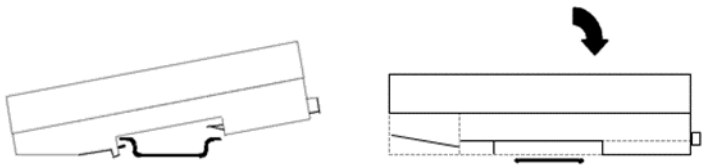
System installation instructions, which give guidelines for carrier, power supply, and module installation, as well as information about field wiring and grounding, are located in the VersaMax Modules, Power Supplies, and Carriers Manual, GFK-1504.

## 4.1 Mounting Instructions

All VersaMax™ modules and carriers in the same PLC "rack" must be installed on a single section of 7.5mm X 35mm DIN rail, 1mm thick. Steel DIN rail is recommended. The DIN rail must be electrically grounded to provide EMC protection. The rail must have a conductive (unpainted) corrosion-resistant finish. DIN rails compliant with DIN EN50022 are preferred. For vibration resistance, the DIN rail should be installed on a panel using screws spaced approximately 15.24cm (6 inches) apart.

The base snaps easily onto the DIN rail. No tools are required for mounting or grounding to the rail.

**Figure 20**



### 4.1.1 Removing the CPU from the DIN Rail

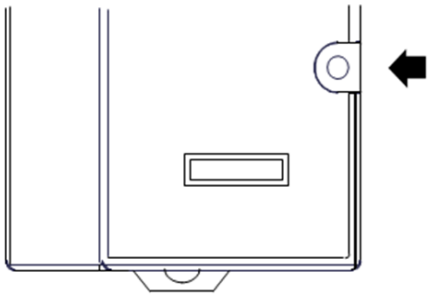
1. Turn off power to the power supply.
2. (If the CPU is attached to the panel with a screw) remove the power supply module. Remove the panel-mount screw.
3. Slide the CPU along the DIN rail away from the other modules until the connector disengages.

4. With a small flathead screwdriver, pull down on the DIN rail latch tab(s) on the bottom of the module and lift the module off the DIN rail.

## 4.1.2 Panel-Mounting

For maximum resistance to mechanical vibration and shock, the equipment must also be installed on a panel. Using the module as a template, mark the location of the module's panel-mount hole on the panel. Drill the hole in the panel. Install the module using an M3.5 (#6) screw in the panel-mount hole.

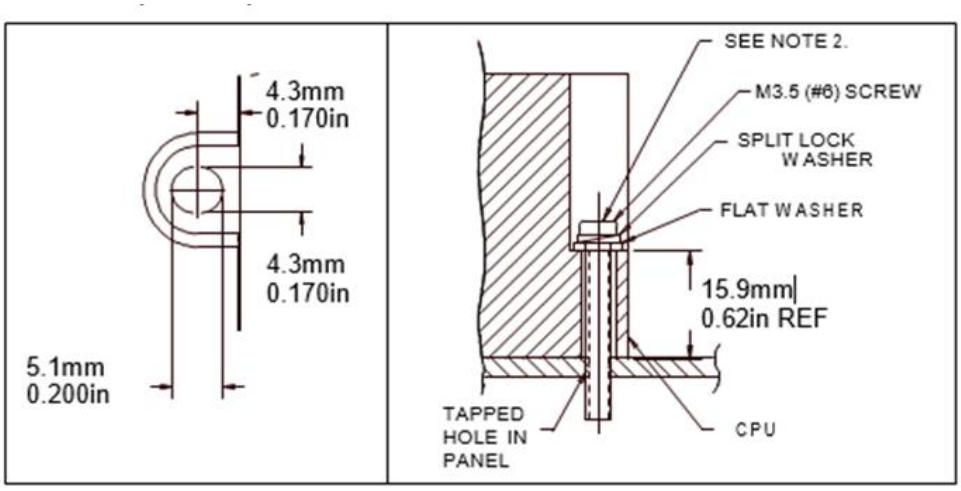
**Figure 21**



*Note:*

1. *Tolerances on all dimensions are +/- 0.13mm +/-0.005in) non- cumulative.*
2. *1.1 to 1.4Nm (10 to 12 in/lbs) of torque should be applied to M3.5 (#6-32) steel screw threaded into material containing internal threads and having a minimum thickness of 2.4mm (0.093in).*

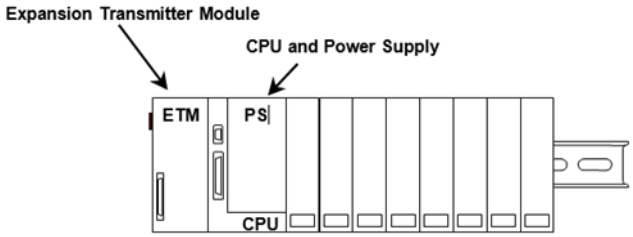
**Figure 22**



# 4.2 Installing an Expansion Transmitter Module

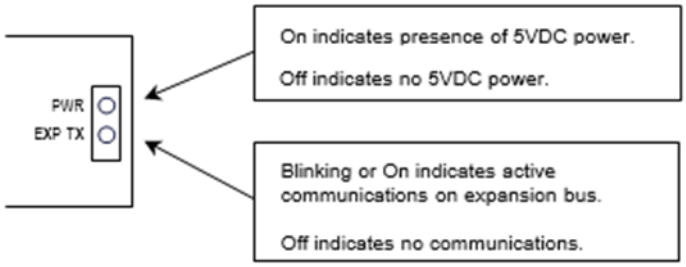
If the VersaMax PLC will have more than one expansion rack or one expansion rack that uses an Isolated Expansion Receiver Module (IC200ERM001) as its interface to the expansion bus, an Expansion Transmitter Module must be installed to the left of the CPU. The Expansion Transmitter Module must be installed on the same section of DIN rail as the rest of the modules in the main "rack" (rack 0).

**Figure 23 VersaMax PLC Main Rack (0)**




1. Make sure rack power is off.
2. Attach the Expansion Transmitter to DIN rail to the left of the CPU position.
3. Install the CPU. Connect the modules and press them together until the connectors are mated.
4. After completing any additional system installation steps, apply power and observe the module LEDs.

**Figure 24**




## 4.2.1 Removing an Expansion Transmitter Module

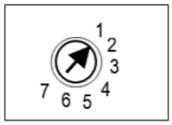
1. Make sure rack power is off.
2. Slide module on DIN rail away from the CPU in the main rack.
3. Using a small screwdriver, pull down on the tab on the bottom of the module and lift the module off the DIN rail.

# 4.3 Installing an Expansion Receiver Module

An Expansion Receiver Module (IC200ERM001 or 002) must be installed in the leftmost slot of each VersaMax expansion "rack".

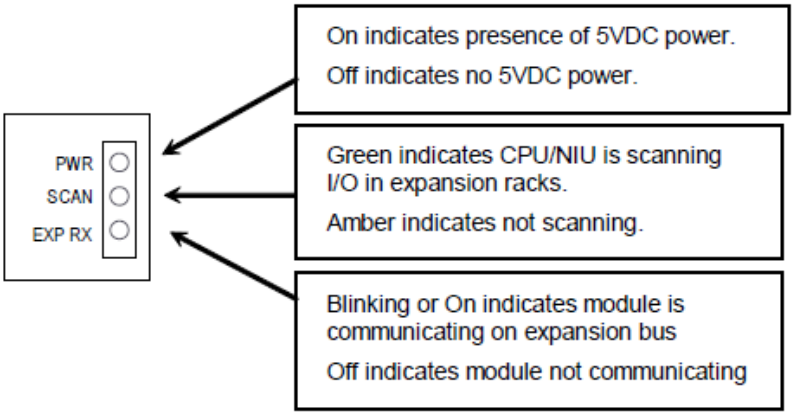
1. Insert the label inside the small access door at the upper left corner of the module.
2. Attach the module to the DIN rail at the left end of the expansion rack.
3. Select the expansion rack ID (1 to 7) using the rotary switch under the access door at upper left corner of the module. Each rack must be set to a different rack ID. With a single-ended cable (one expansion rack only), set the Rack ID to 1.

**Figure 25**




4. Install a VersaMax Power Supply module on top of the Expansion Receiver. See "Installing Power Supply Modules" in this chapter for details.
5. Attach the cables. If the system includes an Expansion Transmitter Module, attach the terminator plug to the EXP2 port on the last Expansion Receiver Module.
6. After completing any additional system installation steps, apply power and observe the module LEDs.

**Figure 26**




## 4.3.1 Removing an Expansion Receiver Module

1. Make sure rack power is off.
2. Uninstall the Power Supply module from the Expansion Receiver Module.
3. Slide the Expansion Receiver Module on DIN rail away from the other modules.
4. Using a small screwdriver, pull down on the tab on the bottom of the module and lift the module off the DIN rail.

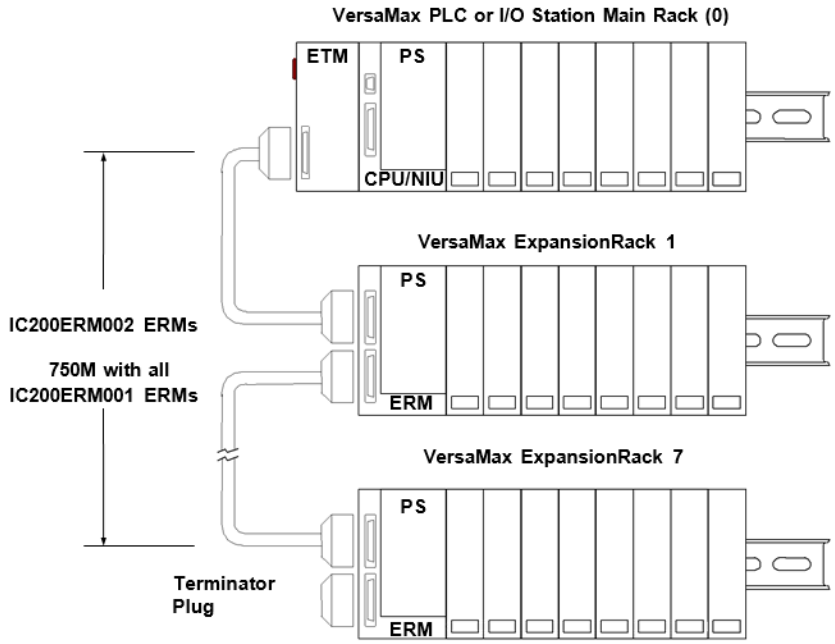
## 4.3.2 Expansion Rack Power Sources

Power for module operation comes from the Power Supply installed on the Expansion Receiver Module. If the expansion rack includes any Power Supply Booster Carrier and additional rack Power Supply, it must be tied to the same source as the Power Supply on the Expansion Receiver Module.

## 4.3.3 Connecting the Expansion Cable: RS-485 Differential

For a multiple-rack expansion system, connect the cable from the expansion port on the Expansion Transmitter to the Expansion Receivers as shown below. If all the Expansion Receivers are the Isolated type (IC200ERM001), the maximum overall cable length is 750 meters. If the expansion bus includes any non-isolated Expansion Receivers (IC200ERM002), the maximum overall cable length is 15 meters.

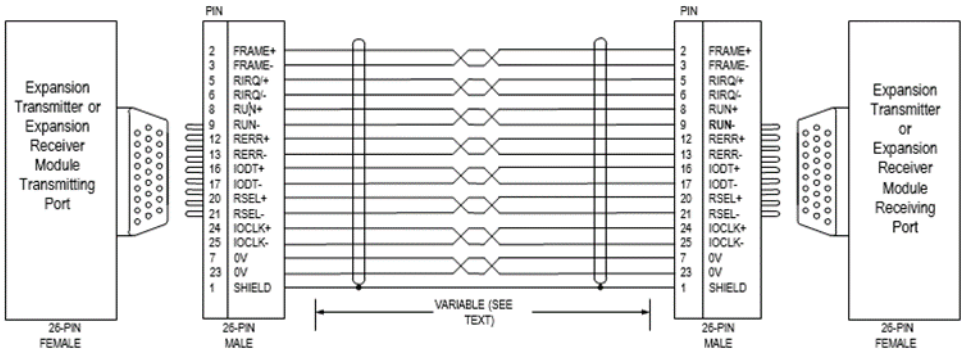
**Figure 27**




Install the Terminator Plug (supplied with the Expansion Transmitter module) into the lower port on the last Expansion Receiver. Spare Terminator Plugs can be purchased separately as part number IC200ACC201 (Qty 2).

## 4.3.4 RS-485 Differential Inter-Rack Connection (IC200CBL601, 602, 615)

**Figure 28**



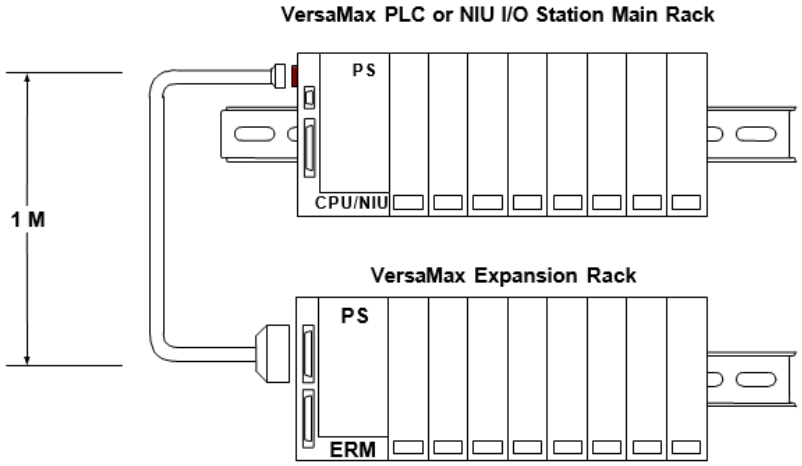

## 4.3.5 Building a Custom Expansion Cable

Custom expansion cables can be built using Connector Kit IC200ACC202, Crimper AMP 90800-1, and Belden 8138, Manhattan/CDT M2483, Alpha 3498C, or equivalent AWG #28 (0.089mm$^2$) cable.

## 4.3.6 Connecting the Expansion Cable: Single-ended

For a system with one non-isolated expansion rack (IC200ERM002) and NO Expansion Transmitter, connect the expansion cable from the serial port on the VersaMax CPU to the Expansion Receiver as shown below. The maximum cable length is one meter. Cables cannot be fabricated for this type of installation; cable IC200CBL600 must be ordered separately.

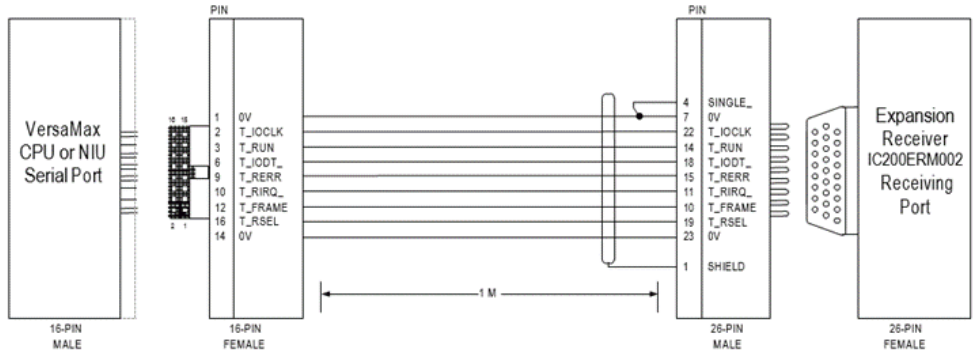
**Figure 29**




No Terminator Plug is needed in a single-ended installation; however, it will not impede system operation if installed.

### 4.3.7 Single-Ended Inter-Rack Connection (IC200CBL600)

**Figure 30**




### 4.3.8 Power Sources for Single-Ended Expansion Rack Systems

When operating the system in single-ended mode, power supplies for the main rack and expansion rack must use the same main power source. The main rack and expansion racks cannot be switched ON and OFF separately; either both must be ON or both must be OFF for proper operation.

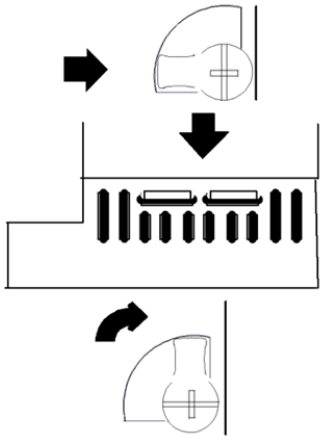
Power for modules in the expansion rack comes from the Power Supply installed on the Expansion Receiver Module. If the expansion rack includes a Power Supply Booster Carrier and additional rack Power Supply, it must be tied to the same source as the Power Supply on the Expansion Receiver Module.

## 4.4 Installing Power Supply Modules

Power Supply modules install directly onto the CPU module, Expansion Receiver Modules, and supplementary power supply carriers.

The Power Supply on the CPU or Expansion Receiver Module supplies +5V and +3.3V to downstream modules through the mating connector. The number of modules that can be supported depends on the power requirements of the modules. Additional booster Power Supplies can be used as needed to meet the power needs of all modules. If the rack includes any Power Supply Booster Carrier and additional rack Power Supply, it must be tied to the same source as the Power Supply on the CPU. The configuration software provides power calculations with a valid hardware configuration. Power Supply installation instructions are given below.

**Figure 31**

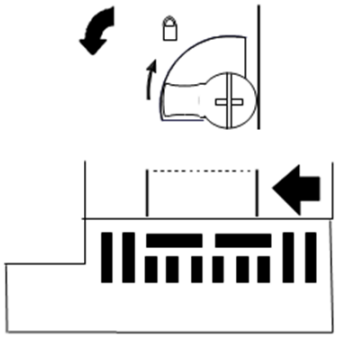


1. The latch on the power supply must be in the unlocked position.
2. Align the connectors and the latch post and press the power supply module down firmly, until the two tabs on the bottom of the power supply click into place. Be sure the tabs are fully inserted in the holes in bottom edge of the CPU, ERM, or carrier.
3. Turn the latch to the locked position to secure the power supply.

## 4.4.1 Removing the Power Supply

Exercise care when working around operating equipment. Devices may become very hot and could cause injury.

**Figure 32**



1. Remove power.
2. urn the latch to the unlocked position as illustrated.
3. Press the flexible panel on the lower edge of the power supply to disengage the tabs on the power supply from the holes in the carrier.
4. Pull the power supply straight off.

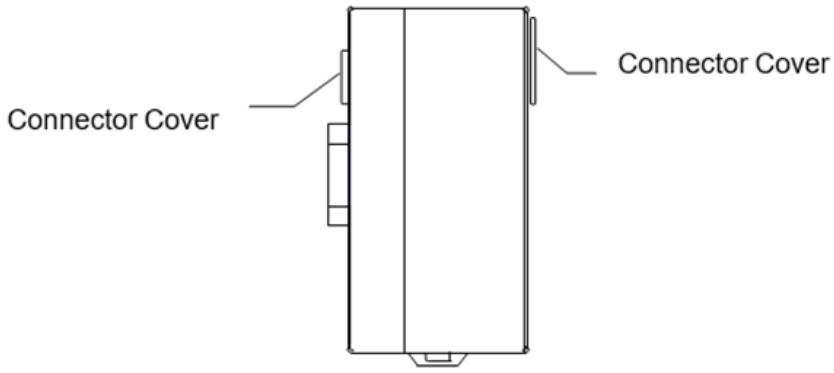
# 4.5 Installing Additional Modules

A CPU or Expansion Receiver Module can serve up to 8 additional I/O and option modules on the same section of DIN rail. Power must be off before adding a carrier to the “rack”.

Before joining carriers to the CPU or ERM, remove the connector cover on the righthand side of the CPU/ERM. Do not discard this cover; you will need to install it on the last carrier. It protects the connector pins from damage and ESD during handling and use.

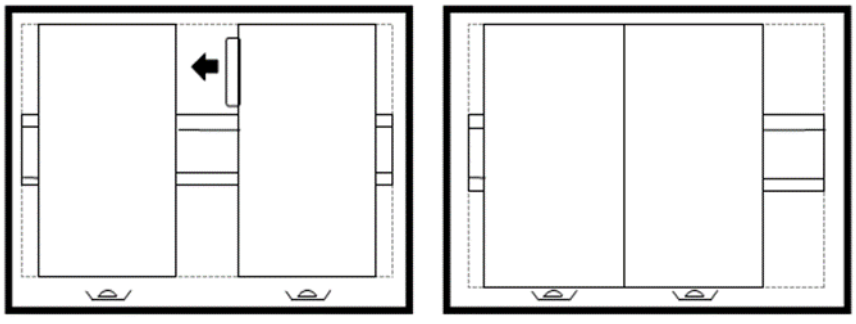
Do not remove the connector cover on the lefthand side.

**Figure 33**




Install each carrier close to the previously-installed carrier, then slide the properly-aligned carriers together to join the mating connectors. To avoid damaging the connector pins, do not force or slam carriers together.

**Figure 34**

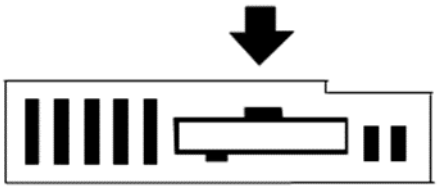


DIN-rail clamps (available as part number IC200ACC313) should be installed at both ends of the station to lock the modules in position.

# 4.6 Activating or Replacing the Backup Battery

The CPU module is shipped with a battery already installed. The battery holder is located in the top side of the CPU module. Before the first use, activate the battery by pulling and removing the insulator tab.

**Figure 35**



## 4.6.1 Lithium Battery Replacement

To replace the battery, use a small screwdriver to gently pry open the battery holder.

Replace battery only with one of the following:

| | |
|---|---|
| Emerson | IC200ACC001 |
| Panasonic | BR2032 |

Use of another battery may present a risk of fire or explosion.

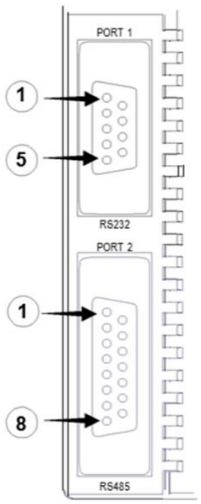
**⚠CAUTION**

**Battery may explode if mistreated.**

Do not recharge, disassemble, heat above 100 deg.C (212 deg.F) or incinerate.

# 4.7 Serial Port Connections

**Figure 36**




## 4.7.1 Providing Power to an External Device from Port 2

If either port is set up for communications with a serial device that requires 100mA or less at 5VDC, the device can obtain power from Port 2.

## 4.7.2 Cable Lengths and Baud Rates

Maximum cable lengths (the total number of feet from the CPU to the last device attached to the cable) are:

Port 1 (RS-232) = 15 meters (50 ft.)

Port 2 (RS-485) = 1200 meters (4000 ft.)

Both ports support configurable baud rates, as listed in the CPU descriptions in this manual.

The following pre-assembled cables are available:

| | |
|---|---|
| IC200CBL001 | CPU Programming Cable RS232 |
| IC200CBL002 | Expansion Firmware Upgrade Cable |

## 4.7.3 Port 1: RS-232

### Pin Assignments for Port 1

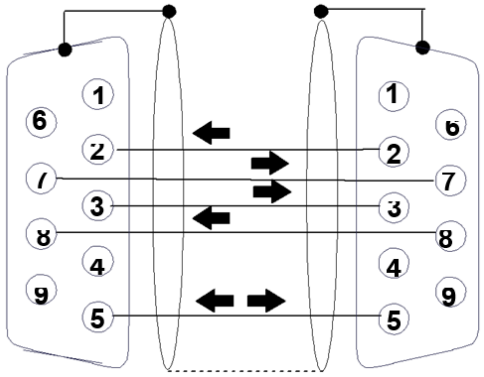
Port 1 is an RS-232 port with a 9-pin female D-sub connector. It is used as the boot loader port for upgrading the CPU firmware. The pinout of Port 1 allows a simple straight-through cable to connect with a standard AT-style RS-232 port. Cable shielding attaches to the shell.

| Pin | Signal | Direction | Function |
|---|---|---|---|
| 1 | n/c | | |
| 2 | TXD | Output | Transmit Data output |
| 3 | RXD | Input | Receive Data input |
| 4 | n/c | | |
| 5 | GND | -- | 0V/GND signal reference |
| 6 | n/c | | |
| 7 | CTS | Input | Clear to Send input |
| 8 | RTS | Output | Request to Send output |
| 9 | n/c | | |
| Shell | SHLD | -- | Cable Shield wire connection / 100% (Continuous) shielding cable shield connection |

## RS-232 Point to Point Connection

In point-to-point configuration, two devices are connected to the same communication line. For RS-232, the maximum length is 15 meters (50ft).

**Figure 37**




The shield must connect to shell of connectors on both ends of the cable.

| PC 9-Pin Serial Port | CPU Port1 |
|---|---|
| 9-pin female | 9-pin male |
| (2) RXD | (2) TXD |
| (3) TXD | (3) RXD |
| (5) GND | (5) GND |
| (7) RTS | (7) CTS |
| (8) CTS | (8) RTS |

## Connector and Cable Specifications for Port 1

Vendor Part numbers below are provided for reference only. Any part that meets the same specification can be used.

| | | | | |
|---|---|---|---|---|
| Cable: Belden 9610 | Computer cable, overall braid over foil shield<br>5 conductor †<br>30 Volt / 80°C (176°F)<br>24 AWG tinned copper, 7x32 stranding | | | |
| 9 Pin Male Connector: | Type: Crimp | Vendor: ITT/Cannon<br>AMP | Plug: DEA9PK87F0<br>205204-1 | Pin:<br>030-2487-017<br>66506-9 |
| | Solder | ITT/Cannon<br>AMP | ZDE9P<br>747904-2 | --<br>-- |
| Connector Shell: | Kit * –ITT Cannon DE121073-54 [9-pin size backshell kit]:<br>Metal-Plated Plastic (Plastic with Nickel over Copper) †<br>Cable Grounding Clamp (included)<br>40° cable exit design to maintain low-profile installation<br>Plus –ITT Cannon 250-8501-010 [Extended Jackscrew]:<br>Threaded with #4-40 for secure attachment to CPU001 port †<br>Order Qty 2 for each cable shell ordered | | | |

† Critical Information – any other part selected should meet or exceed this requirement.

* Use of this kit maintains the 70mm installed depth.

# 4.7.4 Port 2: RS-485

## Pin Assignments for Port 2

Port 2 is an RS-485 port with a 15-pin female D-sub connector. This can be attached directly to an RS-485 to RS-232 adapter.

| Pin | Signal | Direction | Function |
|---|---|---|---|
| 1 | SHLD | -- | Cable Shield Drain wire connection |
| 2, 3, 4 | n/c | | |
| 5 | P5V | Output | +5.1VDC to power external devices (100mA max.) |
| 6 | RTSA | Output | Request to Send (A) output |
| 7 | GND | -- | 0V/GND reference signal |
| 8 | CTSB’ | Input | Clear to Send (B) input |
| 9 | RT | -- | Resistor Termination (120 ohm) for RDA’ |
| 10 | RDA’ | Input | Receive Data (A) input |
| 11 | RDB’ | Input | Receive Data (B) input |
| 12 | SDA | Output | Transmit Data (A) output |

| Pin | Signal | Direction | Function |
|---|---|---|---|
| 13 | SDB | Output | Transmit Data (B) output |
| 14 | RTSB | Output | Request to Send (B) output |
| 15 | CTSA' | Input | Clear to Send (A) input |
| Shell | SHLD | -- | Cable Shield wire connection / 100% (Continuous) shielding cable shield connection |

## Connector and Cable Specifications for Port 2

Vendor Part numbers below are provided for reference only. Any part that meets the same specification can be used.

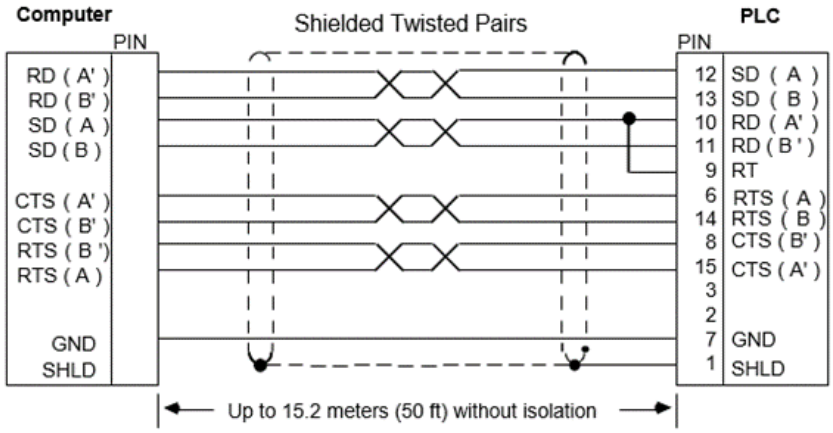
| | | | | |
|---|---|---|---|---|
| Cable: Belden 8105 | Low Capacitance Computer cable, overall braid over foil shield<br>5 Twisted-pairs †<br>Shield Drain Wire †<br>30 Volt / 80°C (176°F)<br>24 AWG tinned copper, 7x32 stranding<br>Velocity of Propagation = 78%<br>Nominal Impedance = 100Ω † | | | |
| 15 Pin Male Connector: | Type: Crimp | Vendor:<br>ITT/Cannon AMP | Plug: DAA15PK87F0<br>205206-1 | Pin:<br>030-2487-017<br>66506-9 |
| | Solder | ITT/Cannon AMP | ZDA15P<br>747908-2 | --<br>-- |
| Connector Shell: | Kit* – ITT Cannon DA121073-50 [15-pin size backshell kit]:<br>Metal-Plated Plastic (Plastic with Nickel over Copper) †<br>Cable Grounding Clamp (included)<br>40° cable exit design to maintain low-profile installation<br>Plus – ITT Cannon 250-8501-009 [Extended Jackscrew]:<br>Threaded with (metric) M3x0.5 for secure attachment †<br>Order Qty 2 for each cable shell ordered | | | |

† Critical Information – any other part selected should meet or exceed this requirement.

# 4.7.5 RS-485 Point to Point Connection with Handshaking

In point-to-point configuration, two devices are connected to the same communication line. For RS-485, the maximum cable length is 1200 meters (4000 feet). Modems can be used for longer distances.

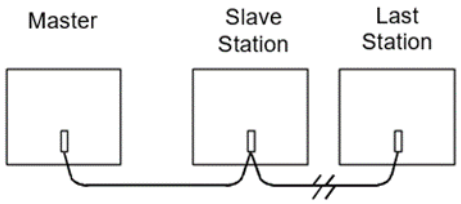
**Figure 38**




## 4.7.6 RS-485 Multidrop Serial Connections

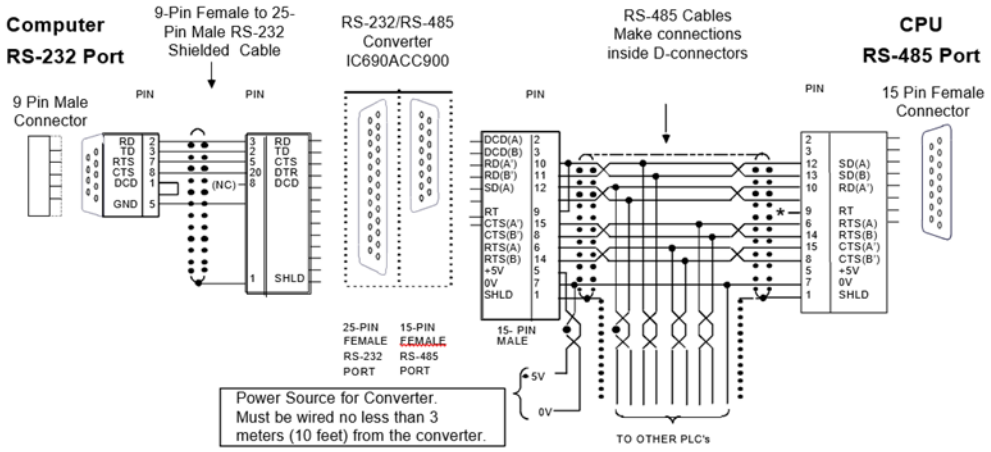
In the multidrop configuration, the host device is configured as the master and one or more PLCs are configured as slaves. The maximum distance between the master and any slave may not exceed 4000 feet (1200 meters). This figure assumes good quality cables and a moderately noisy environment. A maximum of 8 slaves can be connected using RS-485 in a daisy chain or multidrop configuration. The RS-485 line must include handshaking and use wire type as specified earlier.

**Figure 39**




When wiring RS-485 multidrop cables, reflections on the transmission line can be reduced by daisy-chaining the cable as shown below. Make connections inside the connector to be attached to the PLC. Avoid using terminal strips to other types of connectors along the length of the transmission line.

**Figure 40**




Termination resistance for the Receive Data (RD) signal must be connected only on units at the ends of lines. This termination is made at the CPU by connecting a jumper between pin 9 and pin 10 inside the D-shell connector. Ground Potential: Multiple units not connected to the same power source must have common ground potential or ground isolation for proper operation of the system.

# 4.8 Ethernet Connection for CPUE05

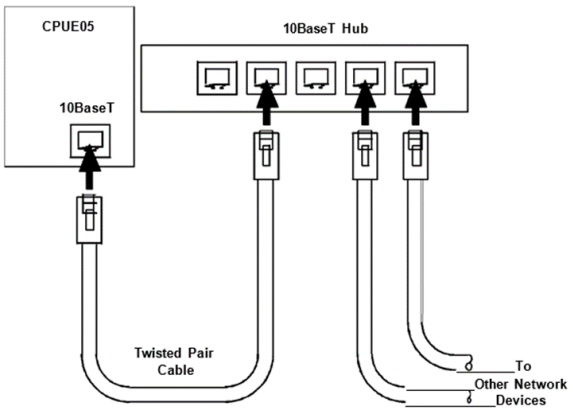
The Ethernet port on PLC module IC200CPUE05 connects directly to a

10BaseT (twisted pair shielded) network without an external transceiver. Connect the port to an external 10BaseT hub or switch or a hub or repeater with auto- sense of 10/100 using a twisted pair cable. Cables are readily available from commercial distributors. Emerson recommends purchasing rather than making cables. Your 10BaseT twisted pair shielded cables must meet the applicable IEEE 802 standards.

## 4.8.1 Network Connection

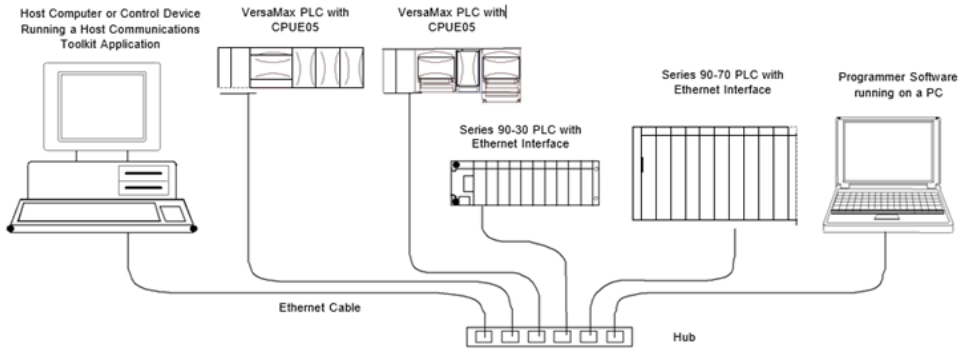
Connection of the CPUE05 to a 10BaseT network is shown below:

**Figure 41**



The cable between each node and a hub or repeater can be up to 100 meters in length. Typical hubs or repeaters support 4 to 12 nodes connected in a star wiring topology.

**Figure 42**




# 4.9 CE Mark Installation Requirements

The following requirements for surge, electrostatic discharge (ESD), and fast transient burst (FTB) protection must be met for applications that require CE Mark listing:

- The VersaMax PLC is considered to be open equipment and should therefore be installed in an enclosure (IP54).
- This equipment is intended for use in typical industrial environments that utilize anti-static materials such as concrete or wood flooring. If the equipment is used in an environment that contains static material, such as carpets, personnel should discharge themselves by touching a safely grounded surface before accessing the equipment.
- If the AC mains are used to provide power for I/O, these lines should be suppressed prior to distribution to the I/O so that immunity levels for the I/O are not exceeded. Suppression for the AC I/O power can be made using line-rated MOVs that are connected line-to-line, as well as line-to-ground. A good high-frequency ground connection must be made to the line-to-ground MOVs.
- AC or DC power sources less than 50V are assumed to be derived locally from the AC mains. The length of the wires between these power sources and the PLC should be less than a maximum of approximately 10 meters.
- Installation must be indoors with primary facility surge protection on the incoming AC power lines.
- In the presence of noise, serial communications could be interrupted.

# Chapter 5: Configuration

This chapter describes the process by which a VersaMax® CPU and the modules it serves are configured. Configuration determines certain characteristics of module operation and also establishes the program references that will used by each module in the system.

- Autoconfiguration or programmer configuration
- Configuring racks and slots
- Configuring CPU parameters
- Configuring CPU memory allocation
- Configuring serial port parameters
- Storing a configuration from a programmer
- Autoconfiguration

## 5.1 Using Autoconfiguration or Programmer Configuration

VersaMax PLCs can be either autoconfigured or configured from a programmer using configuration software. Both types of configuration are described in this chapter.

### 5.1.1 Autoconfiguration

Autoconfiguration occurs at powerup, when the PLC CPU automatically reads the configuration of the modules installed in the system and creates the overall system configuration. Modules that have software-configurable features can only use their default settings when autoconfigured.

### 5.1.2 Software Configuration

Most PLC systems use a customized configuration that is created using configuration software and stored to the CPU from a programmer.

The CPU retains a software configuration across power cycles. After a software configuration is stored to the CPU, the CPU will not autoconfigure when power-cycled.

The configuration software can be used to:

- Create a new configuration
- Store (write) a configuration to the CPU
- Load (read) an existing configuration from a CPU
- Compare the configuration in a CPU with a configuration file stored in the programmer
- Clear a configuration that was previously stored to the CPU

The CPU stores a software configuration in its non-volatile RAM. Storing a configuration disables autoconfiguration, so the PLC will not overwrite the configuration during subsequent startups.

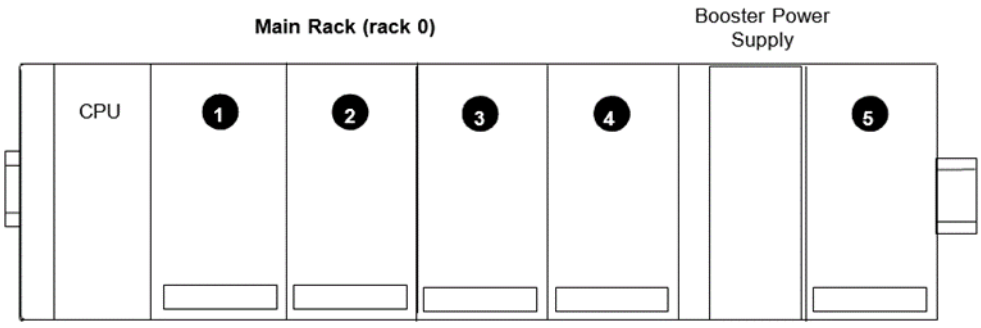
However, actually clearing a configuration from the programmer does cause a new autoconfiguration to be generated. In that case, autoconfiguration is enabled until a configuration is stored from the programmer again.

One of the parameters that can be controlled by the software configuration is whether the CPU reads the configuration and program from Flash at powerup, or from RAM. If Flash is the configured choice, the CPU will read a previously-stored configuration from its Flash memory at powerup. If RAM is the choice, the CPU will read a configuration and application program from its RAM memory at powerup.

## 5.2 Configuring “Racks” and “Slots”

Even though a VersaMax PLC does not have a module rack, both autoconfiguration and software configuration use the traditional convention of “racks” and “slots” to identify module locations in the system. Each logical rack consists of the CPU or an Expansion Receiver module plus up to 8 additional I/O and option modules mounted on the same DIN rail. Each I/O or option module occupies a “slot”. The module next to the CPU or Expansion Receiver module is in slot 1. Booster power supplies do not count as occupying slots.

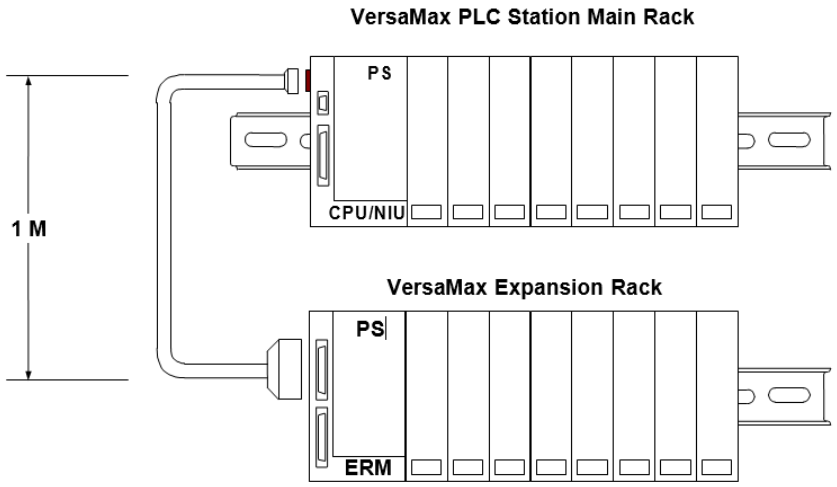
**Figure 43**




The main rack is rack 0. Additional racks are numbered 1 to 7.

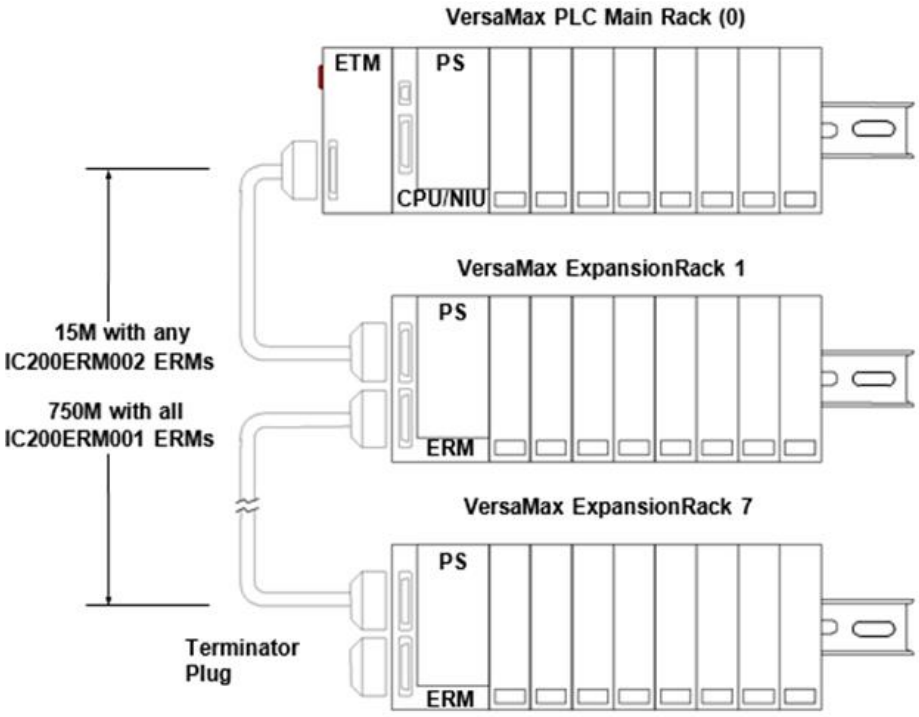
In a system that uses just one expansion rack which is attached to the expansion bus by a non-isolated Expansion Receiver Module (IC200ERM002), the expansion rack must be configured as rack 1.

**Figure 44**




In a system with an Expansion Transmitter Module (IC200BTM001) and up to seven expansion “racks”, each with an Isolated Expansion Receiver Module (IC200ERM001 or IC200ERM002), the additional racks are configured as rack 1 through rack 7.

**Figure 45**



# 5.3 Software Configuration

The configuration software makes it possible to create a customized configuration for the VersaMax PLC system. For CPUE05, it is also used to configure Ethernet Global Data.

When you enter Hardware Configuration for VersaMax equipment folders, the default view is the Rack (Main). A new configuration already includes a default power supply (PWR001) and CPU (CPU001). Both can easily be changed to match the actual hardware in the PLC system.

To configure the PLC, you will:

- Configure the rack type (non-expanded, single-ended expanded, or multi-rack expanded).
- Configure the power supply type and any booster power supplies and carriers. (Note that CPU005 and CPUE05 both require an expanded 3.3V supply.)
- Configure the CPU. This includes changing the CPU type if necessary, and assigning its parameters as described in this chapter.
- Configure the parameters of the CPU serial ports, as explained in this chapter.
- For CPUE05, configure its Ethernet parameters, as explained in chapter 6 Ethernet Configuration.
- Configure the expansion modules if the system has expansion racks.
- Add module carriers and define wiring assignments.
- Place modules on carriers and select their parameters. Configurable parameters of I/O modules are described in the VersaMax Modules, Power Supplies, and Carriers User's Manual (GFK-1504).
- Save the configuration file so that it can be stored to the PLC.

Step-by-step instructions for using the configuration software are provided in the VersaPro Software User's Manual (GFK-1670). Additional information is available in the online help.

## 5.3.1 Configuring CPU and Expansion Parameters

The table below lists configurable parameters for VersaMax PLC CPUs, and for expansion racks.

| Parameter | Description | Default | Choices |
|---|---|---|---|
| **Scan Parameters** | | | |
| Sweep Mode | Normal: sweep runs until it is complete.<br>Constant: sweep runs for time specified in Sweep Tmr. | Normal | Normal, Constant Sweep |
| Sweep Times (mSecs) | If Constant Sweep mode was selected, a Constant Sweep<br>Time (in milliseconds) can be specified. | 100mS | 5–200mS |

| Parameter | Description | Default | Choices |
|---|---|---|---|
| **Settings Parameters** | | | |
| I/O Scan-Stop | Determines whether I/O is to be scanned while the PLC is in STOP mode. | No | Yes, No |
| Powerup Mode | Selects powerup mode. | Last | Last, Stop, Run |
| Logic/Configuration From | Source of program and configuration when the PLC is powered up. | RAM | RAM, Flash |
| Registers | Selects source of register data when PLC is powered up. | RAM | RAM, Flash |
| Passwords | Determines whether the password feature is enabled or disabled. (If passwords are disabled, the only way to enable them is to clear the PLC memory.) | Enabled | Enabled, Disabled |
| Checksum Words per Sweep | The number words in the application program to be checksummed each sweep | 8 | 8 to 32 |
| Default Modem Turnaround Time | Modem turnaround time (10ms/unit) This is the time required for the modem to start data transmission after receiving the transmit request. | 0mS | 0–255mS |
| Default Idle Time | Time (in seconds) the CPU waits to receive the next message from the programming device before it assumes that the programming device has failed and proceeds to its base state. Communication with the programmer is terminated and will have to be reestablished. | 10 | 1–60 |
| SFC Timer Faults | Enables or disables viewing of SFC Timer faults. | Disabled | Enabled /Dis abled |
| SNP ID | | None | Editable |
| Switch Run/Stop | Determines whether the switch will control Run/Stop mode operation | Enabled | Enabled, Disabled |
| Switch Memory Protect | Determines whether the switch will control RAM memory protection. | Disabled | Enabled, Disabled |
| Diagnostics | Unless your application requires unusually fast power up, leave this setting ENABLED. The DISABLED setting causes the PLC to power up without running diagnostics. | Enabled | Enabled, Disabled |
| Fatal Fault Override | Determines whether fatal faults will normally be overridden. | Disabled | Enabled, Disabled |

| Parameter | Description | Default | Choices |
|---|---|---|---|
| EZ Program Store | Specifies where data that is read from the EZ Program Store device will be loaded. | RAM only | RAM only, RAM & Flash |

## Configuring CPU Memory Allocation

CPU001 and CPU002 (release 2.0 or later), CPU005 and CPUE05 have configurable user memory. The configurable memory is equal to the sum of the application program, hardware configuration, registers (%R), analog inputs (%AI), and analog outputs (%AQ). The amount of memory allocated to the application program and hardware configuration are automatically determined by the actual program and configuration entered from the programmer.

The rest of the configurable memory can easily be configured to suit the application. For example, an application may have a relatively large program that uses only a small amount of registers and analog memory. Similarly, there might be a small logic program but a larger amount of memory needed for registers and analog inputs and outputs.

### 5.3.2 Configurable Memory for CPU Module IC200CPU001, CPU002, CPU005

| | |
|---|---|
| Configurable memory | CPU001: 34K bytes maximum<br>CPU002: 42K bytes maximum<br>CPU005: 128K bytes maximum |
| Application program size (not configurable)<br>CPU001, for rel. 1.50 compatibility<br>CPU002, for rel. 1.50 compatibility | 128 bytes minimum<br>12K bytes<br>20K bytes |
| Hardware configuration size (not configurable) | 400 bytes minimum |
| Registers (%R)<br>CPU001/002, for rel. 1.50 compatibility | 256 bytes (128 words) minimum<br>4,096 bytes (2048 words) |
| Analog Inputs (%AI) | 256 bytes (128 words) minimum |
| Analog Outputs (%AQ) | 256 bytes (128 words) minimum |

### 5.3.3 Configurable Memory for CPU Module IC200CPUE05

| | |
|---|---|
| Configurable memory | 128K bytes maximum |
| Application program size (not configurable) | 128 bytes minimum |
| Hardware configuration size (not configurable) | 528 bytes minimum |
| Registers (%R) | 256 bytes (128 words) minimum |
| Analog Inputs (%AI) | 256 bytes (128 words) minimum |
| Analog Outputs (%AQ) | 256 bytes (128 words) minimum |

If you reconfigure memory allocation from the default sizes, storing a hardware configuration to the PLC in the future will clear memory contents. If you want to retain memory contents, first load memory contents from the PLC to the programmer. Then, restore memory when you store the hardware configuration from the programmer to the PLC.

## 5.3.4 Configuring Serial Port Parameters

Both ports on a VersaMax PLC CPU are configurable for SNP slave or RTU slave operation. 4-wire and 2-wire RTU are supported. For CPUE05 only, port 1 can also be configured (on another tab) for Local Station Manager operation. The Local Station Manager parameters may differ from the Port A parameters.

| Feature | Description | Default | Choices |
|---|---|---|---|
| Port Mode | Defines the protocol. | SNP | SNP, Serial I/O, RTU, Disabled. CPUE05 can also be configured as a Local Station Manager. |
| Parity | Determines whether parity is added to words | Odd. For CPUE05, when Port Mode is Local Station Manager, default is None. | Odd, Even, None |
| Data Rate (bps) | Data transmission rate (in bits per second). | Serial comms modes: 19200 | SNP: 4800, 9600, 19200, 38400 |
| | | | RTU: 1200, 2400, 4800, 9600, 19200, 38400, 57600 |
| | | | Serial I/O: 4800, 9600, 19200, 38400, 57600 |
| | | CPUE05 in Local Station Manager mode: 9600 | Local Station Manager mode: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 |
| Flow Control (not required if Port Mode is SNP) | Specifies the method of flow control to use. When changing “Flow Control” from “None” to “Hardware”, Turnaround Delay is reset to 0. | None | RTU mode: None, Hardware |
| | | | Serial I/O mode: None, Hardware, Software |
| | | | CPUE05 in Local Station Manager mode: None, Hardware |
| Timeout (If Port Mode is SNP) | Specifies the set of timeout values to be used by Protocol. | Long | Long, Medium, Short, None |
| Stop Bits (If Port Mode is SNP or Serial I/O) | Number of stop bits used in transmission. (Most serial devices use one stop bit; slower devices use two.) | 1 | 1, 2 |

| Feature | Description | Default | Choices |
|---|---|---|---|
| SNP ID | 8-byte ID for Port 1. | None | Editable |
| Receive to transmit delay | Delay between receiving last character of a message to asserting RTS | 0 | SNP: Not available<br>RTU and Serial IO: 0-255 (units of 10ms, e.g. 10=100ms) |
| Turnaround delay | Delay between asserting RTS and transmitting a message | SNP: none<br>RTU & Serial IO: 0 | SNP: Long, Medium, Short, none<br>RTU & Serial IO: 0-255 (units of 10ms, e.g. 10=100ms) |
| RTS drop delay | Delay between when the last character of a message is transmitted and when RTS is dropped. | 0 | SNP: Not Available<br>RTU and Serial IO: 0-255 (units of 10ms, e.g. 10=100ms) |

The VersaPro software allows configuration of RTU and Serial I/O at 115.2K baud. However, these baud rates are not supported by the CPU. If a configuration using these baud rates is stored to the PLC:

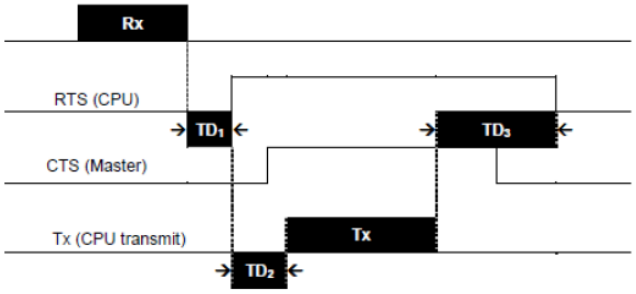
For RTU, an "Unsupported Feature in Configuration" fault is logged and the PLC transitions to Stop Faulted mode.

For Serial I/O, the same fault is logged when the transition to Run mode occurs. The PLC will immediately transition to Stop Faulted mode.

## 5.3.5 RTU and Serial IO Delays

- The "receive to transmit", "turnaround", and "RTS drop delay" parameters can be configured to customize communications timing for radio modems.
- **receive to transmit delay:** The minimum length of time between the CPU receiving the last character of an incoming message and the CPU asserting RTS. Asserting RTS is followed by the transmission of the response message. This delay is configured as a "minimum" time because the actual delay is dependent upon the CPU sweep time.
- **turnaround delay:** The length of time between the CPU asserting RTS and the CPU beginning to transmit a message.
- **RTS drop delay**: The length of time between the CPU transmitting the last character of a response message and the CPU dropping RTS. The RTS drop delay can vary by ± 1 ms.

**Figure 46**




- $TD_1$ is the Receive to Transmit delay
- $TD_2$ is the Turnaround Delay
- $TD_3$ is the RTS Drop Delay

## 5.3.6 Configuration Required to use Winloader

The Winloader utility, which can be used for firmware updates, requires SNP configuration. If Port 1 is configured for another mode or forced to Local Station Manager operation, Winloader will not be able to do a firmware update on port 1.

## 5.3.7 Note for RTU Communications

When using RTU communications, it may be necessary to increase the RTU timeout configured on the master device as the PLC slave scan time increases. It is not necessary to change the configuration of the VersaMax CPU itself, however.

## 5.3.8 Storing a Configuration from a Programmer

Ordinarily, a VersaMax PLC system is configured by creating a configuration file on the programmer (computer), then transferring the file from the programmer to the PLC CPU via the CPU port. The CPU stores the configuration file in its non-volatile RAM memory. The configuration is stored whether I/O scanning is enabled or not. After the configuration is stored, I/O scanning is enabled or disabled according to the newly-stored configuration parameters.

### Autoconfiguration and Storing a Configuration

Clearing a configuration from the programmer causes a new autoconfiguration to be generated. Autoconfiguration remains enabled until the configuration is stored from the programmer again. Storing a configuration disables autoconfiguration.

### Storing a Configuration with Non-default Memory Allocation

If you reconfigure reference tables from the default sizes, storing a hardware configuration to the PLC in the future will clear memory contents. If you want to retain memory contents, first load reference memory contents from the PLC to the programmer. Then, re-store reference memory when you store the hardware configuration from the programmer to the PLC.

### Default Serial Port Parameters

When a programmer is first connected, the PLC communicates using the default communications parameters: 19,200 baud, odd parity, one start bit, one stop bit, and eight data bits. If these parameters are re-configured, the new settings will be used at powerup instead.

### Serial Port Configuration Takes Effect After Removing Programmer

If a hardware configuration is stored to the CPU, the configuration for the serial port to which the programmer is connected is not actually installed until the programmer is removed. After removal of the programmer, there is a delay before the new protocol begins operating. This delay is equal to the configured T3' time.

## 5.4 Autoconfiguration

When autoconfiguration is enabled and no previous autoconfiguration exists, at powerup the CPU automatically reads the configuration of the modules installed in the system and creates an overall system configuration. If a previous autoconfiguration is present at powerup, the configuration is processed as described on the next page.

Modules that have software-configurable features use their default settings when autoconfigured. These features are described in the VersaMax Modules, Power Supplies, and Carriers Manual (GFK-1504).

At powerup, the CPU by default automatically generates a configuration that includes all of the modules that are physically present in the system, starting at slot 1 of rack 0 (the main rack). Autoconfiguration of a rack stops at the first empty slot or faulted module and continues with the next rack. For example, if there are modules physically present in slots 1, 2, 3, 5, and 6, the modules in slots 5 and 6 are not autoconfigured.

To autoconfigure a system with expansion racks, either all racks must be powered from the same source or the expansion racks must be powered up before the main rack.

## 5.4.1 Autoconfiguration Assigns Reference Addresses

Modules are automatically assigned reference addresses in ascending order. For example, if the system contains a 16-point input module, an 8- point input module, a 16-point output module, and another 16-point input module, in that order, the input modules are assigned reference addresses of %I0001, %I0017, and %I0025, respectively. For modules that utilize multiple data types (for example, mixed I/O modules), each data type is assigned reference addresses individually.

## 5.4.2 Autoconfiguration Diagnostics

**Module Present But Non-Working During Autoconfiguration**: If a module is physically present but not working during autoconfiguration, the module is not configured and the CPU generates an extra module diagnostic.

**Empty Slot During Autoconfiguration**: Autoconfiguration of a rack stops at the first empty slot. Modules located after the empty slot are not autoconfigured. The CPU generates an extra module diagnostic for each of them.

**Previously-Configured Modules** Present During Autoconfiguration: Previously- configured modules are not removed from the configuration during autoconfiguration unless no modules are present in the system. For example, if modules are configured in slots 1, 2, and 3 then power is removed and the module in slot 1 is removed, when power is reapplied the modules in slots 2 and 3 are autoconfigured normally. The original module in slot 1 is not removed from the configuration. The CPU generates a loss of module diagnostic for slot 1.

**Different Module Present During Autoconfiguration**: If a slot was previously- configured for one module type but has a different module installed during autoconfiguration, the CPU generates a configuration mismatch diagnostic. The slot remains configured for the original module type.

**Unconfigured Module Installed After Autoconfiguration**: If a module that was not previously-configured is installed-after powerup, the CPU generates an extra module diagnostic and the module is not added to the configuration.

**Previously-configured Module Installed After Autoconfiguration**: If a module that was previously-configured but missing at powerup is installed-after powerup, the CPU generates an addition of module diagnostic and the module is added back into the I/O scan.

**All Modules Removed After Autoconfiguration:** If all modules are absent at powerup, the CPU clears the configuration. This allows modules to be inserted and added to the configuration at the next powerup.

## 5.4.3 Diagnostic Message Summary

| | |
|---|---|
| **addition of module** | A module is present at powerup but not configured. It is added to the configuration.<br>Autoconfiguration is enabled and the module is capable of being autoconfigured. |
| **addition of module** | A previously-configured module is inserted after powerup. The CPU resumes scanning of the module. |
| **configuration mismatch** | A module was found at or after powerup that does not match the configuration for that slot. |
| **extra module** | 1. A module is present at powerup but not configured.<br>2. Autoconfiguration is not enabled.<br>3. A previously-unconfigured module is inserted after powerup. |
| **loss of module** | A configured module is missing during powerup or normal operation. |
| **addition of rack** | 1. An Expansion Receiver Module that was not previously configured is present during configuration.<br>2. During normal operation, communication is restored with a previously missing or failed Expansion Receiver Module. The CPU starts scanning I/O for the modules in that rack. “Addition of Module” faults are not generated when scanning resumes. However, if communications cannot be restored with any modules in the rack, “Loss of Module” faults are generated. |
| **loss of rack** | 1. A previously configured Expansion Receiver Module is not present during configuration.<br>2. During normal operation, a previously working Expansion Receiver Module stops working. Modules in the same expansion rack are terminated. |
| **extra rack** | A previously-unconfigured Expansion Receiver module is inserted after powerup. Modules in the expansion rack are ignored. |
| **Expansion Transmitter mismatch** | 1. An Expansion Transmitter Module (IC200ETM001) is present but not configured.<br>2. An Expansion Transmitter Module (IC200ETM001) is configured but not present. |
| **expansion bus speed change** | The expansion bus speed automatically calculated by the CPU during autoconfiguration has changed. |
| **unsupported feature** | A module is present that is not supported by the CPU. |

# Chapter 6: Ethernet Configuration

This chapter describes the configuration needed for the Ethernet interface of VersaMax CPU module IC200CPUE05:

- Ethernet configuration overview
- Configuring the characteristics of the Ethernet interface
- Configuring Ethernet Global Data
- Configuring Advanced User Parameters

The Ethernet interface configuration described in this chapter must be set up in addition to the basic CPU configuration described in chapter 5, Configuration.

## 6.1 Ethernet Configuration Overview

The Ethernet configuration for CPU module IC200CPUE05 includes:

- Configuring the characteristics of the Ethernet interface. This is part of the CPU configuration.
- Configuring Ethernet Global Data. This is reached via the "rack operations" configuration.
- (Optional, not required for most systems). Configuring advanced parameters. This requires creating a separate ASCII parameter file that is stored to the PLC with the hardware configuration.
- (Optional, not required for most systems). Setting up Port 1 for Local Station Manager operation. This is part of the basic CPU configuration as described in chapter 5. Note that Local Station Manager parameters are configured independently of the Port 1 parameters.

After the configuration is completed and stored to the PLC, it is maintained in memory by the PLC CPU. The configuration may be saved into and retrieved from Flash memory, which provides nearly permanent backup of the configuration data across loss of power and battery backup. Every time CPUE05 is powered up or has its configuration changed or cleared, it delivers the Ethernet configuration data back to the Ethernet interface.

The Ethernet interface portion of CPUE05 saves its configuration data in battery-backed memory. If the CPU battery backup is lost and the configuration has not been saved to Flash, the Ethernet interface loses its backup configuration data. If that happens, after powerup the Ethernet interface operates with its factory default settings until it is reconfigured. This default operation includes reverting to an IP address of 0.0.0.0. Because the backup Ethernet configuration data is actually stored by the Ethernet interface portion of CPUE05, it is not affected by a PLC Clear Configuration operation. When the PLC Configuration is cleared, the CPU operates in Autoconfiguration mode, as described below.

### 6.1.1 Autoconfiguration

If the PLC CPU has not had a configuration stored from the programmer, it automatically creates its own configuration at powerup. To create the Autoconfiguration, the CPU reads configuration data from each module and from the Ethernet interface. This includes an Advanced User Parameter file for the Ethernet interface. When an Autoconfiguration is present in the PLC CPU, it is possible to edit some of the Ethernet configuration parameters from the Station Manager. This changes the parameters that are stored in the Ethernet interface itself. If the PLC is power-cycled or cleared, the edited configuration will be retrieved by the CPU from the Ethernet interface.

## 6.2 Configuring the Ethernet Interface

The CPU's fundamental Ethernet operating characteristics must be correctly configured for proper operation over an Ethernet network. The default configuration cannot supply valid network address data.

| Parameters | Description |
|---|---|
| **Configuration Mode** | This is fixed as TCP/IP. It cannot be changed. |
| **IP Address, Subnet Mask, and Gateway IP Address** | The IP Address is the unique address of the Ethernet interface as a node on the network. On a large network, a subnet mask can be used to identify a section of the overall network. A gateway address can be used to identify a gateway that joins one network with another.<br>These parameters must be correct, or the Ethernet interface may be unable to communicate on the network and/or network operation may be disrupted. It is especially important that each node on the network is assigned a *unique* IP address.<br>These values should be assigned by the person in charge of your network (the network administrator). TCP/IP network administrators are familiar with these parameters. If you have no network administrator and are using a simple isolated network with no gateways, you can use the following values as local IP addresses:<br>10.0.0.2 First PLC<br>10.0.0.3 Second PLC<br>10.0.0.4 Third PLC<br>..<br>10.0.0.254 PLC Programmer or host<br>Also, in this case, set the subnet mask and Gateway IP address to 0.0.0.0.<br>See chapter 13 for more detailed information about IP Addressing and gateways.<br>***Note**: If this simple, isolated network is ever connected to another network, the IP addresses 10.0.0.2 through 10.0.0.254 must not be used and the subnet mask and Gateway IP address must be assigned by the network administrator. **The IP addresses must be assigned so that they are compatible with the connected network**.* |

| Parameters | Description |
|---|---|
| **Status Address** | The beginning reference for 10 bytes of Ethernet status data. The content of this data is described in chapter 13, "Checking the Status of the Ethernet Interface."<br>The Status address can be assigned to %I, %Q, %R, %AI or %AQ memory. The default value is the next available %I address.<br>*Note: Do not use the 10 bytes assigned to the Status bits for other purposes or your data will be overwritten.* |
| **Status Length** | This value is automatically set to either 80 bits (for %I and %Q Status address locations) or 5 words (for %R, %AI, and %AQ Status address locations). |
| **Network Time Servers** | IP addresses of up to 3 NTP time servers used to synchronize timestamps in produced Ethernet Global Data exchanges. If no NTP time servers are configured here, the Ethernet interface is initialized from the clock in the CPU instead. See "Timestamping of Ethernet Global Data Exchanges" in chapter 13 for more information. This feature is supported IC200CPUE05-HK and previous versions only.) |

# 6.3 Configuring Ethernet Global Data

VersaMax CPU IC200CPUE05 can be configured for up to 32 Ethernet Global Data exchanges (any combination of produced and consumed). (See "Ethernet Global Data" in chapter 13 for a discussion of this feature).

Configuration defines both the content of an exchange, its data ranges, and

its operational characteristics. Each Ethernet Global Data produced, or consumed exchange must be configured individually for each PLC.

You can configure:

- Up to 1200 data ranges for all Ethernet Global Data exchanges.
- Up to 100 data ranges per exchange.
- A data length of 1 byte to 1400 bytes per exchange. The total size of an exchange is the sum of the lengths of all of the data ranges configured for that exchange.

Different exchanges may have different data ranges. Multiple exchanges can also share some or all of the same data ranges even if the exchanges are produced at different rates.

*Note: The programming software will not permit consumed exchanges to share data ranges.*

The Ethernet Global Data configuration screens are reached via the rack configuration (not the CPU configuration).

## 6.3.1 Before You Configure EGD Exchanges

Before configuring Ethernet Global Data exchanges, you will need to collect information about the PLCs that will be exchanging the data. Note that this information will be needed for each PLC's configuration. See chapter 13 Ethernet Communications for details.

- Determine for each PLC what data needs to be produced and consumed.
- Make a list of the IP addresses of the Ethernet Interfaces in the PLCs that are being used to produce or consume the exchanges.
- Identify the members of up to 32 groups of devices that will share Ethernet Global Data exchanges.
- Decide on appropriate repetition rates and timeout periods for the exchanges.
- Identify the content of each exchange in the producer and identify appropriate data ranges in the consumers to receive the data.
- It is not necessary to consume all of the data from a produced exchange in each consumer. A consumed exchange may be configured to ignore specified data ranges.

## 6.4 Configuring a Global Data Exchange for a Producer

Each Global Data exchange must be configured in the producer as defined below. The exchange must also be configured in each consumer, as explained next.

| Parameters | Description |
|---|---|
| Local Producer ID | The address that uniquely identifies the CPUE05 as an Ethernet Global Data device across the network. It is a dotted-decimal number. The default is the same as the IP address of the CPUE05. The default can be changed. |
| Exchange ID | A number that identifies a specific data exchange. |
| Adapter Name | Always 0.0 for CPUE05. |
| Consumer Type | Select whether the data's destination will be a single device (IP address) or one of 32 predefined device groups (Group ID). See "Ethernet Global Data" in chapter 13 for more information. |
| Consumer Address | If the "Consumer Type" above is IP Address, this is the IP address of a single device to receive the exchange. If the "Consumer Type" is Group ID, this is the group's ID number (1–32). See chapter 13 Ethernet Communications for more information about IP Addresses. |
| Send Type | Currently fixed at "always". Ethernet Global Data will always be sent when the PLC's I/O scan is enabled. It will not be sent when the I/O scan is disabled. |

<table>
<tr><th>Parameters</th><th>Description</th></tr>
<tr><td>Producer Period</td><td>The scheduled repetition period for sending the data on the network. The range is 10–3,600,000 milliseconds (10 milliseconds to 1 hour). The default is 200 milliseconds. Round this value to the nearest 10 milliseconds before you enter it. The producer period has a resolution of 10 milliseconds. If you enter a value such as 12 milliseconds, the actual producer period will be rounded up to 20 milliseconds.<br>For easier troubleshooting and efficient network usage, set the Producer Period to the same value as the Consumer Period. Do not produce data faster than is required by your application. For example, it is usually not useful to produce data faster than the scan time of the producer or consumer PLCs. This reduces the load on the network and on the devices, providing capacity for other transfers.</td></tr>
<tr><td>Reply Rate</td><td>Currently not used.</td></tr>
<tr><td>Status Word<br><br>example:</td><td>A data range that identifies the memory location where the status value for the produced exchange will be placed. See “Checking the Status of an Exchange” in chapter 13 for details. Note that the Status Word address must be unique; it is not automatically assigned the next highest address.
<table>
<tr><th>Offset</th><th>Reference</th><th>Low Point</th><th>High Point</th><th>Description</th></tr>
<tr><td>Status</td><td>%R</td><td>99</td><td>99</td><td>Status: Where the PLC will place the status data.</td></tr>
</table></td></tr>
<tr><td>Exchange Data Ranges<br><br>example:</td><td>A list of 1 to100 data ranges that will be sent in the exchange. Data is sent as a contiguous set of bytes. See “Checking the Status of an Ethernet Global Data Exchange” in chapter 13 for details. The total size can be up to 1400 bytes. The list of data ranges to be sent in an exchange specifies:
<table>
<tr><th>Offset</th><th>Reference</th><th>Low Point</th><th>High Point</th><th>Description</th></tr>
<tr><td>0.0</td><td>%R</td><td>100</td><td>105</td><td>Conveyor1 in PLC1</td></tr>
<tr><td>10.0</td><td>%I</td><td>345</td><td>352</td><td>Conveyor1 limit switch in PLC1</td></tr>
</table></td></tr>
</table>

# 6.5 Configuring a Global Data Exchange for a Consumer

To receive a Global Data Exchange, configure the following information:

| Parameters | Description |
|---|---|
| **Local Producer ID** | The address that uniquely identifies the CPUE05 as an Ethernet Global Data device across the network. The default is the same as the IP address of the CPUE05. The default can be changed. |
| **Exchange ID** | A number that identifies that specific data exchange. It must match the Exchange ID specified in the produced exchange (in the sending device). |

| Parameters | Description |
|---|---|
| **Adapter Name** | Always 0.0 for CPUE05 |
| **Producer ID** | The Local Producer ID of the device sending the exchange. |
| **Group ID** | Used only if the same data is consumed by more than one consuming device. Enter the same Group ID that has been configured as the “Consumer Address” in the producer device. |
| **Consumer Period** | Not used. Default is 200mS. |
| **Update Timeout** | The maximum time the Ethernet interface allows between seeing samples on the network without reporting a refresh error status. This error status means a first or subsequent packet of data has not arrived within the specified time. The range is 0, or 10–3,600,000 milliseconds. The value should be at least double the producer’s producer period value. The default is 0, which disables timeout detection.<br>The update timeout period should be greater than the exchange production period. (A value at least twice the production period is recommended.)<br>Round this value to the nearest 10 milliseconds before you enter it. The update timeout has a resolution of 10 milliseconds. If you enter a value such as 22 milliseconds, the actual update timeout will be rounded up to 30 milliseconds. |
| **Status Word** | A data range that identifies the memory location where the status value for the consumed exchange will be placed. See chapter 13 Ethernet Communications for details of the status value. Note that the Status Word address must be unique; it is not automatically assigned the next highest address. |

| example | **Offset** | **Reference** | **Low Point** | **High Point** | **Description** |
|---|---|---|---|---|---|
| | Status | %R | 99 | 99 | Status: Where the PLC will place the status data. |

| **Parameters** | **Description** |
|---|---|

<table>
<tr><th>Parameters</th><th colspan="5">Description</th></tr>
<tr><td>Time Stamp</td><td colspan="5">A data range that identifies the memory location where the timestamp of the last data packet will be placed. The timestamp is not an actual date; it is an 8-byte value representing the time elapsed since midnight, January 1, 1970. The first four bytes contain a signed integer representing seconds and the next four bytes contain a signed integer representing nanoseconds. This value represents the time in the producer when the data sample originated. It can be examined to determine if a new packet received from the network has a new data sample or if it is the same data received previously.<br>The timestamp information produced by the PLC currently has a resolution of 100 microseconds if no network synchronization is used. If NTP is used to perform network time synchronization, the timestamp information has a resolution of 1 millisecond and has ±10 millisecond accuracy between PLCs on the same LAN.<br>NTP may be enabled in the configuration of the CPUE05. Once NTP time synchronization is configured, the CPUE05 will synchronize itself to an external NTP time server if one exists.<br>This feature is supported IC200CPUE05-HK and previous versions only.)</td></tr>
<tr><td rowspan="2">example:</td><td>Offset</td><td>Reference</td><td>Low Point</td><td>High Point</td><td>Description</td></tr>
<tr><td>Time Stamp</td><td>%R</td><td>91</td><td>94</td><td>Time Stamp: Optional place for the PLC to put the timestamp.</td></tr>
<tr><td>Exchange Data Ranges</td><td colspan="5">A list of 1 to100 data ranges that will be received in the exchange. Data is received as a contiguous set of bytes. The total size of all combined elements can be up to<br>1400 bytes. For consumed exchanges, %S memory types and override references are not allowed. See Table 4-2 for valid memory types.<br>Note: If the consumed exchange length does not match that of the produced exchange, PLC Faults and Ethernet exception entries occur.<br>The list of data ranges to be received in an exchange specifies:</td></tr>
<tr><td rowspan="3">example:</td><td>Offset</td><td>Reference</td><td>Low Point</td><td>High Point</td><td>Description</td></tr>
<tr><td>0.0</td><td>%R</td><td>100</td><td>104</td><td>Conveyor1 in PLC1</td></tr>
<tr><td>10.0</td><td>%I</td><td>257</td><td>264</td><td>Conveyor1 limit switch in PLC1</td></tr>
</table>

### 6.5.1 Selective Consumption

Not all data ranges within a produced exchange need to be consumed by each PLC. For example, a producer is producing an exchange consisting of a 4-byte floating point value, followed by a 2-byte integer, followed by a 2- byte analog value. If the consuming PLC wants to consume only the analog value and place it into %AI003, the consumer might be configured as shown below.

| Offset | Reference | Low Point | High Point | Description |
|---|---|---|---|---|
| 0 | Ignore (bytes) | 1 | 6 | Ignore float and integer |
| 6 | %AI | 3 | 3 | |

Note that the total length of the exchange must be the same in producer and consumer, even if the consumer is ignoring bytes at the end of the message. Failure to configure any ignored bytes in the consumed exchange will result in exchange exception log and fault table entries, error status in the exchange status data, and no data being transferred for the exchange.

## 6.6 Configuring Advanced User Parameters

Advanced User Parameters are internal operating parameters used by the Ethernet interface. For most applications, the default Advanced User Parameters should not be changed.

If it is necessary to modify any of these parameters, it must be done by creating an Advanced User Parameter file, using any ASCII text editor. This file must contain the names and values of only those parameters that are being changed. The file must be named “AUP_0_0.apf”. The completed file must be placed into the PLC folder that contains the PLC configuration. When the entire hardware configuration is stored from the programmer to the PLC, the programmer software also stores the parameters from the AUP_0_0.apf file.

### 6.6.1 Format of the Advanced User Parameters File

The Advanced User Parameters file must have this format:

AUP_0_0

<parameter name> = <parameter value>

<parameter name> = <parameter value>

<parameter name> = <parameter value>

All parameter names are lowercase. The equal sign ( = ) is required between the parameter name and parameter value.

Parameter values are converted to lowercase unless they are enclosed in a pair of double quotes. The format for the individual parameter values depends on the parameter. Numeric parameters are entered in decimal or hexadecimal format; hexadecimal values must end with an 'h' or 'H' character. IP address parameters must be entered in standard dotted decimal format. Character string values are case-sensitive. Uppercase parameter values must be enclosed within a pair of double quotes. (The enclosing quotes are not part of the data and are removed during processing).

Comments in the file must start with a semicolon character. All characters in the same line following a semicolon are ignored. Blank lines are also ignored.

The following example sets the station manager password to "system" and the IP time-to-live for point-to-point Ethernet Global Data exchanges to 4.

## 6.6.2 Example Advanced User Parameter File

AUP_0_0

| | | |
|---|---|---|
| stpasswd = "system" | ; | set the password to "system" |
| gucast_ttl=4 | ; | set the EGD unicast IP TTL to 4 |

## 6.6.3 Advanced User Parameter Definitions

The following Advanced User Parameters can be configured for the CPUE05 Ethernet interface.

| Name | Description | Default | Range |
|---|---|---|---|
| staudp | Remote Station Manager UDP port | 18245 (4745H) | 0-65535 (ffffH) |
| stpasswd | Station Manager password | "system" | 0-8 char, case sensitive, no spaces |
| crsp_tout | Transfer/Response timeout value (in seconds) | 16 (0010H) | 10 – 3600 (0e10H) |
| fflush | ARP cache timeout interval (in seconds) | 0 – 604800 (93a80H) | 600 (0258H) |
| gctl_port | UDP port for Ethernet Global Data control messages | 7937 (1f01H) | 0-65535 (ffffH) |
| gdata_port | UDP port for point-to-point Ethernet Global Data messages | 18246 (4746H) | 0-65535 (ffffH) |
| gbcast_ttl | IP time-to-live for global broadcast messages (hop count) | 1 (1H) | 0-255 (00ffH) |
| gucast_ttl | IP time-to-live for point-to-point messages (hop count) | 16 (10H) | 0-255 (00ffH) |
| gXX_udp | UDP port for host group XX | 18246 (4746H) | 0-65535 (ffffH) |
| gXX_ttl | IP time-to-live for host group (multicast) messages (hop count) | 1 (1H) | 0-255 (00ffH) |

| Name | Description | Default | Range |
|---|---|---|---|
| gXX_addr | IP group address for host group XX (must be class D address) | 224.0.7.XX | 224.0.0.2 - 239.255.255.255 |
| ittl | IP header default time-to-live (hop count) | 64 (0040H) | 0-255 (00ffH) |
| ifrag_tmr | IP fragment timeout interval (in seconds) | 3 (0003H) | 0-65535 (ffffH) |
| wnodelay | TCP nodelay option (0=inactive, 1=active) | 0 (000H) | 0,1 |
| wkal_idle | TCP keepalive timer value (in seconds) | 240 (00f0H) = 4.0 minutes | 0-65535 (ffffH) |
| wkal_cnt | TCP keepalive probe count | 2 (0002H) | |
| wkal_intvl | TCP keepalive probe interval (in seconds) | 60 (003cH) | |
| wmsl | TCP maximum segment lifetime (in seconds) | 30 (001eH) | |
| wsnd_buf | TCP send buffer size in bytes | 4096 (1000H) | 0-32767 (7fffH) |
| wrcv_buf | TCP receive buffer size in bytes | 4096 (1000H) | |
| nmin_poll1 | NTP min. poll interval for host 1. The value specifies log(2) of the interval in seconds (eg: the value 3 means 8 secs, 4 means 16 sec, etc) | 6 (0006H) = 64 seconds | 4 – 14 (000eH) (16 – 16384 sec) |
| nmax_poll1 | NTP maximum poll interval for host 1 (in log(2) of seconds) | 10 (000aH) = 1024 sec. | |
| nmin_poll2 | NTP min. poll interval for host 2 (in log(2) of seconds) | 6 (0006H) = 64 sec. | |
| nmax_poll2 | NTP max. poll interval for host 2 (in log(2) of seconds) | 10 (000aH) = 1024 sec. | |
| nmin_poll3 | NTP min. poll interval for host 3 (in log(2) of seconds) | 6 (0006H) = 64 sec. | |
| nmax_poll3 | NTP max. poll interval for host 3 (in log(2) of seconds) | 10 (000aH) = 1024 sec. | |
| nsync_tout | NTP synchronization timeout period (in seconds). The max. time between network time updates to remain synchronized). | 300 (012cH) | 150 – 65535 (0096H – ffffH) |

*Note: The NTP feature is supported in IC200CPUE05-HK and previous versions only*

# Chapter 7: CPU Operation

This chapter describes the operating modes of the VersaMax™ PLC CPUs, and shows the relationship between the application program execution and other tasks performed by the CPU.

### CPU Operating Modes

The application program in a PLC executes repeatedly. In addition to executing the application program, the PLC CPU regularly obtains data from input devices, sends data to output devices, performs internal

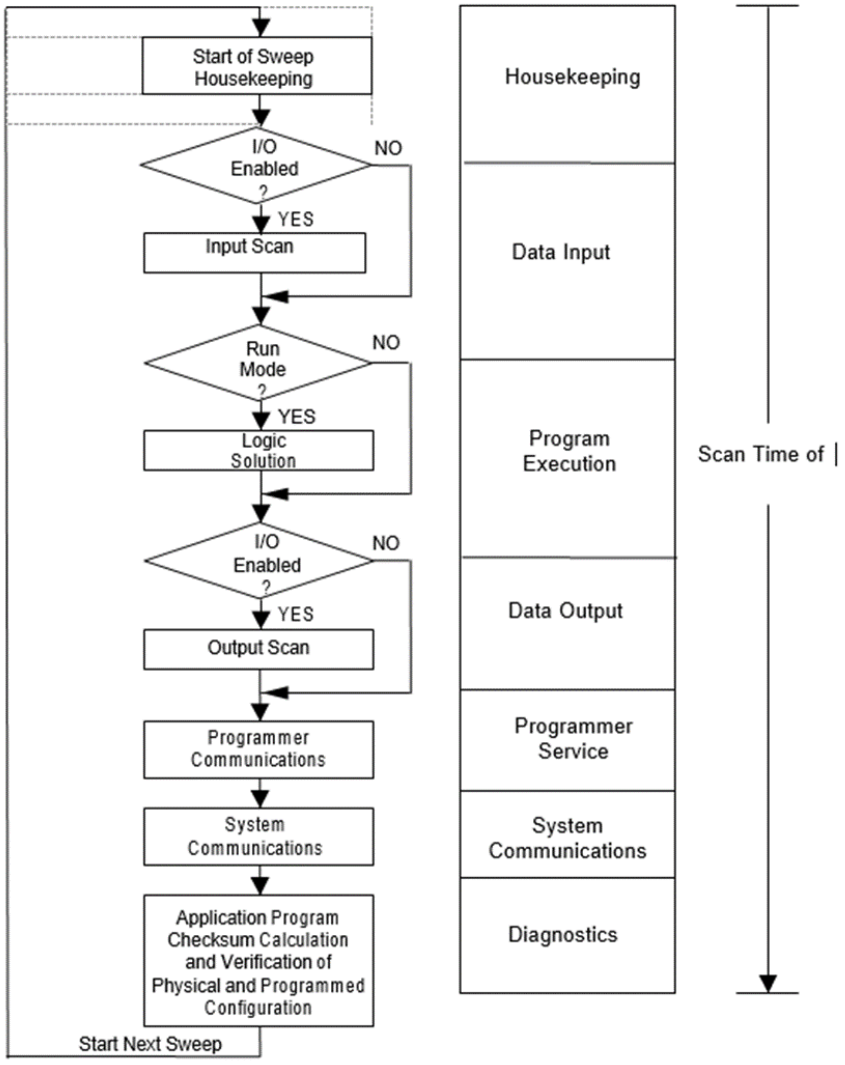
housekeeping, and performs communications tasks. This sequence of

operations is called the sweep.

- The basic operating mode of the PLC is called Standard Sweep mode. In this mode, the CPU performs all parts of its sweep normally. Each sweep executes as quickly as possible with a different amount of time consumed each sweep.
- The PLC may instead operate in Constant Sweep Time mode. In this mode, the CPU performs the same series of actions but each sweep takes the same amount of time.
- The PLC may also be in either of two Stop modes:
  - Stop with I/O Disabled mode.
  - Stop with I/O Enabled mode.

# 7.1 Parts of the CPU Sweep

**Figure 47**



### Parts of the CPU Sweep

| | |
|---|---|
| **Start of Sweep Housekeeping** | Housekeeping includes the tasks necessary to prepare for the start of the sweep. Before starting the actual sweep, the CPU: |
| | Calculates the sweep time |
| | Schedules the start of the next sweep |
| | Determines the mode of the next sweep |
| | Updates the fault reference tables |
| | Resets the Watchdog timer |
| | If the PLC is in Constant Sweep Time mode, the sweep is delayed until the required sweep time elapses. If the required time has already elapsed, the OV_SWP %SA0002 contact is set, and the sweep continues without delay. Next, the CPU updates timer values (hundredths, tenths, and seconds). |
| **Input Scan** | When the sweep starts, the CPU first scans inputs from input modules and option modules that provide input-type data. Modules are scanned in ascending reference address order. Discrete input modules are scanned before analog input modules. The CPU stores this new input data in the appropriate memories. If the CPU has been configured to not scan I/O in Stop mode, the input scan is skipped when the CPU is in Stop mode. |
| | For CPUE05, if the CPU is in run mode and the consumer period of an Ethernet Global Data exchange has expired, the CPU copies the data for that exchange from the Ethernet interface to the appropriate reference memory. |
| **Application Program Logic Scan** | Next, the CPU solves the application program logic. It always starts with the first instruction in the program. It ends when the END instruction is executed. Solving the logic creates a new set of output data. |
| **Output Scan** | Immediately after the logic solution, the CPU scans all output modules in ascending reference address order. The output scan is completed when all output data has been sent. |
| | If the CPU has been configured to not scan I/O in Stop mode, the output scan is also skipped when the CPU is in Stop mode. |
| | For CPUE05, if I/O is enabled and the producer period of an Ethernet Global Data exchange has expired, the CPU copies the data for that exchange from the appropriate reference memory to the Ethernet interface. |
| **Programmer Communications Window** | If there is a programming device attached, the CPU next executes the programmer communications window. The programmer communications window will not execute if there is no programmer attached. |
| | In the default limited window mode, each sweep the CPU honors one service request. The time limit for programmer communications is 6 milliseconds. If the programmer makes a request that requires more than 6 milliseconds to process, the processing is spread out over multiple sweeps. |

| | |
|---|---|
| **System Communications Window** | Next, the CPU processes communications requests from intelligent option modules. The modules are polled in round-robin fashion, so no module has priority. |
| | In default ("Run to Completion") mode, the length of the system communications window is limited to 400 milliseconds. If a module makes a request that requires more than 400 milliseconds to process, the request is spread out over multiple sweeps. |
| | In Limited mode, option modules that communicate with the PLC using the system window have less impact on sweep time, but response to their requests is slower. |
| **Diagnostics** | A checksum calculation is performed on the application program at the end of every sweep. You can specify from 0 to 32 words to be checksummed. If the calculated checksum does not match the reference checksum, the program checksum failure exception flag is raised. A fault is entered in the PLC fault table and the PLC goes to Stop mode. If the checksum calculation fails, the programmer communications window is not affected. |
| | Each sweep, the CPU verifies the physical configuration of one module against its programmed configuration. A missing, additional, or mismatched module causes a fault to be generated. |

## 7.2 Standard CPU Sweep Operation

Standard Sweep operation is the normal operating mode of the PLC CPU. In Standard Sweep operation, the CPU repeatedly executes the application program, updates I/O, and performs communications and other tasks shown in the diagram:

1. The CPU performs its start-of-sweep housekeeping tasks.
2. It reads inputs.
3. It executes the application program.
4. It updates outputs
5. If a programming device is present, the CPU communicates with it.
6. It communicates with other devices.
7. It performs diagnostics

Except for communicating with a programmer, all these steps execute every sweep. Programmer communications occur only when needed.

In this mode, the CPU performs all parts of its sweep normally. Each sweep executes as quickly as possible with a different amount of time consumed each sweep.

## 7.2.1 The Sweep Windows

The programmer communications window and the system communications window have two operating modes:

**Limited Mode** — The execution time of the window is 6ms. The window terminates when it has no more tasks to complete or when 6ms has elapsed.

**Run to Completion Mode** — Regardless of the time assigned to a particular window the window runs until all tasks within that window are completed (up to 400ms).

SVCREQ 2 can be used in the application program to obtain the current times for each window.

## 7.2.2 The Watchdog Timer

When the CPU is in Standard Sweep mode, the Watchdog Timer catches failure conditions that could cause an unusually long sweep. The length of the Watchdog Timer is 500 milliseconds. It restarts from zero at the beginning of each sweep.

If the sweep takes longer than 500mS, the OK LED on the CPU module goes off. The CPU resets, executes its powerup logic, generates a watchdog failure fault, and goes to Stop mode. Communications are temporarily interrupted.

## 7.2.3 Constant Sweep Time Operation

If the application requires that each CPU sweep take the same amount of time, the CPU can be configured to operate in Constant Sweep Time mode. This operating mode assures that the inputs and outputs in the system are updated at constant intervals. This mode can also be used to implement a longer sweep time, to assure that inputs have time to settle after receiving output data from the program.

### Changing the Configured Default for Constant Sweep Mode

If the PLC is in STOP mode, its Configured Constant Sweep mode can be edited. After this is done, the configuration must be Stored to the CPU for the change to take effect. Once stored, Constant Sweep Time mode becomes the default sweep mode.

### The Constant Sweep Timer

During operation in Constant Sweep Time mode, the CPU's Constant Sweep Timer controls the length of the sweep. The timer length can be 5 to 500 milliseconds. The time should be at least 10 milliseconds longer than the CPU's sweep time when it is in Standard Sweep mode, to prevent extraneous oversweep faults.

If the Constant Sweep Timer expires before the sweep completes, the CPU still completes the entire sweep, including the windows. However, it automatically provides notice when a too-long sweep has occurred. On the next sweep after the oversweep, the CPU places an oversweep alarm in the PLC fault table. Then, at the beginning of the following sweep, the CPU sets the OV_SWP fault contact (%SA0002). The CPU automatically resets the OV_SWP contact when the sweep time no longer exceeds the Constant Sweep Timer. The CPU also resets the OV_SWP contact if it is not in Constant Sweep Time mode.

As with other fault contacts, the application program can monitor this contact to keep informed about the occurrence of oversweep conditions.

### Enabling/Disabling Constant Sweep Time, Reading or Setting the Length of the Timer

SVCREQ 1 can be included in the application program to enable or disable Constant Sweep Time mode, change the length of the Constant Sweep Time, read whether Constant Sweep Time is currently enabled, or read the Constant Sweep Time length.

## 7.3 CPU Stop Modes

The PLC may also be in either of two Stop modes:

- Stop with I/O Disabled mode
- Stop with I/O Enabled mode

When the PLC is in Stop mode, the CPU does not execute the application program logic. You can configure whether or not the I/O will scanned during Stop mode. Communications with the programmer and intelligent option modules continue in Stop mode. In addition, faulted board polling and board reconfiguration execution continue in Stop mode.

SVCREQ 13 can be used in the application program to stop the PLC at the end of the next sweep. All I/O will go to their configured default states, and a diagnostic message will be placed in the PLC Fault Table.

## 7.4 Flash Memory

A VersaMax PLC stores the current configuration and application in non- volatile battery-backed RAM. The programmer software can be used to store a copy of the current configuration, application program, and reference tables (excluding overrides) to Flash memory. The programmer can also be used to read a previously-stored configuration, application program, or reference tables from Flash into RAM, or to verify that Flash and RAM contain identical data.

By default, the PLC reads the configuration, program logic, and reference tables from RAM at powerup. However, it can be configured to read them from Flash. This is recommended, because data in Flash is non-volatile, even in the case of a battery failure.

# 7.5 Controlling the Execution of a Program

The VersaMax CPU Instruction Set contains several powerful Control functions that can be included in an application program to limit or change the way the CPU executes the program and scans I/O.

## 7.5.1 Calling a Subroutine Block

The CALL function can be used to cause program execution to go to a specific subroutine. Conditional logic placed before the Call function controls the circumstances under which the CPU performs the subroutine logic. After the subroutine is finished, program execution resumes at the point in the logic directly after the CALL instruction.

## 7.5.2 Creating a Temporary End of Logic

The END function can be used to provide a temporary end of logic. It can be placed anywhere in a program. No logic beyond the END function is executed, and program execution goes directly back to the beginning. This ability makes the END function useful for debugging a program.

The END function should not be placed in logic associated with or called by a Sequential Function Chart control structure. If this occurs, the PLC will be placed in STOP/FAULT mode at the end of the current sweep and an SFC_END fault will be logged.

## 7.5.3 Executing Rungs of Logic without Logical Power Flow

The nested Master Control Relay can be used to execute a portion of the program logic with no logical power flow. Logic is executed in a forward direction and coils in that part of the program are executed with negative power flow. Master Control Relay functions can be nested to 8 levels deep.

## 7.5.4 Jumping to Another Part of the Program

The Jump function can be used to cause program execution to move either forward or backward in the logic. When a nested Jump function is active, the coils in the part of the program that is skipped are left in their previous states (not executed with negative power flow, as they are with a Master Control Relay). Jump functions can also be nested.

Jumps cannot span blocks, SFC actions, SFC transitions, or SFC pre- or post-processing logic.

# 7.6 Run/Stop Mode Switch Operation

The CPU Run/Stop mode switch can be configured to place the CPU in Stop or Run mode. It can also be configured to prevent writing to program or configuration memory and forcing or overriding discrete data. It defaults to enabled Run/Stop mode selection and disabled memory protection.

## 7.6.1 Configurable Run/Stop Mode Operation

If Run/Stop mode switch operation is enabled, the switch can be used to place the CPU in Run mode.

- If the CPU has non-fatal faults and is not in Stop/Fault mode, placing the switch in Run position causes the CPU to go to Run mode. Faults are NOT cleared.
- If the CPU has fatal faults and is in Stop/Fault mode, placing the switch in Run position causes the Run LED to blink for 5 seconds. While the Run LED is blinking, the CPU switch can be used to clear the fault table and put the CPU in Run mode. After the switch has been in Run position for at least ½ second, move it to Stop position for at least ½ second. Then move it back to Run position. The faults are cleared and the CPU goes to Run mode. The LED stops blinking and stays on. This can be repeated if necessary.
- If the switch is not toggled as described, after 5 seconds the Run LED goes off and the CPU remains in Stop/Fault mode. Faults stay in the fault table.

## 7.6.2 Configurable Memory Protection

Operation of the switch can be configured to prevent writing to program memory and configuration, and to prevent forcing or overriding data.

## 7.6.3 Summary of CPU Switch Run/Stop Operation

| Run/Stop Mode Configuration | I/O Scan Stop Configuration | Switch Position | CPU Operation |
|---|---|---|---|
| Off | has no effect | has no effect | All modes are allowed. |
| On | has no effect | Run/On | All modes are allowed. |
| On | has no effect | Stop/Off | CPU not allowed to go to Run mode. |
| Off | has no effect | Toggle Switch from Stop to Run | CPU goes to Run mode if no fatal faults are present; otherwise, the Run LED blinks for 5 seconds. |
| On | No | Toggle switch from Run to Stop | PLC goes to STOP–NO IO |
| On | Yes | Toggle switch from Run to Stop | PLC goes to STOP–IO |

# 7.7 Privilege Levels and Passwords

Passwords are an optional configurable feature of the VersaMax PLC. Passwords provide different levels of access privilege to the PLC when the programmer is in Online or Monitor mode. Passwords are not used if the programmer is in Offline mode. Passwords can restrict:

- Changing I/O and PLC configuration data
- Changing programs
- Reading PLC data
- Reading programs

There is one password for each privilege level in the PLC. Each password may be unique or the same password can be used for more than one level. Passwords are one to seven ASCII characters in length.

By default, there is no password protection. Passwords are set up, changed, or removed using the programming software. After passwords have been set up, access to the PLC is restricted unless the proper password is entered. Entering a correct password allows access to the requested level and to all lower levels. For example, the password for level 3 allows access to levels 0, 1, 2, and 3. If PLC communications are suspended, protection automatically returns to the highest unprotected level. For example: If a password is set at levels 2 & 3, but none at level 4, if the software disconnects and reconnects, the access level is 4. Privilege level 1 is always available because no password can be set for this level.

| Level | Access Description |
|---|---|
| 4<br>Least<br>Protected | • Write to all configuration or logic. Configuration may only be written in Stop mode; logic may be written in Stop or Run mode (if run-mode store is supported).<br>• Set or delete passwords for any level.<br>• Plus all access from levels 3,2 and 1.<br>*Note: This is the default if no passwords are defined.* |
| 3 | • Write to all configuration and logic when the CPU is in Stop mode, including word-for-word changes (when supported), the addition/deletion of program logic, and the overriding of discrete I/O.<br>• Read/Write/Verify user flash.<br>• Store reference/override tables.<br>• Change sweep mode.<br>• Plus all access from levels 2 and 1. |

| Level | Access Description |
|---|---|
| 2 | • Write to any data memory, but this does not include storing tables.<br>• This includes the toggle/force of reference values but does not include overriding discrete I/O.<br>• The PLC can be started or stopped.<br>• PLC and I/O fault tables can be cleared.<br>• Plus all access from level 1. |
| 1<br>Most<br>Protected | Read any PLC data except for passwords. This includes reading fault tables, current status, performing datagrams, verifying logic/config, and loading program and configuration from the PLC. No PLC memory may be changed. |

## 7.7.1 Protection Level Request from Programmer

Upon connection to the CPU, the programming software automatically requests the CPU to move to the highest unprotected level. That gives the programmer access to the highest unprotected level without having to specifically request a particular level. A privilege change may be to a lower level or to a higher level. The privilege level is changed from the programmer by entering the new level and the correct password for that level. If the wrong password is entered, the change is denied, and a fault is logged in the PLC fault table. A request to change to a privilege level that is not password-protected is made by supplying the new level and an empty password.

### Notes on Using Passwords

- To re-enable passwords after passwords have been disabled, the PLC must be power-cycled with the battery removed for long enough to completely discharge the super-capacitor and erase the PLC's memory.
- If the passwords prevent changing the run/stop mode, firmware upgrades cannot be performed if the PLC is in run mode.
- The Run/Stop switch (if configured) will place the PLC in run or stop mode regardless of the passwords.

## 7.7.2 The OEM Protection Feature

The OEM protection feature is similar to the passwords and privilege levels and provides an even higher level of security. The feature is enabled or disabled using a 1 to 7 character password called the OEM key. When OEM protection is enabled, no write-access to the PLC program and configuration is permitted. Reading the configuration from the PLC is permitted. In this mode, no user flash operations are allowed. When the OEM key password has been created, the OEM key can be locked in two ways: by choosing the locked setting from the programming software or by power-cycling the PLC. (The OEM key locked status does not change when PLC communications are suspended.)

## Clearing Logic/Configuration, and References

It is possible to clear logic, configuration, and references from the programmer with the CPU at any privilege level, even with the OEM key locked. Operators can clear logic, configuration, and references, and store a new application program to the CPU without knowing passwords. If passwords and/or the OEM key have been set and written to flash, a read from flash updates the protection level. In this case, it is not necessary to reenter the password to gain access to a particular level. A Clear All does not clear user flash.

# Chapter 8: Elements of an Application Program

This chapter provides basic information about the application program for a VersaMax™ PLC.

- Structure of an application program
- Subroutines
- Program languages
- The Instruction Set

## 8.1 Structure of an Application Program

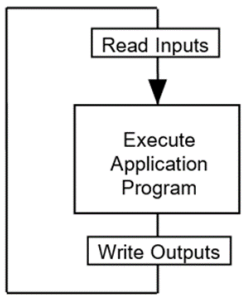
The application program consists of all the logic needed to control the operations of the PLC CPU and the modules in the system.

Application programs are created using the programming software and transferred to the PLC. Programs are stored in the CPU's non-volatile memory.

During the CPU Sweep (described in the previous chapter), the CPU reads input data from the modules in the system and stores the data in its configured input memory locations. The CPU then executes the entire application program once, utilizing this fresh input data. Executing the application program creates new output data that is placed in the configured output memory locations.

After completing the end of the application program, the CPU writes the output data to modules in the system.

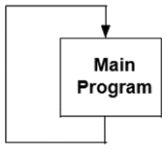
**Figure 48**



# 8.2 Subroutines

The program can consist of one Main program that executes completely during each CPU sweep.

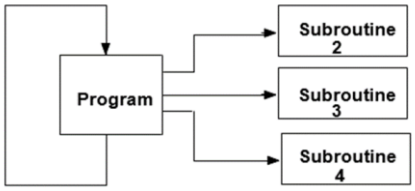
**Figure 49**




Or a program can be divided into subroutines. The maximum size of a main program or subroutine block is 64kB. The program can contain up to 255 subroutines.

Subroutines can simplify programming and reduce the overall amount of logic. Each subroutine can be called as needed. The main program might serve primarily to sequence the subroutine blocks.

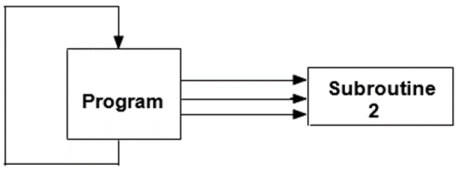
**Figure 50**




A subroutine block can be called many times as the program executes.

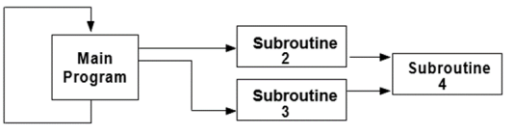
Logic that should be repeated can be placed in a subroutine block, reducing total program size.

**Figure 51**




In addition to being called from the program, subroutine blocks can also be called by other subroutine blocks. A subroutine block can even call itself.

**Figure 52**



The main program is level 1. The program can include up to eight additional nested call levels.

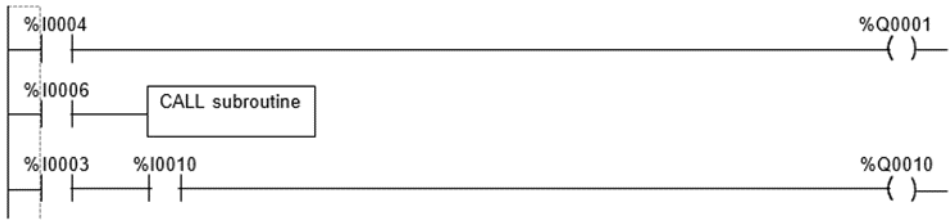
### 8.2.1 Declaring a Subroutine

A subroutine must be declared through the block declaration editor of the programming software.

### 8.2.2 Calling a Subroutine

A subroutine invoked in the program is using a CALL instruction. Up to 64 subroutine block declarations and 64 CALL instructions are allowed for each block in the program.

**Figure 53**




## 8.3 Program Languages

Programs can be created in Ladder Diagram or Instruction List format. The main program or subroutines within the program can also be created in Sequential Function Chart format. The PLC programming software can be used to create both types of logic.

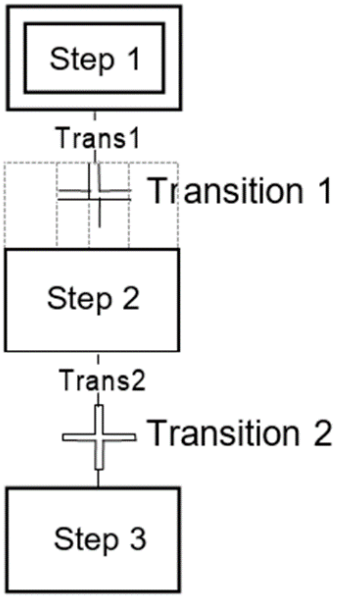
### 8.3.1 Sequential Function Chart

Sequential Function Chart (SFC) is a graphic method of representing the functions of a sequential automated system as a sequence of steps and transitions. Each step represents commands or actions that are either active or inactive.

The flow of control passes from one step to the next through a conditional transition that is either true (1) or false (0). If the transition condition is true (1), control passes from the current step (which becomes inactive) to the next step, which then becomes active.

The logic associated with a step is executed when the step is active. This logic is programmed in Ladder Diagram format. The transitions between steps are also programmed as Ladder Diagram logic.

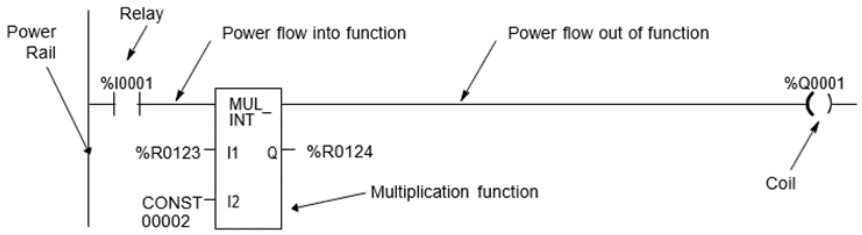
**Figure 54**




## 8.3.2 Ladder Diagram

This traditional PLC programming language, with its rung-like structure, executes from top to bottom. The logic execution is thought of as “power flow”, which proceeds down along the left “rail” of the ladder, and from left to right along each rung in sequence.

**Figure 55**




The flow of logical power through each rung is controlled by a set of simple program functions that work like mechanical relays and output coils.

Whether or not a relay passes logical power flow along the rung depends on the content of a memory location with which the relay has been associated in the program. For instance, a relay might pass power flow if its associated memory location contained the value 1. The same relay would not pass power flow if the memory location contained the value 0.

If a relay or other function in a rung does not pass logical power flow, the rest of that rung is not executed. Power then flows down along the left rail to the next rung.

Within a rung, there are many complex functions that can be used for operations like moving data stored in memory, performing math operations, and controlling communications between the CPU and other devices in the system.

Some program functions, such as the Jump function and Master Control Relay, can be used to control the execution of the program itself.

Together, this large group of Ladder Diagram relays, coils, and functions is called the "Instruction Set" of the CPU.

# 8.4 The Instruction Set

The VersaMax PLC CPU provides a powerful Instruction Set for building application programs.

As a guide to the programming capabilities of the VersaMax PLC, all of the relays, coils, functions, and other elements of the Instruction Set are summarized on the following pages. Complete reference information is included in the documentation and online help for the programming software.

## 8.4.1 Contacts

| | | |
|---|---|---|
| –\| \|– | Normally Open | Passes power if the associated reference is ON. |
| –\|/\|– | Normally Closed | Passes power if the associated reference is OFF. |
| <+>––– | Continuation | Passes power to the right if the preceding continuation coil is set ON. |

## 8.4.2 Coils

| | | |
|---|---|---|
| –( )– | Normally Open | Sets the associated reference ON if the coil receives power. Otherwise OFF. |
| –(/)– | Negated | Sets the associated discrete reference ON if the coil does not receive power. Otherwise OFF. |
| –( )– | Positive Transition | If power flow was OFF to this coil the last time it was executed and is ON this time, then the coil is turned ON. Otherwise, the coil is turned OFF. |
| –( )– | Negative Transition | If power flow was ON to this coil the last time it was executed and is OFF this time, then the coil is turned ON. Otherwise, the coil is turned OFF. |
| –(S)– | SET | Sets the associated discrete reference ON if the coil receives power. It remains set until reset by an –(R)– coil. |
| –(R)– | RESET | Sets the associated discrete reference OFF if the coil receives power. It remains reset until set by an –(S)– coil. |
| –(SM)– | Retentive SET | Sets the associated reference is set ON if the coil receives power. The reference remains set until reset by an –(RM)– coil. Its state is retained through power failure and STOP-TO-RUN |

| | | |
|---|---|---|
| –(RM)– | Retentive RESET | Resets the associated discrete reference OFF if the coil receives power. The reference remains reset until set by an –(SM)– coil. Its state is retained through power failure and **STOP-TO-RUN** transition. |
| –(/M)– | Negated Retentive | Sets the associated discrete reference ON if the coil does not receive power. The state is retained through power failure and **STOP-TO-RUN** transition. Otherwise OFF. |
| –(M)– | Retentive | Sets the associated discrete reference ON if the coil receives power. The state is retained through power failure and **STOP-TO-RUN** transition. Otherwise OFF. |
| ––––<+> | Continuation | If power to the coil is ON, the continuation coil sets the next continuation contact ON. If power is OFF, the continuation coil sets the next continuation contact OFF. |

## 8.4.3 Timers and Counters

| | | |
|---|---|---|
| ondtr | On-Delay Stopwatch Timer | Accumulates time while receiving power The current value is reset to zero when the Reset input receives power. |
| oftd | Off-Delay Timer | Accumulates time while NOT receiving power. |
| tmr | On-Delay Timer | Accumulates time while receiving power The current value is reset to zero when there is no power flow. |
| upctr | Up Counter | Increments by 1 each time the function receives transitional power. |
| dnctr | Down Counter | Counts down from a preset value every time the function receives transitional power. |

## 8.4.4 Math Functions

| | | |
|---|---|---|
| add | Addition | Adds two numbers. |
| sub | Subtraction | Subtracts one number from another. |
| mul | Multiplication | Multiplies two numbers. |
| div | Division | Divides one number by another, yielding a quotient. |
| mod | Modulo Division | Divides one number by another, yielding a remainder. |
| expt | Power of X | Raises X to the power specified by IN and places the result in Q. |
| sin | Trigonometric Sine | Finds the trigonometric sine of a real number. |
| cos | Trigonometric Cosine | Finds the trigonometric cosine of a real number. |
| tan | Trigonometric Tangent | Finds the trigonometric tangent of a real number. |
| asin | Inverse Sine | Finds the inverse sine of a real number. |
| acos | Inverse Cosine | Finds the inverse cosine of a real number. |
| atan | Inverse Tangent | Finds the inverse tangent of a real number. |

| | | |
|---|---|---|
| deg | Convert to Degrees | Performs a RAD_TO_DEG conversion on a real radian value. |
| rad | Convert to Radians | Performs a DEG_TO_RAD conversion on a real degree value. |
| scale | Scaling | Scales an input constant or word value. |
| sqroot | Square Root | Finds the square root of an integer or real value. |
| Log | Base 10 Logarithm | Finds the base 10 logarithm of a real value. |
| ln | Natural Logarithm | Finds the natural logarithm base of a real number. |
| exp | Power of e | Raises the natural logarithm base to the power specified by input. |

## 8.4.5 Relational Functions

| | | |
|---|---|---|
| eq | Equal | Tests for equality between two numbers. |
| ne | Not Equal | Tests for non-equality between two numbers. |
| gt | Greater Than | Tests whether one number is greater than another. Passes power if the first number is greater than the second. |
| ge | Greater Than or Equal To | Tests whether one number is greater than or equal to another |
| lt | Less Than | Tests whether one number is less than another. |
| le | Less Than or Equal To | Test whether one number is greater than or equal to another. |
| range | Range | Test the input value against a range of two numbers. |

## 8.4.6 Bit Operation Functions

| | | |
|---|---|---|
| and | Logical AND | Performs Logical AND of two bit strings. |
| or | Logical OR | Performs Logical OR of two bit strings. |
| xor | Logical | performs Logical Exclusive OR of two bit strings. |
| not | Logical Invert | Performs a logical inversion of a bit string. |
| shl | Shift Left | Shifts a bit string left. |
| shr | Shift Right | Shifts a bit string right. |
| rol | Rotate Left | Rotates a bit string left. |
| ror | Rotate Right | Rotates a bit string right. |
| bittst | Bit Test | Test a bit within a bit string. |
| bitset | Bit Set | Sets one bit within a string to true. |
| bitclr | Bit Clear | Sets one bit within a string to false. |
| bitpos | Bit Position | Locates a bit set to true within a bit string. |
| mskcmp | Masked Compare | Performs a masked compare of two arrays. |

## 8.4.7 Data Move Functions

| | | |
|---|---|---|
| move | Move | Moves one or more bits of data. |
| blkmov | Block Move | Moves a block of up to 7 constants. |
| blkclr | Block Clear | Clears to zero one or more bytes/words of memory. |
| shfreg | Shift Register | Shifts one or more words or bits of data through a block of memory. |
| bitseq | Bit Sequencer | Sequences a 1 through a group of bits in PLC memory. |
| comreq | Communication Request | Sends a communications request. |

## 8.4.8 Table Functions

| | | |
|---|---|---|
| arrmov | Array Move | Copies a specified number of data elements from a source array to a destination array. |
| srh eq | Search Equal | Searches array for values equal to a specified value. |
| srh ne | Search Not Equal | Searches array for values not equal to a specified value. |
| srh gt | Search Greater Than | Searches array for values greater than a specified value. |
| srh ge | Search Greater Than or Equal | Searches array for values greater than or equal to a specified value. |
| srh lt | Search Less Than | Searches array for values less than a specified value. |
| srh le | Search Less Than or Equal | Searches array for values less than or equal to a specified value. |

## 8.4.9 Conversion Functions

| | | |
|---|---|---|
| →bcd-4 | Convert to BCD-4 (From INT) | Converts a number to 4-digit BCD format. |
| →word | Convert to Word (From REAL) | Converts a Real value to Word format. |
| →int | Convert to INT (From BCD-4 or REAL) | Converts a number to signed integer format. |
| →tdint | Convert to DINT (From BCD-4 or REAL) | Converts a number to double precision integer format. |
| →real | Convert to Real (From INT, DINT, BCD-4 or WORD) | Converts a value to real value format. |
| →→int | Truncate to INT (from REAL) | Truncates to a 16-bit signed number. The range is –32,768 to +32,767. |

| | | |
|---|---|---|
| →→dint | Truncate to Double Precision INT (from REAL) | Truncates to a 32-bit signed number. The range is -2,147,483,648 to +2,147,483,647. |

## 8.4.10 Control Functions

| | | |
|---|---|---|
| call | Call | Causes a program execution to go to a specified subroutine block. |
| do io | Do I/O | Services a specified range of inputs or outputs immediately (all inputs or outputs on a module will be serviced if any addresses on that module are included in the function – partial I/O module updates are not performed |
| pidind | Independent PID Algorithm | Selects the non-interacting independent PID algorithm. |
| pidisa | ISA PID Algorithm | Selects the ISA PID algorithm. |
| end | Temporary End of Logic | The program executes from the first rung to the last rung or the END instruction, whichever is encountered first. This instruction is useful for debugging purposes. |
| commnt | Comment | A rung explanation. |
| svcreq | Service | A special PLC service function. |
| mcr | Master Control Relay | Starts a master control relay range. An MCR causes all rungs between the MCR and its subsequent ENDMCR to be executed with no power flow. Up to 8 MCRs can be nested. |
| endmcr | End Master Control Relay | Ends a master control relay range. |
| jump | Jump | Jumps to a specified location indicated by a LABEL in the logic. |
| label | Label | The target location of a JUMP instruction. Multiple Jump instructions can reference the same label. |
| drumseq | Drum Sequencer | (future) Operates like a mechanical drum sequencer, selecting a 16- bit output pattern from an array of stored patterns, and sending it to a set of outputs. |

# Chapter 9: Program Data

This chapter describes the types of data that can be used in an application program, and explains how that data is stored in the VersaMax™ PLC's memory.

- Data memory references
- Retentiveness of data
- Using names and descriptions for program references
- System status references
- Time tick contacts
- How program functions handle numerical data

## 9.1 Data Memory References

The PLC stores program data in both bit memory and word memory. Both bit memory and word memory are divided into different types with specific characteristics.

By convention, each type is normally used for a specific type of data, as explained below. However, there is great flexibility in actual memory assignment.

Individual memory locations are indexed using alphanumeric identifiers called references. The reference's letter prefix identifies the memory area. The numerical value is the offset within that memory area.

### 9.1.1 Word Memory References

Each word memory address (reference) is on a 16-bit word boundary. The PLC uses three types of references for data stored in word memory.

**%AI** Normally used for analog inputs.

**%AQ** Normally used for analog outputs.

**%R** Registers are normally used to store program data in word format.

Word memory is represented below. The example below shows ten addresses. Each has 16 bits that together contain one value. The PLC cannot access individual bits in word memory.

| | | |
|---|---|---|
| **addresses** | 1 | 12467 |
| | 2 | 12004 |
| | 3 | 231 |
| | 4 | 359 |
| | 5 | 14 |
| | 6 | 882 |
| | 7 | 24 |
| | 8 | 771 |
| | 9 | 735 |
| | 10 | 0 |

## 9.1.2 Bit Memory References

Each bit memory address (reference) is on a bit boundary. Data is stored in bit memory as represented below. The illustration shows 160 individually-addressed bits, with address 1 in the upper left and address 160 in the lower right.

**addresses**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

The PLC uses six types of references for data stored in bit memory.

| | |
|---|---|
| %I | Normally used for discrete inputs, and viewable in the Input Status Table. |
| %Q | Normally used for physical output references, and viewable in the Output Status Table. A %Q reference may be either retentive or non-retentive, depending on its use in the program. |
| %M | Normally used to represent internal references. A specific %M reference may be either retentive or non-retentive, depending on its use in the program. |
| %T | Used for temporary references that can be used many times in a program. Data with %T references is not retained through loss of power or RUN-TO-STOP-TO-RUN transitions. %T references cannot be used with retentive coils. |
| %S | System status references, which have specific predefinitions.<br>– %S, %SA, %SB, and %SC can be used for any type of logic contact.<br>– %SA, %SB, and %SC can be used for retentive coils.<br>– %S can be used as inputs to functions or function blocks.<br>– %SA, %SB, and %SC can be used as inputs or outputs of functions and function blocks. |
| %G | Used for Global Data. Data in %G references is retained through power loss. %G references can be used with contacts and retentive coils, but not on non-retentive coils. |

### 9.1.3 Transition Bits and Override Bits

%I, %Q, %M, and %G references have associated transition and override bits. %T, %S, %SA, %SB, and %SC references have associated transition bits only. The CPU uses transition bits for transitional coils. When override bits are set, the associated references can only be changed from the programmer.

## 9.2 Retentiveness of Data

Data is retentive if it is automatically saved when the PLC is stopped, or power cycled. The following data is retentive:

- Program logic
- Fault tables and diagnostics
- Overrides
- Word data (%R, %AI, %AQ)
- Bit data (%I, %SC, %G, fault bits and reserved bits)
- Word data stored in %Q and %M
- Data in %Q or %M references that are used as function block outputs or with retentive coils:
  - -(M)- retentive coils
  - -(/M)- negated retentive coils
  - -(SM)- retentive SET coils
  - -(RM)- retentive RESET coils

The last time a %Q or %M reference is used with a coil, the coil type determines whether the data is retentive or non-retentive. For example, if %Q0001 was last programmed as the reference of a retentive coil, the %Q0001 data is retentive.

However, if %Q0001 was last programmed on a non-retentive coil, then the % Q0001 data is non-retentive.

- %Q or %M references that have been made retentive by specifically declaring them to be retentive. %Q and %M references default to nonretentive.

The following data is non-retentive:

- The states of transition coils.
- %T data
- %S, %SA, and %SB data (but %SC bit data IS retentive)
- %Q and %M references that have not been declared to be retentive.
- %Q and %M references that are used with non-retentive coils:
  - -( )- coils
  - -(/)- negated coils
  - -(S)- SET coils
  - -(R)- RESET coils

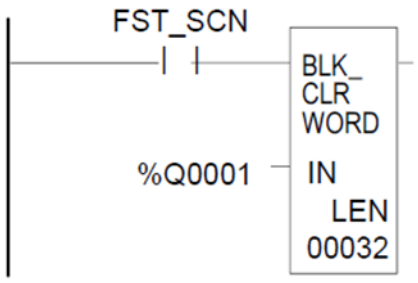
# 9.3 System Status References

The PLC stores system status data in predefined references in %S, %SA, %SB, and %SC memory. Each system status reference has a descriptive name. For example, time tick references are named T_10MS, T_100MS, T_SEC, and T_MIN. Examples of convenience references include FST_SCN, ALW_ON, and ALW_OFF.

## 9.3.1 Using the System Status References

System status references can be used as needed in application programs. For example, the following function block uses the FST_SCN (first scan) status reference to control power flow to a Block Clear function. In this example, at powerup, 32 words of %Q memory (512 points) beginning at %Q0001 are filled with zeros.

**Figure 56**



## 9.3.2 %S References

References in %S memory are read only.

| Reference | Name | Definition |
|---|---|---|
| %S0001 | FST_SCN | Set to 1 when the current sweep is the first sweep. |
| %S0002 | LST_SCN | Reset from 1 to 0 when the current sweep is the last sweep. |
| %S0003 | T_10MS | 0.01 second timer contact. |
| %S0004 | T_100MS | 0.1 second timer contact. |
| %S0005 | T_SEC | 1.0 second timer contact. |
| %S0006 | T_MIN | 1.0 minute timer contact. |
| %S0007 | ALW_ON | Always ON. |
| %S0008 | ALW_OFF | Always OFF. |
| %S0009 | SY_FULL | Set when the PLC fault table fills up. Cleared when an entry is removed and when the PLC fault table is cleared. |
| %S0010 | IO_FULL | Set when the I/O fault table fills up. Cleared when an entry is removed from the I/O fault table and when the I/O fault table is cleared. |
| %S0011 | OVR_PRE | Set when an override exists in %I, %Q, %M, or %G memory. |
| %S0012 | | reserved |
| %S0013 | PRG_CHK | Set when background program check is active. |
| %S0014 | PLC_BAT | Set to indicate a bad battery in the CPU. The contact reference is updated once per sweep. |
| %S0015, 16 | | reserved |
| %S0017 | SNPXACT | SNP-X host is actively attached to CPU port 1. (Port 2 defaults to disabled, and must be activated with a CRQ). |
| %S0018 | SNPX_RD | SNP-X host has read data from CPU port 1. |
| %S0019 | SNPX_WT | SNP-X host has written data to CPU port 1. |
| %S0020 | | Set ON when a relational function using REAL data executes successfully. It is cleared when either input is NaN (Not a Number). |
| %S0021 | FF_OVR | Set to report a Fatal Fault Overide. |
| %S0022 | USR_SW | Set to reflect the state of the CPU mode switch.<br>1 = Run/On<br>0 = Stop/Off |
| %S0023-32 | | reserved |

## 9.3.3 %SA, %SB, and %SC References

References in %SA, %SB, and %SC memory can be both read and written to.

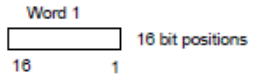
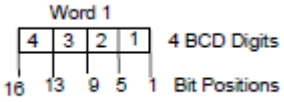
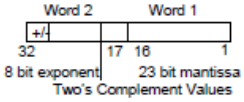
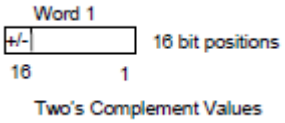
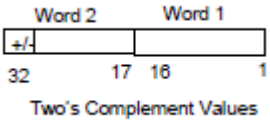
| Reference | Name | Definition |
|---|---|---|
| %SA0001 | PB_SUM | Set when a checksum calculated on the application program does not match the reference checksum. If the fault was due to a temporary failure, the discrete bit can be cleared by again storing the program to the CPU. If the fault was due to a hard RAM failure, the CPU must be replaced. |
| %SA0002 | OV_SWP | Set when a PLC in CONSTANT SWEEP mode detects that the previous sweep took longer than the time specified. Cleared when the PLC detects that the previous sweep did not take longer than specified. Also cleared during transition from STOP to RUN mode. |
| %SA0003 | APL_FLT | Set when an application fault occurs. Cleared when the PLC transitions from STOP to RUN mode. |
| %SA0004-8 | | reserved |
| %SA0009 | CFG_MM | Set when a configuration mismatch is detected during power-up or a configuration store. Cleared by powering up the PLC after correcting the condition. |
| %SA0010 | HRD_CPU | Set when the diagnostics detects a problem with the CPU hardware. Cleared by replacing the CPU module. |
| %SA0011 | LOW_BAT | Set when a low battery fault occurs. Cleared by replacing the battery then powering up the PLC. |
| %SA0012,13 | | reserved |
| %SA0014 | LOS_IOM | Set when an I/O module stops communicating with the CPU. Cleared by replacing the module and cycling system power. |
| %SA0015 | LOS_SIO | Set when an option module stops communicating with the CPU. Cleared by replacing the module and cycling power on the main rack. |
| %SA0016-18 | | reserved |
| %SA0019 | ADD_IOM | Set when an I/O module is added. Cleared by cycling PLC power and when the configuration matches the hardware after a store. |
| %SA0020 | ADD_SIO | Set when an option module is added. Cleared by cycling PLC power and when the configuration matches the hardware after a store. |
| %SA0021-26 | | reserved |
| %SA0027 | HRD_SIO | Set when a hardware failure is detected in an option module. Cleared by replacing the module and cycling PLC power. |
| %SA0028-30 | | reserved |
| %SA0031 | SFT_SIO | Set when an unrecoverable software fault is detected in an option module. Cleared by cycling PLC power and when the configuration matches the hardware. |
| %SB0001-9 | | reserved |

| Reference | Name | Definition |
|---|---|---|
| %SB0010 | BAD_RAM | Set when the CPU detects corrupted RAM memory at powerup. Cleared when RAM memory is valid at powerup. |
| %SB0011 | BAD_PWD | Set when a password access violation occurs. Cleared when the PLC fault table is cleared. |
| %SB0012 | | reserved |
| %SB0013 | SFT_CPU | Set when the CPU detects an unrecoverable error in the software. Cleared by clearing the PLC fault table. |
| %SB0014 | STOR_ER | Set when an error occurs during a programmer store operation. Cleared when a store operation is completed successfully. |
| %SC0001-8 | | reserved |
| %SC0009 | ANY_FLT | Set when any fault occurs. Cleared when both fault tables have no entries. |
| %SC0010 | SY_FLT | Set when any fault occurs that causes an entry to be placed in the PLC fault table. Cleared when the PLC fault table has no entries. |
| %SC0011 | IO_FLT | Set when any fault occurs that causes an entry to be placed in the I/O fault table. Cleared when the I/O fault table has no entries. |
| %SC0012 | SY_PRES | Set as long as there is at least one entry in the PLC fault table. Cleared when the PLC fault table has no entries. |
| %SC0013 | IO_PRES | Set as long as there is at least one entry in the I/O fault table. Cleared when the I/O fault table has no entries. |
| %SC0014 | HRD_FLT | Set when a hardware fault occurs. Cleared when both fault tables have no entries. |
| %SC0015 | SFT_FLT | Set when a software fault occurs. Cleared when both fault tables have no entries. |

## 9.4 How Program Functions Handle Numerical Data

Regardless of where data is stored in memory—in one of the bit memories or one of the word memories–the application program can handle it as different data types.

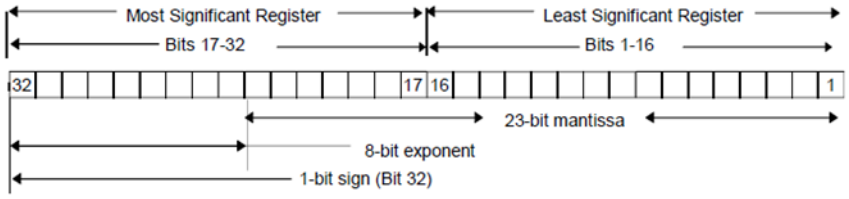
| Type | Name | Description | Data Format |
|---|---|---|---|
| BIT | Bit | A Bit data type is the smallest unit of memory. It has two states, 1 or 0. The programmer functions use the term BOOL for bit-type data. | |
| BYTE | Byte | A Byte data type has an 8-bit value. The valid range is 0 to 255 (0 to FF in hexadecimal). | |

| Type | Name | Description | Data Format |
|---|---|---|---|
| WORD | | A Word data type uses 16 consecutive bits of data memory; but, instead of the bits in the data location representing a number, the bits are independent of each other. Each bit represents its own binary state (1 or 0). The valid range of word values is 0 to +65,535 (FFFF). |  |
| BCD-4 | Four-Digit Binary Coded Decimal | Four-digit BCD numbers use 16-bit data memory locations. Each BCD digit uses four bits and can represent numbers between 0 and 9. BCD coding of the 16 bits has a value range of 0 to 9999. |  |
| REAL | Floating-Point | Real numbers use two consecutive 16-bit memory locations. The range of numbers that can be stored in this format is ± 1.401298E-45 to ± 3.402823E+38. See the next page for more information. |  |
| INT | Signed Integer | Signed integer data uses 16-bit memory locations. Signed integers are represented in 2's complement notation. Bit 16 is the sign bit, (0 = positive, 1 = negative). Their range is -32,768 to +32,767. |  |
| DINT | Double Precision Signed Integer | Double precision signed integers data uses two consecutive 16-bit memory locations. They are represented in 2's complement notation. Bit 32 is the sign bit, (0 = positive, 1 = negative). Their range is -2,147,483,648 to +2,147,483,847. |  |

### 9.4.1 Real Numbers

The REAL data type, which can be used for some Math functions and Numerical functions, is actually floating point data. Floating-point numbers are stored in single precision IEEE-standard format. This format requires 32 bits, which occupy two (adjacent) 16-bit PLC words.

**Figure 57**




For example, if the floating-point number occupies registers %R0005 and %R0006, then %R0005 is the least significant register and %R0006 is the most significant register.

The range of numbers that can be stored in this format is from ± 1.401298E–45 to ± 3.402823E+38 and the number zero.

### 9.4.2 Errors in Real Numbers and Operations

Overflow occurs when a number greater than 3.402823E+38 or less than -3.402823E+38 is generated by a REAL function. The ok output of the function is set OFF; and the result is set to positive infinity (for a number greater than 3.402823E+38) or negative infinity (for a number less than –3.402823E+38). You can determine where this occurs by testing the sense of the ok output.

POS_INF = 7F800000h – IEEE positive infinity representation in hex.

NEG_INF = FF800000h – IEEE negative infinity representation in hex.

If the infinities produced by overflow are used as operands to other REAL functions, they may cause an undefined result. This result is referred to as NaN (Not a Number). For example, the result of adding positive infinity to negative infinity is undefined. When the ADD_REAL function is invoked with positive infinity and negative infinity as its operands, it produces NaN for its result.

## 9.5 Time-tick Contacts

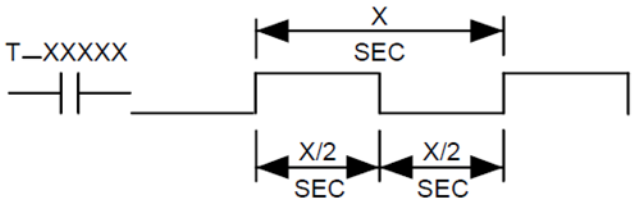
There are four time-tick contacts. They can be used to provide regular pulses of power flow to other program functions. The four time-tick contacts have time durations of 0.01 second, 0.1 second, 1.0 second, and 1 minute.

The state of these contacts does not change during the execution of the sweep. These contacts provide a pulse having an equal on and off time duration.

The contacts are referenced as T_10MS (0.01 second), T_100MS (0.1 second), T_SEC (1.0 second), and T_MIN (1 minute).

The following timing diagram represents the on/off time duration of these contacts

**Figure 58**




These time-tick contacts represent specific locations in %S memory.

# Chapter 10: Instruction Set Reference

This section is a reference to the functions in the VersaMax® PLC Instruction Set:

| | |
|---|---|
| **Bit Operation Functions**<br>Logical AND, Logical OR<br>Exclusive OR, Logical Invert (NOT)<br>Shift Right/Shift Left<br>Rotate Right/Rotate Left<br>Bit Test<br>Bit Set, Bit Clear<br>Masked Compare<br>Bit Position<br>Bit Sequencer | **Math and Numerical Functions**<br>Add, Subtract, Multiply, Divide<br>Modulo Division<br>Scaling Square Root<br>Trigonometric Functions<br>Logarithmic/Exponential Functions<br>Convert Radians / Degrees |
| **Control Functions**<br>Do I/O<br>Call End<br>Comment<br>Master Control Relay<br>Drum Sequencer<br>Service Request (see chapter 11)<br>PID (see chapter 14) | **Relational Functions**<br>Equal<br>Not Equal<br>Greater Than<br>Less Than<br>Greater or Equal<br>Less or Equal<br>Range |
| **Data Move Functions**<br>Move<br>Block Move<br>Block Clear<br>Shift Register<br>Communication Request | **Relay Functions**<br>Contacts, Coils<br>Fault and No Fault Contacts<br>Alarm Contacts<br>**Table Functions**<br>Array Move<br>Search |
| **Data Type Conversion Functions**<br>Convert to BCD-4<br>Convert to Signed Integer<br>Convert to Double Precision Signed Integer<br>Convert to Real<br>Convert Real to Word<br>Truncate Real Number<br>PID (see chapter 14) | **Timer and Counter Functions**<br>Time-tick Contacts<br>On Delay Stopwatch Timer<br>On Delay Timer<br>Off Delay Timer<br>Up Counter<br>Down Counter |

# 10.1 Bit Operation Functions

The Bit Operation functions perform comparison, logical, and move operations on bit strings. The Bit Operation functions are:

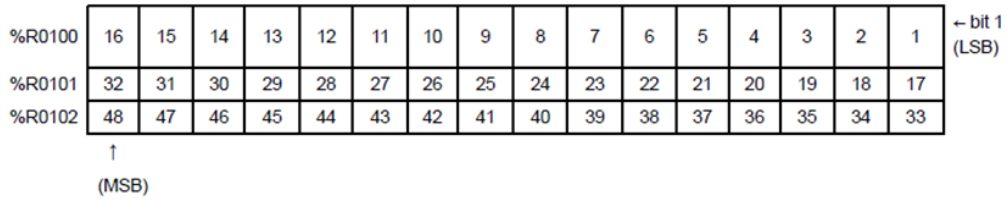
- Logical AND
- Logical OR
- Exclusive OR
- Logical Invert (NOT)
- Shift Right/Shift Left
- Rotate Right/Rotate Left
- Bit Test
- Bit Set, Bit Clear
- Masked Compare
- Bit Position
- Bit Sequencer

## 10.1.1 Data Lengths for the Bit Operation Functions

The Logical AND, OR, XOR, and NOT (Invert) functions operate on a single word of data. The other Bit Operation functions may operate on up to 256 words.

All Bit Operation functions require Word-type data. However, they operate on data as a continuous string of bits, with bit 1 of the first word being the Least Significant Bit (LSB). The last bit of the last word is the Most Significant Bit (MSB). For example, if you specified three words of data beginning at reference %R0100, it would be operated on as 48 contiguous bits.

**Figure 59**




Overlapping input and output reference address ranges in multi-word functions is not recommended, it can produce unexpected results.

## 10.1.2 Bit Operation Functions Logical AND, Logical OR

Each scan that power is received, a Logical AND or Logical OR function examines each bit in bit string I1 and the corresponding bit in bit string I2, beginning at the least significant bit in each. A string length of 256 words can be selected.

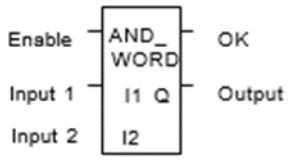
## Logical AND

If both bits examined by the Logical AND function are 1, a 1 is placed in the corresponding location in output string Q. If either or both bits are 0, a 0 is placed in string Q in that location. The Logical AND function can be used to build masks or screens, where only certain bits are passed through (bits opposite a 1 in the mask), and all other bits are set to 0. The Logical AND function can also be used to clear an area of word memory by ANDing the bits with another bit string known to contain all 0s. The I1 and I2 bit strings specified may overlap.

## Logical OR

If either or both bits examined by the Logical OR function is 1, a 1 is placed in the corresponding location in output string Q. If both bits are 0, a 0 is placed in string Q in that location. The Logical OR function can be used to combine strings or to control many outputs with one simple logical structure. The Logical OR function is the equivalent of two relay contacts in parallel multiplied by the number of bits in the string. It can be used to drive indicator lamps directly from input states, or to superimpose blinking conditions on status lights.

**Figure 60**



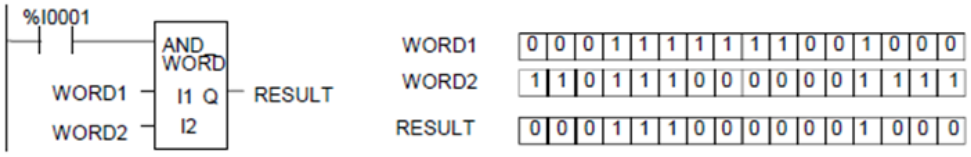

## Parameters of the Logical AND and Logical OR Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| I1 | I, Q, M, T, S, G, R, AI, AQ, constant | Constant or reference for the first word of the first string. |
| I2 | I, Q, M, T, S, G, R, AI, AQ, constant | Constant or reference for the first word of the second string. |
| ok | flow, none | The OK output is energized whenever enable is energized. |
| Q | I, Q, M, T, SA, SB, SC (not S), G, R, AI, AQ | Output Q contains the result of the operation. |

## Example of the Logical AND Function

In the example, when input %I0001 is set, the 16-bit strings represented by nicknames WORD1 and WORD2 are examined. The results of the Logical AND are placed in output string RESULT.

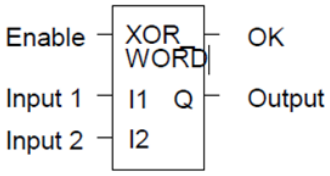
**Figure 61**




## 10.1.3 Bit Operation Functions Exclusive OR

The Exclusive OR function compares each bit in bit string I1 with the corresponding bit in string I2. If the bits are different, a 1 is placed in the corresponding position in the output bit string.

**Figure 62**




I1 and the corresponding bit in string I2, beginning at the least significant bit in each. For each two bits examined, if only one is 1, then a 1 is placed in the corresponding location in bit string Q. The Exclusive OR function passes power flow to the right whenever power is received.

If string I2 and output string Q begin at the same reference, a 1 placed in string I1 will cause the corresponding bit in string I2 to alternate between 0 and 1, changing state with each scan as long as power is received. Longer cycles can be programmed by pulsing the power flow to the function at twice the desired rate of flashing; the power flow pulse should be one scan long (one-shot type coil or self-resetting timer).

The Exclusive OR function is useful for quickly comparing two bit strings, or to blink a group of bits at the rate of one ON state per two scans.

### Parameters of the Exclusive OR Function

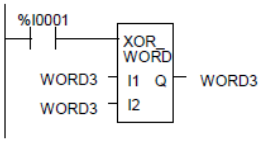
| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| I1 | I, Q, M, T, S, G, R, AI, AQ, constant | Constant or reference for the first word to be XORed. |
| I2 | I, Q, M, T, S, G, R, AI, AQ, constant | Constant or reference for the second word to be XORed. |
| ok | flow, none | The OK output is energized whenever enable is energized. |
| Q | I, Q, M, T, SA, SB, SC (not S), G, R, AI, AQ | Output Q contains the result of the operation. |

## 10.1.4 Bit Operation Functions Exclusive OR

### Example

In the example, whenever %I0001 is set, the bit string represented by the nickname WORD3 is cleared (set to all zeros).

**Figure 63**




| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I1 (WORD3) | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| I2 (WORD3) | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

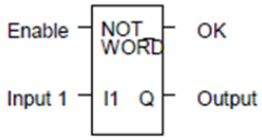
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q (WORD3) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 10.1.5 Bit Operation Functions Logical Invert (NOT)

The Logical Invert (NOT) function sets the state of each bit in the output bit string Q to the opposite of the state of the corresponding bit in bit string I1.

All bits are altered on each scan that power is received, making output string Q the logical complement of I1. The function passes power flow to the right whenever power is received. A length of 256 words can be selected.

**Figure 64**



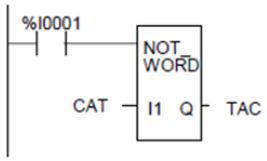
## Parameters of the Logical Invert Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| I1 | I, Q, M, T, S, G, R, AI, AQ, constant | Constant or reference for the word to be negated. |
| ok | flow, none | The OK output is energized whenever enable is energized. |
| Q | I, Q, M, T, SA, SB, SC (not S), G, R, AI, AQ | Output Q contains the result of the operation. |

## Example

In the example, whenever input %I0001 is set, the bit string represented by the nickname TAC is set to the inverse of bit string CAT.

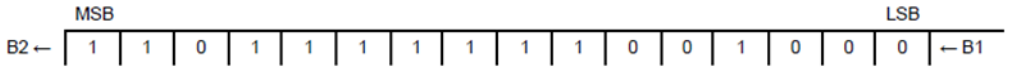
**Figure 65**




# 10.1.6 Bit Operation Functions Shift Bits Right, Shift Bits Left

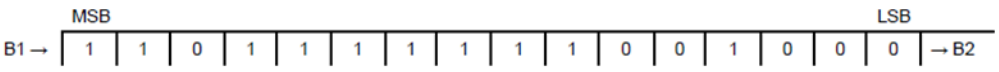
The Shift Left function shifts all the bits in a word or group of words to the left by a specified number of places. When the shift occurs, the specified number of bits is shifted out of the output string to the left. As bits are shifted out of the high end of the string, the same number of bits is shifted in at the low end.

**Figure 66**



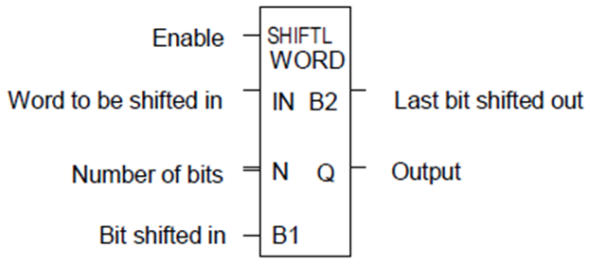

The Shift Right function is used to shift all the bits in a word or group of words a specified number of places to the right. When the shift occurs, the specified number of bits is shifted out of the output string to the right. As bits are shifted out of the low end of the string, the same number of bits is shifted in at the high end.

**Figure 67**




A string length of 1 to 256 words can be selected for either function.

**Figure 68**




If the number of bits to be shifted (N) is greater than the number of bits in the array * 16, the array (Q) is filled with copies of the input bit (B1), and the input bit is copied to the output power flow (B2). If the number of bits to be shifted is zero, then no shifting is performed; the input array is copied into the output array; and input bit (B1) is copied into the power flow.

The bits being shifted into the beginning of the string are specified via input parameter B1. If a length greater than 1 has been specified as the number of bits to be shifted, each of the bits is filled with the same value (0 or 1). This can be:

- The boolean output of another program function.
- All 1s. To do this, use the special reference nickname ALW_ON as a permissive to input B1.
- All 0s. To do this, use the special reference nickname ALW_OFF as a permissive to input B1.

The function passes power flow to the right, unless the number of bits specified to be shifted is zero. Output Q is the shifted copy of the input string. If you want the input string to be shifted, the output parameter Q must use the same memory location as the input parameter IN. The entire shifted string is written on each scan that power is received. Output B2 is the last bit shifted out. For example, if four bits were shifted, B2 would be the fourth bit shifted out.

## Parameters of the Shift Right / Left Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the shift is performed. |
| IN | I, Q, M, T, S, G, R, AI, AQ | IN contains the first word to be shifted. |
| N | I, Q, M, T, G, R, AI, AQ, constant | N contains the number of places (bits) that the array is to be shifted. |
| B1 | flow | B1 contains the bit value to be shifted into the array. |
| B2 | flow, none | B2 contains the bit value of the last bit shifted out of the array. |
| Q | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | Output Q contains the first word of the shifted array. |

## 10.1.7 Bit Operation Functions Rotate Bits Right,Rotate Bits Left

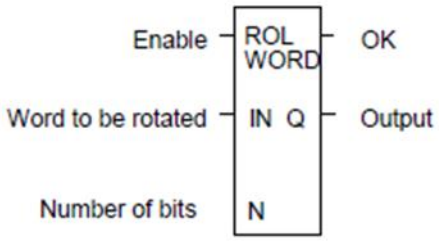
The Rotate Left function rotates all the bits in a string a specified number of places to the left. When rotation occurs, the specified number of bits is rotated out of the input string to the left and back into the string on the right.

The Rotate Right function rotates the bits in the string to the right. When rotation occurs, the specified number of bits is rotated out of the input string to the right and back into the string on the left.

A length of 1 to 256 words can be selected for either function. The number of places to rotate must be more than zero and less than the number of bits in the string.

The Rotate Bits function passes power flow to the right, unless the number of bits specified to be rotated is greater than the total length of the string or is less than zero. The result is placed in output string Q. If you want the input string to be rotated, the output parameter Q must use the same memory location as the input parameter IN. The entire rotated string is written on each scan that power is received.

**Figure 69**



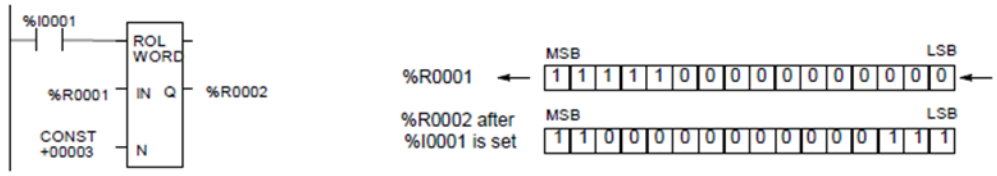

### Parameters of the Rotate Bits Right / Left Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the rotation is performed. |
| IN | I, Q, M, T, S, G, R, AI, AQ | IN contains the first word to be rotated. |
| N | I, Q, M, T, G, R, AI, AQ, constant | N contains the number of places the array is to be rotated. |
| ok | flow, none | The OK output is energized when the rotation is energized and the rotation length is not greater than the array size. |
| Q | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | Output Q contains the first word of the rotated array. |

### Example

In the example, whenever input %I0001 is set, the input bit string in location %R0001 is rotated 3 bits. The result is placed in %R0002. The input bit string %R0001 is not changed by the function. If the same reference is used for IN and Q, a rotation will occur in place.

**Figure 70**



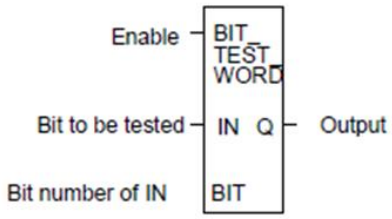

## 10.1.8 Bit Operation Functions Bit Test

The Bit Test function tests a bit within a bit string to determine whether that bit is currently 1 or 0. The result of the test is placed in output Q.

Each sweep power is received, the Bit Test function sets its output Q to the same state as the specified bit. If a register rather than a constant is used to specify the bit number, the same function block can test different bits on successive sweeps. If the value of BIT is outside the range (1 < BIT < (16 * length) ), then Q is set OFF.

A string length of 1 to 256 words can be selected.

**Figure 71**



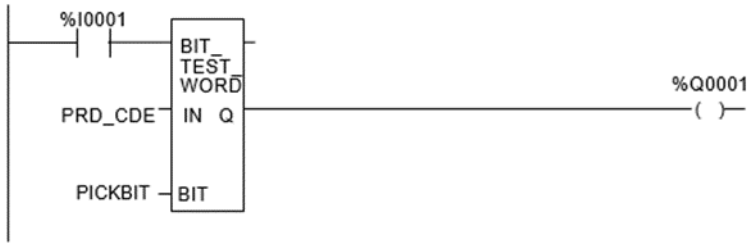

### Parameters of the Bit Test Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the bit test is performed. |
| IN | I, Q, M, T, S, G, R, AI, AQ | IN contains the first word of the data to be operated on. |
| BIT | I, Q, M, T, G, R, AI, AQ, constant | BIT contains the bit number of IN that should be tested. Valid range is (1 < BIT < (16 * length)). |
| Q | flow, none | Output Q is energized if the bit tested was a 1. |

### Example

In the example, whenever input %I0001 is set, the bit at the location contained in reference PICKBIT is tested. The bit is part of string PRD_CDE. If it is 1, output Q passes power flow and the coil %Q0001 is turned on.

**Figure 72**



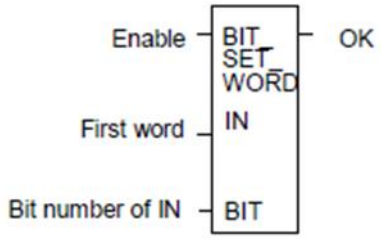

## 10.1.9 Bit Operation Functions Bit Set and Bit Clear

The Bit Set function sets a bit in a bit string to 1. The Bit Clear function sets a bit in a string to 0.

Each sweep that power is received, the function sets the specified bit. If a variable (register) rather than a constant is used to specify the bit number, the same function block can set different bits on successive sweeps.

A string length of 1 to 256 words can be selected. The function passes power flow to the right, unless the value for BIT is outside the range (1 < BIT < (16 * length) ). Then, OK is set OFF.

**Figure 73**



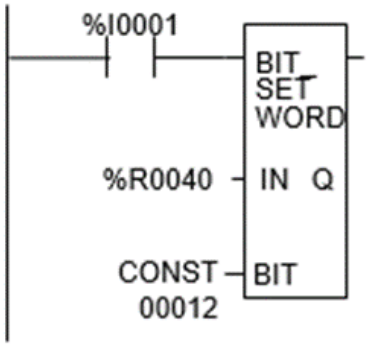

### Parameters of the Bit Set and Bit Clear Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the bit operation is performed. |
| IN | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | IN contains the first word of the data to be operated on. |
| BIT | I, Q, M, T, G, R, AI, AQ, constant | BIT contains the bit number of IN that should be set or cleared. Valid range is (1 < BIT < (16 * length) ). |
| ok | flow, none | The OK output is energized whenever the bit input is valid and enable is energized. |

### Example

In the example, whenever input %I0001 is set, bit 12 of the string beginning at reference %R0040 is set to 1.

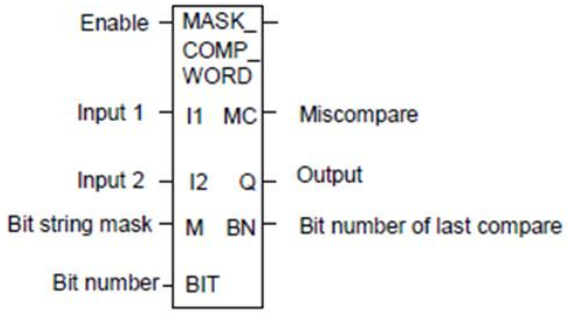
**Figure 74**




## 10.1.10 Bit Operation Functions Masked Compare

The Masked Compare function compares the contents of two separate bit strings. It provides the ability to mask selected bits. Input string 1 might contain the states of outputs such as solenoids or motor starters. Input string 2 might contain their input state feedback, such as limit switches or contacts.

**Figure 75**




When the function receives power flow, it begins comparing the bits in the first string with the corresponding bits in the second string. Comparison continues until a miscompare is found or until the end of the string is reached.

The BIT input stores the bit number where the next comparison should start (a 0 indicates the first bit in the string). The BN output stores the bit number where the last comparison occurred (where a 1 indicates the first bit in the string). Using the same reference for BIT and BN causes the compare to start at the next bit position after a miscompare; or, if all bits compared successfully upon the next invocation of the function block, the compare starts at the beginning.

If you want to start the next comparison at some other location in the string, you can enter different references for BIT and BN. If the value of BIT is a location that is beyond the end of the string, BIT is reset to 0 before starting the next comparison.

## Parameters of the Masked Compare Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | Permissive logic to enable the function. |
| I1 | R, AI, AQ<br>For WORD only: I, Q, M, T, S, G | Reference for the first bit string to be compared. |
| I2 | R, AI, AQ<br>For WORD only: I, Q, M, T, S, G | Reference for the second bit string to be compared. |
| M | R, AI, AQ<br>For WORD only: I, Q, M, T, SS, SB, SC, G | Reference for the bit string mask. |
| BIT | I, Q, M, T, S, G, R, AI, AQ, constant | Reference for the bit number where the next comparison should start. |
| MC | flow, none | User logic to determine if a miscompare has occurred. |
| Q | R, AI, AQ<br>For WORD only: I, Q, M, T, SA, SB, SC, G | Output copy of the mask (M) bit string. |
| BN | I, Q, M, T, S, G, R, AI, AQ | Bit number where the last miscompare occurred. |
| length | Constant | The number of words in the bit string. Max. is 4095 for WORD and 2047 for DWORD. |

## Operation of the Masked Compare

If all corresponding bits in strings I1 and I2 match, the function sets the “miscompare” output MC to 0 and BN to the highest bit number in the input strings. The comparison then stops. On the next invocation of a Masked Compare Word, it is reset to 0. When the two bits currently being compared are not the same, the function checks the correspondingly numbered bit in string M (the mask). If the mask bit is a 1, the comparison continues until it reaches another miscompare or the end of the input strings.

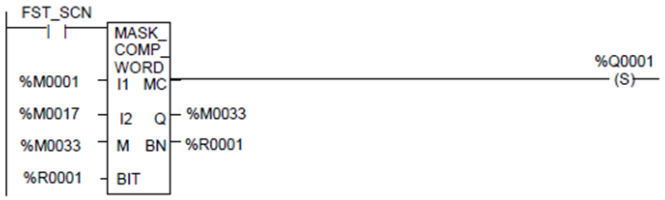
If a miscompare is detected and the corresponding mask bit is a 0, the function does the following:

1. Sets the corresponding mask bit in M to 1.
2. Sets the miscompare (MC) output to 1.
3. Updates the output bit string Q to match the new content of mask string M.
4. Sets the bit number output (BN) to the number of the miscompared bit.
5. Stops the comparison.

## Example

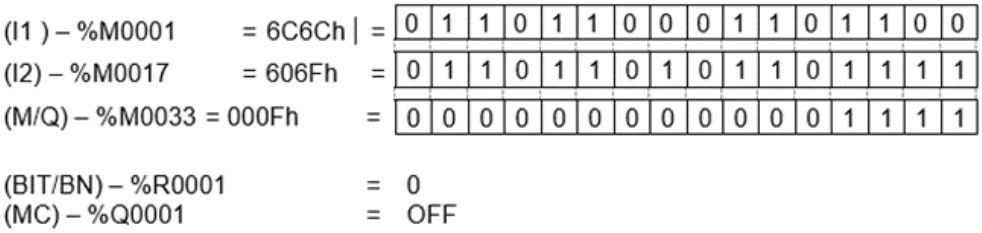
In the example, after first scan the Masked Compare Word function executes. It compares %M0001–16 with %M0017–32. %M0033–48 contain the mask. The value in %R0001 determines the bit position in the two input strings where the comparison starts.

**Figure 76**



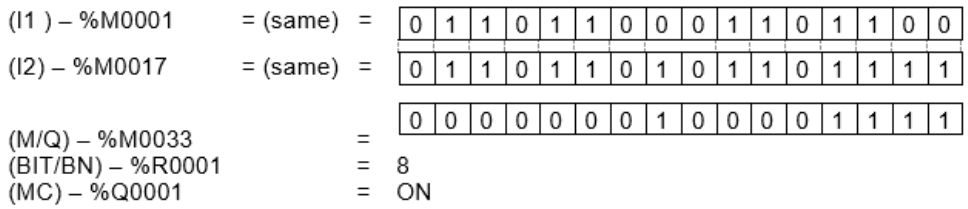

Before the function block is executed, the contents of the above references are:

**Figure 77**




The contents of these references after the function block is executed are:

**Figure 78**




In this example, contact %T1 and coil %M100 force one and only one execution; otherwise the function would repeat with possibly unexpected results.

## 10.1.11 Bit Operation Functions Bit Position

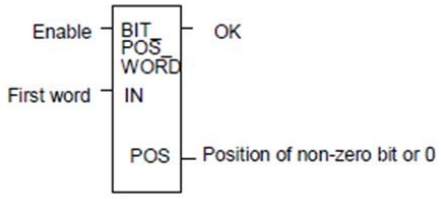
The Bit Position function locates a bit set to 1 in a bit string.

Each sweep that power is received, the function scans the bit string starting at IN. When the function stops scanning, either a bit equal to 1 has been found or the entire length of the string has been scanned.

POS is set to the position within the bit string of the first non-zero bit; POS is set to zero if no non-zero bit is found.

A string length of 1 to 256 words can be selected. The function passes power flow to the right whenever enable is ON.

**Figure 79**



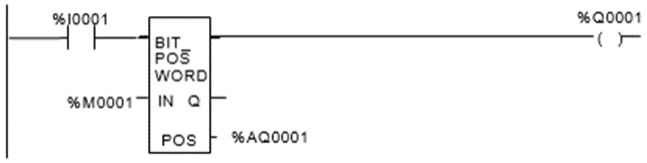

## Parameters for the Bit Position Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, a bit search operation is performed. |
| IN | I, Q, M, T, S, G, R, AI, AQ | IN contains the first word of the data to be operated on. |
| ok | flow, none | The OK output is energized whenever enable is energized. |
| POS | I, Q, M, T, G, R, AI, AQ | The position of the first non-zero bit found, or zero if a non-zero bit is not found. |

## Example

In the example, if %I0001 is set, the bit string starting at %M0001 is searched until a bit equal to 1 is found. Coil %Q0001 is turned on. If a bit equal to 1 is found, its location within the bit string is written to %AQ001. If %I0001 is set, bit %M0001 is 0, and bit %M0002 is 1, then the value written to %AQ001 is 2

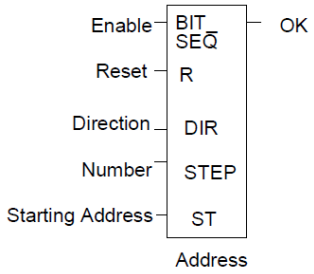
**Figure 80**




# 10.1.12 Bit Operation Functions Bit Sequencer

The Bit Sequencer function performs a bit sequence shift through an array of bits.

**Figure 81**




The operation of the function depends on the previous value of the parameter EN:

| R Current Execution | EN Previous Execution | EN Current Execution | Bit Sequencer Execution |
|---|---|---|---|
| OFF | OFF | OFF | Bit sequencer does not execute. |
| OFF | OFF | ON | Bit sequencer increments/decrements by 1. |
| OFF | ON | OFF | Bit sequencer does not execute. |
| OFF | ON | ON | Bit sequencer does not execute. |
| ON | ON/OFF | ON/OFF | Bit sequencer resets. |

The reset input (R) overrides the enable (EN) and always resets the sequencer. When R is active, the current step number is set to the value passed in via the step number parameter. If no step number is passed in, step is set to 1. All of the bits in the sequencer are set to 0, except for the bit pointed to by the current step, which is set to 1.

When Enable is active and Reset is not active, the bit pointed to by the current step number is cleared. The current step number is incremented or decremented, based on the direction parameter. Then, the bit pointed to by the new step number is set to 1.

The parameter ST is optional. If it is not used, the Bit Seqencer function operates as described above, except that no bits are set or cleared. The function just cycles the current step number through its legal range.

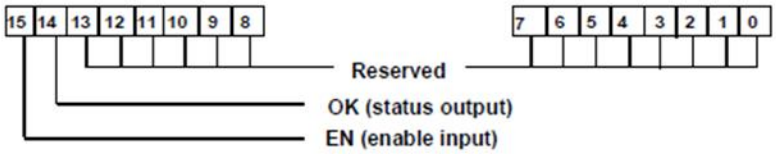
## Memory Required for a Bit Sequencer

Each bit sequencer uses three words (registers) of %R memory to store the information:

| | |
|---|---|
| word 1 | current step number |
| word 2 | length of sequence (in bits) |
| word 3 | control word |

Word 3 (the control word) stores the state of the boolean inputs and outputs of its associated function block, in the following format:

**Figure 82**




## Parameters for the Bit Sequencer Function

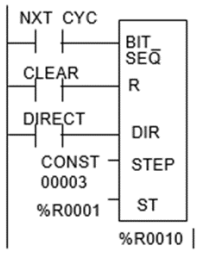
| Input/ Output | Choices | Description |
|---|---|---|
| address | R | Address is the location of the bit sequencer's current step, length, and the last enable and OK status. |
| enable | flow | When the function is enabled, if it was not enabled on the previous sweep and if R is not energized, the bit sequence shift is performed. |
| R | flow | When R is energized, the bit sequencer's step number is set to the value in STEP (default = 1), and the bit sequencer is filled with zeros, except for the current step number bit. |
| DIR | flow | When DIR is energized, the bit sequencer's step number is incremented prior to the shift. Otherwise, it is decremented. |
| STEP | I, Q, M, T, G, R, AI, AQ, constant, none | When R is energized, the step number is set to this value. |
| ST | I, Q, M, T, SA, SB, SC, G, R, AI, AQ, none | ST contains the first word of the bit sequencer. Optional. |
| ok | flow, none | The OK output is energized whenever the function is enabled. |

## Example

In the example, the Bit Sequencer operates on register memory %R0001. Its static data is stored in registers %R0010–12. When CLEAR is active, the sequencer is reset and the current step is set to step number 3. The first 8 bits of %R0001 are set to zero.

When NXT_SEQ is active and CLEAR is not active, the bit for step number 3 is cleared and the bit for step number 2 or 4 (depending on whether DIR is energized) is set.

**Figure 83**




# 10.2 Control Functions

This section describes the control functions, which may be used to limit program execution and to change the way the CPU executes the application program.

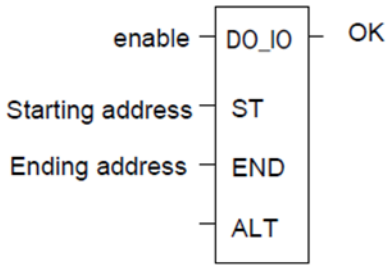
- Service specified I/O: DO IO
- Go to a subroutine block: CALL
- Temporary program end: END
- Execute a group of logic rungs without power flow: MCR
- Go to a specified location in the program: JUMP, LABEL
- Place a text explanation in the program logic: COMMENT
- Provide predefined On/Off patterns to a set of 16 discrete outputs in the manner of a mechanical DRUM SEQUENCER.

The more complex Control Functions; Service Request and the PID algorithms, are described in other chapters of this manual.

## 10.2.1 Control Functions Do I/O

The Do I/O function updates inputs or outputs for one scan while the program is running. The Do I/O function can also be used to update selected I/O during the program in addition to the normal I/O scan. I/O is serviced in increments of entire I/O modules; the PLC adjusts the references, if necessary, while the function executes.

**Figure 84**



Execution of the function continues until all inputs in the selected range have reported or all outputs have been serviced on the I/O modules. Program execution then returns to the next function.

If the range of references includes an option module, all the input data (%I and %AI) or all the output data (%Q and %AQ) for that module will be scanned. The ALT parameter is ignored while scanning intelligent I/O modules or the Ethernet interface.

The function passes power to the right whenever power is received, unless:

- Not all references of the type specified are present within the selected range.
- The CPU is not able to properly handle the temporary list of I/O created by the function.
- The range specified includes modules that are associated with a “Loss of I/O" fault.

## Parameters of the Do I/O Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, a limited input or output scan is performed. |
| ST | I, Q, AI, AQ | The starting address of the I/O to be serviced. |
| END | I, Q, AI, AQ | The ending address of the I/O to be serviced. |
| ALT | I, Q, M, T, G, R, AI, AQ, constant, none | For the input scan, ALT specifies the address to store scanned input point/word values. For the output scan, ALT specifies the address to get output point/word values from to send to the I/O modules. |
| ok | flow, none | OK is energized when the scan completes normally. |

## Do I/O for Inputs

**If input references are specified**, when the function receives power flow, the PLC scans input points from the starting reference (ST) to the END reference. If a reference is specified for ALT, copies of the new input values are placed in memory beginning at that reference, and the real input values are not updated. ALT must be the same size as the reference type scanned. If a discrete reference is used for ST and END, ALT must also be discrete. If no reference is specified for ALT, the real input values are updated. This allows inputs to be scanned one or more times during the program execution portion of the CPU sweep.

## Example Do I/O for Inputs:

In this example, when the function receives power flow, the PLC scans references % I0001-64 and %Q0001 is turned on. Copies of the scanned inputs are placed in internal memory from %M0001-64. Because a reference is specified for ALT, the real inputs are not updated. This allows the current values of inputs to be compared with their values at the beginning of the scan.

**Figure 85**


## Do I/O for Outputs

**If output references are specified**, when the function receives power flow, the PLC writes the latest output values from the starting reference (ST) to the END reference to the output modules. If outputs should be written to the output modules from internal memory other than %Q or %AQ, the beginning reference can be specified for ALT.

## Example Do I/O For Outputs:

In the next example, when the function receives power flow, the PLC writes values from references %R0001-0004 to analog output channels %AQ001-004 and %Q0001 is turned on. Because a reference is entered for ALT, the values at %AQ001-004 are not written to output modules.

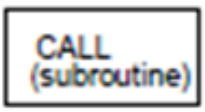
**Figure 86**


If no reference were specified for ALT, the PLC would write values at references % AQ001-004 to analog output channels.

# 10.2.2 Control Functions Call

The Call function causes program execution to go to a specified subroutine block.

**Figure 87**




When the Call function receives power flow, it causes the scan to go immediately to the designated subroutine block and execute it. After the subroutine block execution is complete, control returns to the point in the logic immediately following the Call instruction.

## Example

**Figure 88**

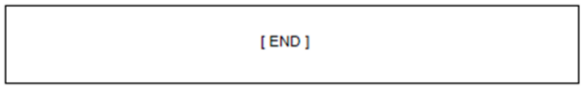

## 10.2.3 Control Functions End of Logic

The End of Logic function provides a temporary end of logic. The program executes from the first rung to the last rung or the End of Logic function, whichever is encountered first. The End of Logic function unconditionally terminates program execution. There can be nothing after the end function in the rung. No logic beyond the End of Logic function is executed, and control is transferred to the beginning of the program for the next sweep.

The End of Logic function is useful for debugging purposes because it prevents any logic which follows from being executed.

The programming software provides an [ END OF PROGRAM LOGIC] marker to indicate the end of program execution. This marker is used if no End of Logic function is programmed in the logic.

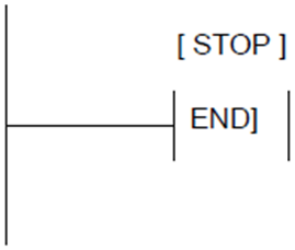
**Figure 89**




### Example

In the example, an End of Logic function is programmed to terminate the end of the current sweep.

**Figure 90**



## 10.2.4 Control Functions Master Control Relay (MCR) / End MCR

All rungs between an active Master Control Relay (MCRN) and its corresponding End Master Control Relay (ENDMCRN) function are executed without power flow to coils.

The ENDMCRN associated with the Master Control Relay is used to resume normal program execution. Unlike Jump functions, Master Control Relays can only move forward; the ENDMCRN must appear after its corresponding Master Control Relay instruction in a program.

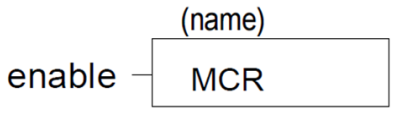
### Nested MCR

A Nested Master Control Relay function can be nested completely within another MCRN/ENDMCRN pair.

There can be multiple Master Control Relay functions with a single ENDMCRN.

The Master Control Relay function has an enable input and a name. This name is used again with the ENDMCRN. The Master Control Relay has no outputs; there can be nothing after it in a rung.

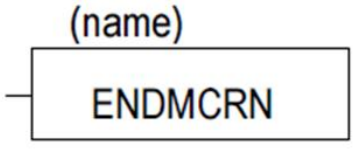
**Figure 91**




With a Master Control Relay, function blocks within the scope of the Master Control Relay are executed without power flow, and coils are turned off.

The ENDMCRN function must be tied to power rail; there can be no logic before it in the rung. The name of the ENDMCRN associates it with the corresponding Master Control Relay(s). The ENDMCRN function has no outputs; there can be nothing after it in a rung.

**Figure 92**

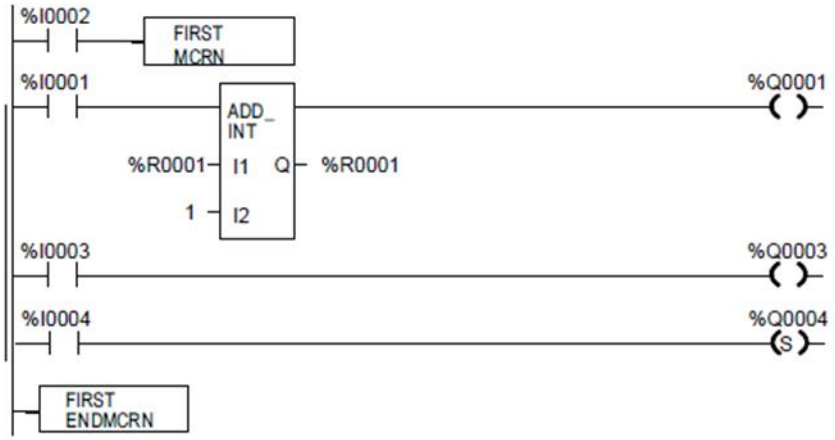



### Example Master Control Relay and ENDMCRN Functions

In the example, when %I0002 is ON, the Master Control Relay is enabled. When the Master Control Relay is enabled–even if %I0001 is ON–the Addition function block is executed without power flow (i.e., it does not add 1 to %R0001), and %Q0001 is turned OFF.

If %I0003 and %I0004 are ON, %Q0003 is turned OFF and %Q0004 remains ON.

**Figure 93**




## 10.2.5 Control Functions Jump, Label

The Nested Jump instruction causes a portion of the program logic to be bypassed. Program execution continues at the Label specified. When the Jump is active, all coils within its scope are left at their previous states. This includes coils associated with timers, counters, latches, and relays.

The Nested Jump instruction has the form ——>>LABEL01, where LABEL01 is the name of the corresponding nested Label instruction.

A nested Jump can be placed anywhere in a program.

There can be multiple nested Jump instructions corresponding to a single nested Label. Nested Jumps can be either forward or backward Jumps.

There can be nothing after the Jump instruction in the rung. Power flow jumps directly from the instruction to the rung with the named label.

**⚠CAUTION**

To avoid creating an endless loop with forward and backward Jump instructions, a backward Jump must contain a way to make it conditional.

### Label

The Label instruction is the target of a Jump. Use the Label instruction to resume normal program execution. There can be only one Label with a particular name in a program.

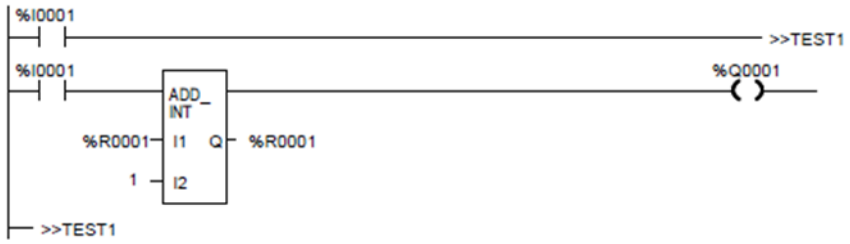
The Label instruction has no inputs and no outputs; there can be nothing either before or after a Label in a rung.

### Example Jump and Label Instructions

In the example, whenever Jump TEST1 is active, power flow is transferred to Label TEST1.

With a Jump, any function blocks between the Jump and the Label are not executed, and coils are not affected. In the example, when %I0002 is ON, the Jump is taken. Since the logic between the Jump and the Label is skipped, %Q0001 is unaffected (if it was ON, it remains ON; if it was OFF, it remains OFF).

**Figure 94**




## 10.2.6 Control Functions Comment

The Comment function is used to enter a comment (rung explanation) in the program. A comment can have up to 2048 characters of text. Longer text can be included in printouts using an annotation text file.

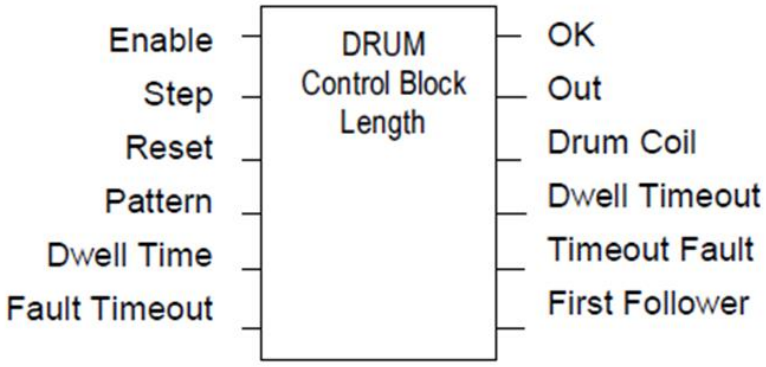
It is represented in the ladder logic like this:

**Figure 95**


## 10.2.7 Control Functions Drum Sequencer

The Drum Sequencer function is a program instruction that operates like a mechanical drum sequencer. The Drum Sequencer steps through a set of potential output bit patterns and selects one based on inputs to the function block. The selected value is copied to a group of 16 discrete output references.

**Figure 96**



Power flow to the Enable input causes the Drum Sequencer to copy the content of a selected reference to the Out reference.

Power flow to the Reset input or to the Step input selects the reference to be copied.

The Control Block input is the beginning reference for the Drum Sequencer function's parameter block, which includes information used by the function.

## Parameters of the Drum Sequencer Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | The Enable input controls execution of the function. |
| Step | flow | The Step input can be used to go one step forward in the sequence. When the Enable input receives power flow and the Step input makes an Off to On transition, the Drum Sequencer moves one step. When Reset is active, the function ignores the Step input. |
| Reset | flow | The Reset input can be used to select a specific step in the sequence. When Enable and Reset both receive power flow, the function copies the Preset Step value in the Control Block to the Active Step reference, also in the Control Block. Then the function block copies the value in the Preset Step reference to the Out-reference bits. When Reset is active, the function ignores the Step input. |
| Pattern | R, AI, AQ | The starting address of an array of words, each representing one step of the Drum Sequencer. The value of each word represents the desired combination of outputs for a particular value of Active Step. The number of elements in the array is equal to the length input. |
| Dwell Time | R, AI, AQ, none | This optional input array of words has one element for each element in the Pattern array. Each value in the array represents the dwell time for the corresponding step of the Drum Sequencer in 0.1 second units. When the dwell time expires for a given step the Dwell Timeout bit is set. If a Dwell Time is specified, the drum cannot sequence into its next step until the Dwell Time has expired. |
| Fault Timeout | R, AI, AQ, none | This optional input array of words has one element for each element in the Pattern array. Each value in the array represents the fault timeout for the corresponding step of the Drum Sequencer in 0.1 second units. When the fault timeout has expired the Fault Timeout bit is set. |
| Control Block | R | The beginning reference address of the function's parameter block. The length of the Control Block is 5 words. A more complete description of what is contained within this block is listed below. |
| Length | CONST | Value between 1 and 128 that specifies the number of steps. |
| ok | flow, none | OK is energized if Enable is On and no error condition is detected. If Enable is Off, this output will always be Off. |
| OUT | I, Q, M, T, G, R, AI, AQ | A word of memory containing the element of the Pattern Array that corresponds to the current Active Step. |

| Input/ Output | Choices | Description |
|---|---|---|
| Drum Coil | I, Q, M, T, G, none | This optional bit reference is set whenever the function block is enabled, and Active Step is not equal to Preset Step. |
| Dwell Timeout | I, Q, M, T, G, none | This optional bit reference is set if the dwell time for the current step has expired. |
| Timeout Fault | I, Q, M, T, G, none | This optional bit reference is set if the drum has been in a particular step longer than the step's specified Fault Timeout. |
| First Follower | I, Q, M, T, G, none | This optional array of bits has one element for each step of the Drum Sequencer. No more than one bit in the array is on at any time and that bit corresponds to the value of the Active Step |

## Parameter Block for the Drum Sequencer Function

The parameter block (control block) for the Drum Sequencer function contains information needed to operate the Drum Sequencer.

| | |
|---|---|
| address | Active Step |
| address + 1 | Preset Step |
| address + 2 | Step Control |
| address + 3 | Timer Control |

**Active Step** The active step value specifies the element in the Pattern array to copy to the output memory location. This is used as the array index into the Pattern, Dwell Time, Fault Timeout, and First Follower arrays.

**Preset Step** A word input that is copied to the Active Step output when the Reset is On.

**Step Control** A word that is used to detect Off to On transitions on both the Step input and the Enable input. The Step Control word is reserved for use by the function block and must not be written to.

**Timer Control** Two words of data that hold values needed to run the timer. These values are reserved for use by the function block and must not be written to.

## Notes on Using the Drum Sequencer Function

1. The Dwell Timeout Output bit is cleared the first time the drum is in a new step. This is true:

- Whether the drum is introduced to a new step by changing the Active Step or by using the Step Input.
- Regardless of the Dwell Time Array value associated with the step (even if it is 0).
- During the first sweep the Active Step is initialized.

2. The Active and Preset Step of the Drum Sequencer's control block must be initialized for the Drum Sequencer to work or to pass power flow. Even if the Active Step is in the correct range (between 1 and length of the Pattern array) and the Preset Step is not used, the drum will not function if the Preset Step is not in the proper range.

# 10.3 Data Move Functions

The Data Move functions of the Instruction Set provide basic data move capabilities.

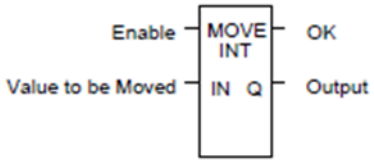
- Move Data. This function copies data as individual bits, so the new location does not have to be the same data type.
- Block Move. This function places constants into seven specified memory locations.
- Block Clear. This function fills an area of memory with zeros.
- Shift Register. This function shifts one or more data words or data bits from a reference location into a specified area of memory. Data already in the area is shifted out.
- Communication Request (COMMREQ). This important function allows the CPU to communicate with intelligent modules in the system, for example, communications modules. The basic format of the COMMREQ function is shown in this chapter. The detailed parameters needed to program specific communications tasks are provided in the documentation for each module.

## 10.3.1 Data Move Functions Move Data

The Move function copies data as individual bits from one location to another. Because the data is copied in bit format, the new location does not need to be the same data type as the original.

When the Move function receives power flow, it copies data from input parameter IN to output parameter Q as bits. If data is moved from one location in discrete memory to another, (for example, from %I memory to %T memory), the transition information associated with the discrete memory elements is updated to indicate whether or not the Move operation caused any discrete memory elements to change state. Data at the input parameter does not change unless there is an overlap in the source and destination.

**Figure 97**



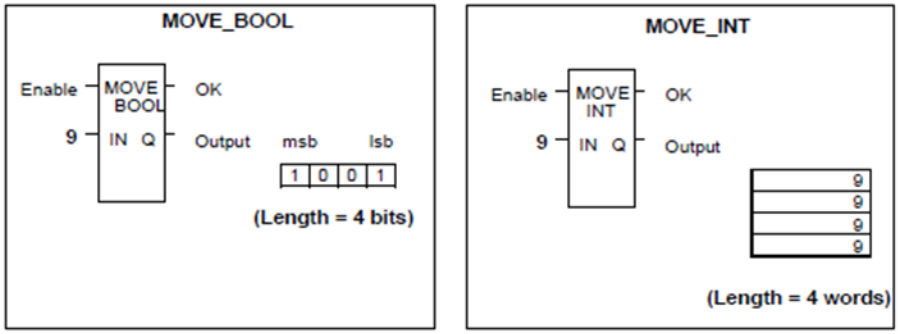

Note that if an array of Bit-type data specified in the Q parameter does not include all the bits in a byte, the transition bits associated with that byte (which are not in the array) are cleared when the Move function receives power flow.

The input IN can be either a reference for the data to be moved or a constant. If a constant is specified, then the constant value is placed in the location specified by the output

reference. For example, if a constant value of 4 is specified for IN, then 4 is placed in the memory location specified by Q. If the length is greater than 1 and a constant is specified, then the constant is placed in the memory location specified by Q and the locations following, up to the length specified. Do not allow overlapping of IN and Q parameters.

The result of the Move depends on the data type selected for the function, as shown below. For example, if the constant value 9 is specified for IN and the length is 4, then 9 is placed in the bit memory location specified by Q and the three locations following:

**Figure 98**




The function passes power to the right whenever power is received.

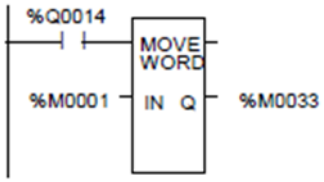
## Parameters for the Move Data Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the move is performed. |
| Length | | The number of bits, words, or double words of data to be copied. This is the length of IN. Length must be from 1 to 256 for all types except BIT. If IN is a constant and Q is type BIT, the length must be between 1 and 16. If IN is type Bit, the length must be between 1 and 256 bits. |
| IN | I, Q, M, T, G, R, AI, AQ, constant<br>For bit or word data only: S<br>For real data: R, AI, AQ | IN contains the value to be moved. For MOVE_BOOL, any discrete reference may be used; it does not need to be byte aligned. However, 16 bits, beginning with the reference address specified, are displayed online. |
| ok | flow, none | The OK output is energized whenever the function is enabled. |
| Q | I, Q, M, T, G, R, AI, AQ, For bit/ word data: SA, SB, SC<br>For real data: R, AI, AQ | When the move is performed, the value at IN is written to Q.<br>For MOVE_BOOL, any discrete reference may be used; it does not need to be byte aligned. However, 16 bits, beginning with the reference address specified, are displayed online. |

## Example 1

When enabling input %Q0014 is ON, 48 bits are moved from memory location %M0001 to memory location %M0033. (%M0001 and %M0003 are defined as WORD types if length 3.)

**Figure 99**




Even though the destination overlaps the source for 16 bits, the move is done correctly.

**Before using the Move function:**

INPUT (%M0001 through %M0048)

**Figure 100**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| %M0016 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| %M0032 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| %M0048 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**After using the Move function:**

INPUT (%M0033 through %M0080)

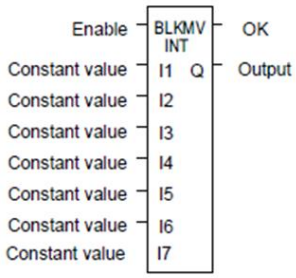
**Figure 101**

| | | | | | | | | | | | | | | | | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| %M0048 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| %M0064 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| %M0080 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | | | | | | | | | | | | | | | 33 |

## 10.3.2 Data Move Functions Block Move

The Block Move function copies a block of seven constants to a specified location. When the Block Move function receives power flow, it copies the constant values into consecutive locations beginning at the destination specified in output Q. The function passes power to the right whenever power is received.

**Figure 102**



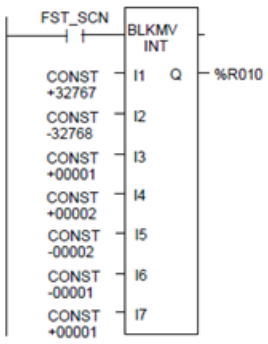

### Parameters of the Block Move Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the block move is performed. |
| I1 to I7 | constant | I1 through I7 contain seven constant values. |
| ok | flow, none | The OK output is energized whenever the function is enabled. |
| Q | I, Q, M, T, G, R, AI, AQ<br>For Word data: SA, SB, SC For Real data: R, AI, AQ | Output Q contains the first element of the moved array. I1 is moved to Q. |

### Example

In the example, when the enabling input represented by the nickname FST_SCN is ON, the Block Move function copies the input constants into memory locations %R0010–16.

**Figure 103**

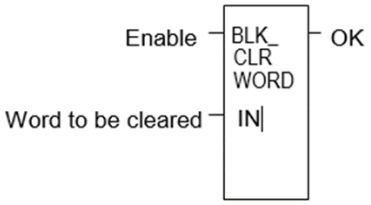


## 10.3.3 Data Move Functions Block Clear

The Block Clear function fills a specified block of data with zeros. When the function receives power flow, it writes zeros into the memory location beginning at the reference specified by IN. When the data to be cleared is from discrete memory (%I, %Q, %M, %G, or %T), the transition information associated with the references is also cleared.

The function passes power to the right whenever power is received.

**Figure 104**



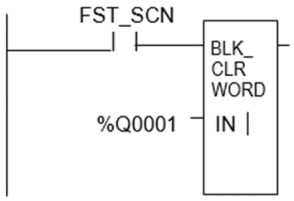

### Parameters of the Block Clear Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the array is cleared. |
| IN | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | IN contains the first word of the array to be cleared. The length of IN must be between 1 and 256 words. |
| Length | | The number of words that will be cleared. This is the length of IN. |
| ok | flow, none | The OK output is energized whenever the function is enabled. |

### Example

In the example, at powerup, 32 words of %Q memory (512 points) beginning at %Q0001 are filled with zeros. %Q is defined as WORD of length 32.

**Figure 105**

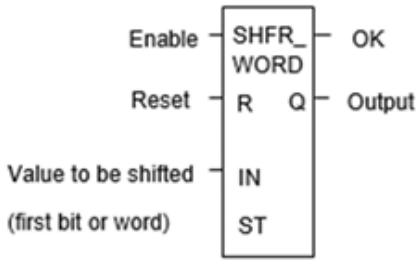


## 10.3.4 Data Move Functions Shift Register

The Shift Register function shifts one or more data words or data bits from a reference location into a specified area of memory. For example, one word might be shifted into an area of memory with a specified length of five words. As a result of this shift, another word of data would be shifted out of the end of the memory area.

The reset input (R) takes precedence over the function enable input. When the reset is active, all references beginning at the shift register (ST) up to the length specified for LEN, are filled with zeros. If the function receives power flow and reset is not active, each bit or word of the shift register is moved to the next highest reference. The last element in the shift register is shifted into Q. The highest reference of the shift register element of IN is shifted into the vacated element starting at ST. The contents of the shift register are accessible throughout the program because they are overlaid on absolute locations in logic addressable memory.

**Figure 106**




For VersaMax CPUs, when this function is used to rotate a bit sequence around a range of discrete references, separate references must be used for ST and Q, and additional logic must be provided to copy the output word from the Q reference back to the ST reference. This function may be used, for example, to control the repetition of a fixed sequence of actions. In this example, each of the discrete references in the range of the SHFR_BIT instruction would be used as the permissive contact for one of the repeated actions.

### Parameters of the Shift Register Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When enable is energized and R is not, the shift is performed. |
| Length | 1 to 256 bits / words. | The length of the shift registers in bits or words. Length is defined as the length of IN. |
| R | flow | When R is energized, the shift register located at ST is filled with zeros. |
| IN | I, Q, M, T, S, G, R, AI, AQ, constant | IN contains the value to be shifted into the first bit or word of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. |

| Input/ Output | Choices | Description |
|---|---|---|
| ST | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | ST contains the first bit or word of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. |
| ok | flow, none | OK is energized whenever the function is enabled, and R is not enabled. |
| Q | I, Q, M, T, SA, SB, SC, G, R, AI, AQ | Output Q contains the bit or word shifted out of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. |

**⚠CAUTION**

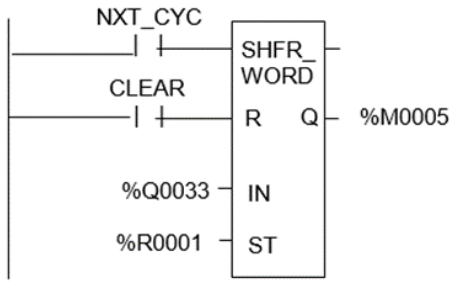
The use of overlapping input and output reference address ranges in multi-word functions is not recommended; it may produce unexpected results.

## Example 1:

In the example, the shift register operates on register memory locations %R0001 through %R0100. (%R0001 is defined as type Word of length 100). When the reset reference CLEAR is active, the Shift Register words are set to zero.

When the NXT_CYC reference is active and CLEAR is not active, the word from output status table location %Q0033 is shifted into the Shift Register at %R0001. The word shifted out of the Shift Register from %R0100 is stored in output %M0005.

**Figure 107**

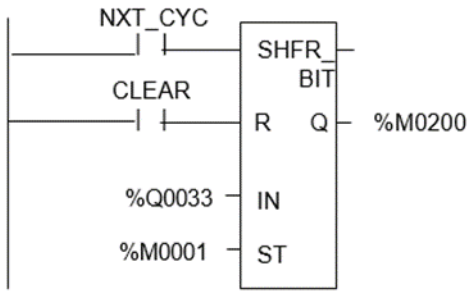



## Example 2:

In this example, the Shift Register operates on memory locations %M0001 through %M0100. (%M0001 is defined as type Boolean of length 100). When the reset reference CLEAR is active, the Shift Register function fills %M0001 through %M0100 with zeros.

When NXT_CYC is active and CLEAR is not, the Shift Register function shifts the data in %M0001 to %M0100 down by one bit. The bit in %Q0033 is shifted into %M0001 while the bit shifted out of %M0100 is written to %M0200.

**Figure 108**

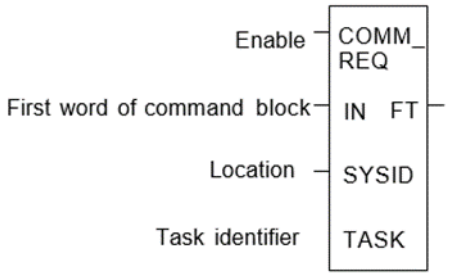



## 10.3.5 Data Move Functions Communication Request

The Communication Request (COMMREQ) function communicates with an intelligent module. Many types of COMM REQs have been defined. The information below describes only the basic format of the function.

When the function receives power flow, a command block of data is sent to the specified module. After sending the COMMREQ, the program can either suspend execution and wait for a reply for a maximum waiting period specified in the command or resume immediately.

**Figure 109**




### Parameters of the COMMREQ Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is energized, the communications request is performed. |
| IN | R, AI, AQ | IN contains the first word of the command block. |
| SYSID | I, Q, M, T, G, R, AI, AQ, constant | SYSID contains the rack number (most significant byte) and slot number (least significant byte) of the target device. |
| TASK | R AI, AQ, constant | TASK contains the task ID of the process on the target device. |

| Input/ Output | Choices | Description |
|---|---|---|
| FT | flow, none | FT is energized if an error is detected processing the COMM REQ:<br>1. The specified target address is not present (SYSID).<br>2. The specified task is not valid for the device (TASK).<br>3. The data length is 0.<br>4. The device's status pointer address (in the command block) does not exist. |

## Command Block for the COMMREQ Function

The Command Block starts at the reference specified in COMMREQ parameter IN. The length of the Command Block depends on the amount of data sent to the device.

The Command Block contains the data to be communicated to the other device, plus information related to the execution of the COMM REQ. The Command Block has the following structure:

| | |
|---|---|
| address | Length (in words) |
| address + 1 | Wait/No Wait Flag |
| address + 2 | Status Pointer Memory |
| address + 3 | Status Pointer Offset |
| address + 4 | Idle Timeout Value |
| address + 5 | Maximum Communication Time |
| address + 6 to address + 133 | Data Block |

## Example

In the example, when enabling input %M0020 is ON, a Command Block starting at %R0016 is sent to communications task 1 in the device located at rack 1, slot 2 of the PLC. If an error occurs processing the COMMREQ, %Q0100 is set.

**Figure 110**

# 10.4 Data Type Conversion Functions

The Data Type Conversion functions are used to change a data item from one number type to another. Many programming instructions, such as math functions, must be used with data of one type.

- Convert data to BDC-4
- Convert data to signed integer
- Convert data to double-precision integer
- Convert data to Real
- Convert data to Word
- Round a Real number toward zero (TRUN)

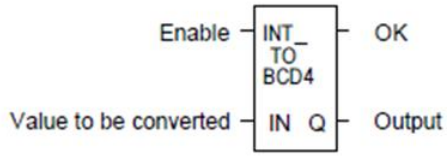
## 10.4.1 Data Type Conversion Functions Convert Signed Integer Data to BCD-4

The Convert to BCD-4 function outputs the four-digit BCD equivalent of signed integer data. The original data is not changed by this function.

Data can be converted to BCD format to drive BCD-encoded LED displays or presets to external devices such as high-speed counters.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is outside the range 0 to 9999.

**Figure 111**



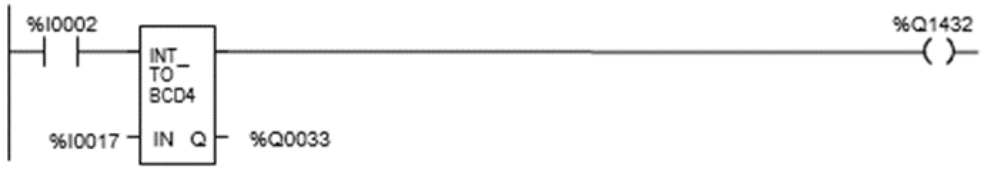

### Parameters of the Convert to BCD-4 Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | I, Q, M, T, G, R, AI, AQ, constant | IN contains a reference for the integer value to be converted to BCD-4. |
| OK | flow, none | The OK output is energized when the function is performed without error. |
| Q | I, Q, M, T, G, R, AI, AQ | Output Q contains the BCD-4 form of the original value in IN. |

## Example

In the example, whenever input %I0002 is set and no errors exist, the integer at input location %I0017 through %I0032 is converted to four BCD digits, and the result is stored in memory locations %Q0033 through %Q0048. Coil %Q1432 is used to check for successful conversion.

**Figure 112**

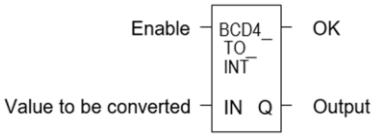



## 10.4.2 Data Type Conversion Functions Convert to Signed Integer

The Convert to Signed Integer function outputs the integer equivalent of BCD-4 or Real data. The original data is not changed by this function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function always passes power flow when power is received, unless the data is out of range.

**Figure 113**



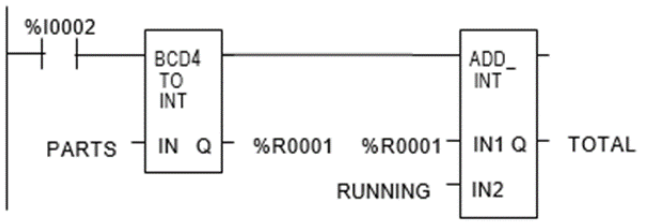

## Parameters of the Convert to Signed Integer Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | For BCD-4: I, Q, M, T, G, R, AI, AQ, constant<br>For REAL: R, AI, AQ | IN contains a reference for the BCD-4, REAL, or Constant value to be converted to integer. |
| ok | flow, none | The OK output is energized whenever enable is energized, unless the data is out of range or NaN (Not a Number). |
| Q | I, Q, M, T, G, R, AI, AQ | Output Q contains the integer form of the original value in IN. |

## Example

In the example, whenever input %I0002 is set, the BCD-4 value in PARTS is converted to a signed integer and passed to the Addition function, where it is added to the signed integer value represented by the reference RUNNING. The sum is output by the Addition function to the reference TOTAL.

**Figure 114**

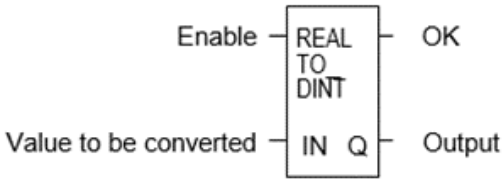



## 10.4.3 Data Type Conversion Functions Convert to Double Precision Signed Integer

The Convert to Double Precision Signed Integer function outputs the double precision signed integer equivalent of real data. The original data is not changed by this function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function always passes power flow when power is received, unless the real value is out of range.

**Figure 115**




Note that loss of precision can occur when converting from Real-type data to Double-Precision Integer, because Real data has 24 significant bits.

## Parameters of the Convert to Double Precision Signed Integer Function

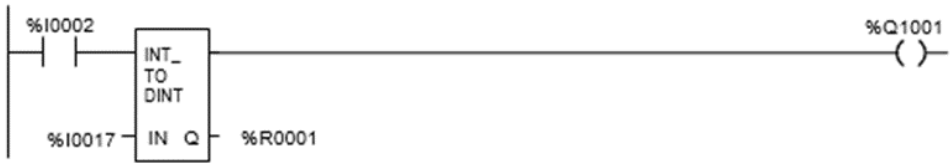
| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | I, Q, M, T, G, R, AI, AQ, constant | Constant or reference for the value to be converted |
| ok | flow, none | OK is energized whenever enable is energized, unless the real value is out of range. |

| Input/ Output | Choices | Description |
|---|---|---|
| Q | R, AI, AQ | Reference that contains the double precision signed integer form of the original value. |

## Example

In the example, whenever input %I0002 is set, the integer value at input location %I0017 is converted to a double precision signed integer and the result is placed in location %R0001. The output %Q1001 is set whenever the function executes successfully.

**Figure 116**



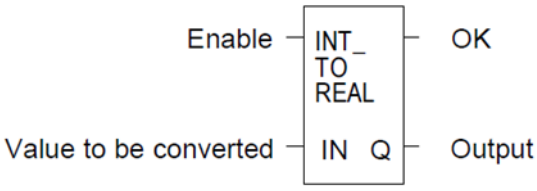

## 10.4.4 Data Type Conversion Functions Convert to Real Data

The Convert to Real function outputs the real value equivalent of the input data. The original data is not changed by this function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is out of range.

Note that loss of precision can occur when converting from Double-Precision Integer to Real data, because since the number of significant bits is reduced to 24.

**Figure 117**




## Parameters of the Convert to Real Function

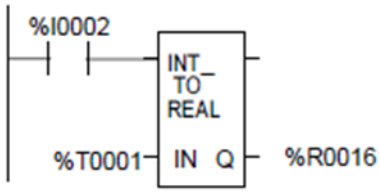
| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | R, AI, AQ, constant<br>For INT only: I, Q, M, T, G | IN contains a reference for the integer value to be converted to Real. |

| Input/ Output | Choices | Description |
|---|---|---|
| ok | flow, none | OK is energized when the function is performed without error. |
| Q | R, AI, AQ | The Real form of the original value in IN. |

## Example

In the example, the integer value of input IN is 678. The result value placed in %T0016 is 678.000.

**Figure 118**

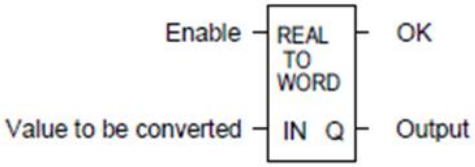



## 10.4.5 Data Type Conversion Functions Convert Real Data to Word Data

The Convert to Word function outputs the Word equivalent of Real data. The original data is not changed by this function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is outside the range 0 to FFFFh.

**Figure 119**

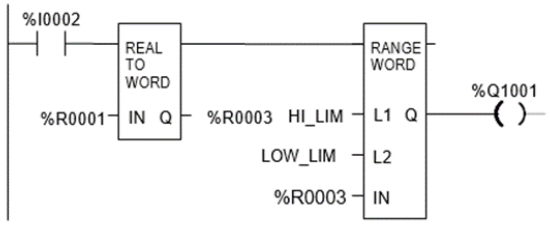


## Parameters of the Convert to Word Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | R, AI, AQ, constant | IN contains a reference for the value to be converted to Word type. |
| ok | flow, none | OK is energized when the function is performed without error. |
| Q | I, Q, M, T, G, R, AI, AQ | Contains the word form of the original value in IN. |

## Example

**Figure 120**

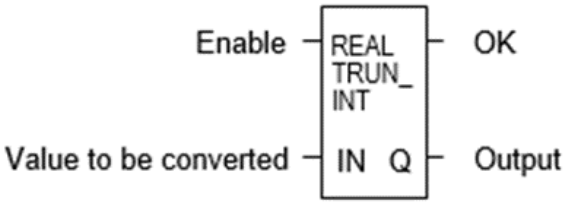



## 10.4.6 Data Type Conversion Functions Truncate Real Number

The Truncate function copies a Real number and rounds the copied number down to an integer or double precision integer. The original data is not changed by this function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is out of range or unless IN is not a number.

**Figure 121**



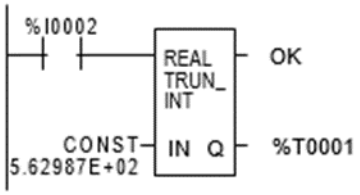
## Parameters of the Truncate Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the conversion is performed. |
| IN | R, AI, AQ, constant | IN contains a reference for the real value to be truncated. |
| ok | flow, none | The OK output is energized when the function is performed without error, unless the value is out of range or IN is NaN. |
| Q | R, AI, AQ<br>For integer only: I, Q, M, T, G | Q contains the truncated INT or DINT value of the original value in IN. |

## Example

In the example, the displayed constant is truncated, and the integer result 562 is placed in %T0001.

**Figure 122**




# 10.5 Math and Numerical Functions

This section describes the Math and Numerical functions of the Instruction Set:

- Standard Math Functions: Addition, Subtraction, Multiplication, Division
- Modulo Division
- Scaling Function
- Square Root
- Trigonometric functions
- Logarithmic/Expontial functions
- Convert to Degrees
- Convert to Radians

### Converting Data for the Math and Numerical Functions

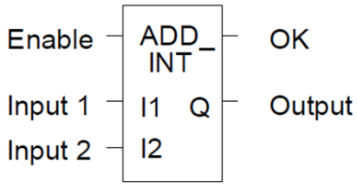
The program may need to include logic to convert data to a different type before using a Math or Numerical function. The description of each function includes information about appropriate data types. The section Data Type Conversion Functions explains how to convert data to a different type.

## 10.5.1 Math and Numerical Functions Add, Subtract,Multiply, Divide

The standard math functions are Addition, Subtraction, Multiplication, and Division. The Division function rounds down; it does not round to the closest integer. (For example, 24 DIV 5 = 4.)

When a math function receives power flow, the operation is performed on input parameters I1 and I2. Parameters I1, I2, and output Q must be the same data type.

**Figure 123**




The math functions pass power if there is no math overflow. If an overflow occurs, the result is the largest value with the proper sign and no power flow.

### Parameters of the Standard Math Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| I1 | All data types: R, AI, AQ, constant<br>INT data type only: I, Q, M, T, G | I1 contains a constant or reference for the first value used in the operation. (I1 is on the left side of the mathematical expression, as in I1 + I2).<br>Range for constants in double-precision signed integer operations is minimum/maximum DINT. |
| I2 | All data types: R, AI, AQ, constant<br>INT data type only: I, Q, M, T, G | I2 contains a constant or reference for the second value used in the operation. (I2 is on the right side of the mathematical expression, as in I1 + I2). Range for constants in double-precision signed integer operations is minimum/maximum DINT. |
| ok | flow, none | The OK output is energized when the function is performed without overflow, unless an invalid operation occurs. |
| Q | All data types: R, AI, AQ<br>INT only: I, Q, M, T, G | Output Q contains the result of the operation. |

## Data Types for Standard Math Functions

Standard math functions operate on these types of data:

| | |
|---|---|
| INT | Signed integer |
| DINT | Double precision signed integer |
| REAL | Floating Point |

The input and output parameter data types must be the same (16 bits or 32 bits).

## Avoiding Overflows

Be careful to avoid overflows when using Multiplication and Division functions.

If you have to convert Integer to Double-Precision Integer values, remember that the CPU uses standard 2's complement with the sign extended to the highest bit of the second word. You must check the sign of the low 16-bit word and extend it into the second 16 bit word. If the most significant bit in a 16-bit INT word is 0 (positive), move a 0 to the second word. If the most significant bit in a 16-bit word is –1 (negative), move a – 1 or hex 0FFFFh to the second word.

Converting from Double-Precision Integer to Integer data is easier, because the low 16-bit word (first register) is the integer portion of a Double-Precision Integer 32-bit word.

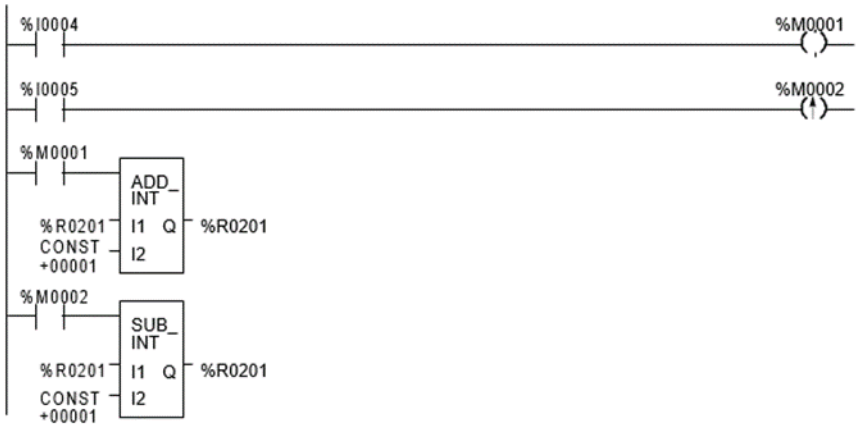
The upper 16 bits or second word should be either a 0 (positive) or –1 (negative) value or the Double-Precision Integer number will be too big to convert to 16 bits.

## Example

This example uses the Addition and Subtraction functions to keep track of the number of parts in a temporary storage area. Each time a part enters the storage area, power flows through relay %I0004 to a positive transition coil with reference %M0001. Relay % M0001 then enables the Addition function, adding the (constant) value 1 to the current total value in %R0201.

Each time a part leaves the storage area, power flows through relay %I0005 to a positive transition coil with reference %M0002. Relay %M0002 then enables the Subtraction function, subtracting the (constant) value 1 from the current total value in %R0201.

**Figure 124**




## 10.5.2 Math and Numerical Functions Modulo Division

The Modulo Division function divides one value by another of the same data type, to obtain the remainder. The sign of the result is always the same as the sign of input parameter I1. The Modulo function operates on these types of data:

| | |
|---|---|
| INT | Signed integer |
| DINT | Double precision signed integer |

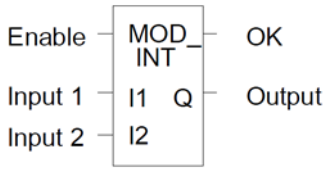
When the function receives power flow, it divides input I1 by input I2. These parameters must be the same data type. Output Q is calculated using the formula:

$$Q = I1-((I1 \text{ DIV } I2) * I2)$$

The division produces an integer. Q is the same data type as inputs I1 and I2.

OK is always ON when the function receives power flow, unless there is an attempt to divide by zero. In that case, it is set OFF.

**Figure 125**



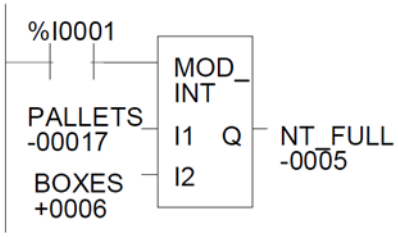
## Parameters of the Modulo Division Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| I1 | All data types: R, AI, AQ, constant<br>INT data type only: I, Q, M, T, G | I1 contains a constant or reference for the value to be divided by I2. Range for constants in double precision signed integer operations is minimum/maximum DINT. |
| I2 | All data types: R, AI, AQ, constant<br>INT data type only: I, Q, M, T, G | I2 contains a constant or reference for the value to be divided into I1. Range for constants in double precision signed integer operations is minimum/maximum DINT. |
| ok | flow, none | The OK output is energized when the function is performed without overflow. |
| Q | All data types: R, AI, AQ<br>INT data type only: I, Q, M, T, G | Output Q contains the result of dividing I1 by I2 to obtain a remainder. |

## Example

In the example, the remainder of the integer division of BOXES into PALLETS is placed into NT_FULL whenever %I0001 is ON.

**Figure 126**

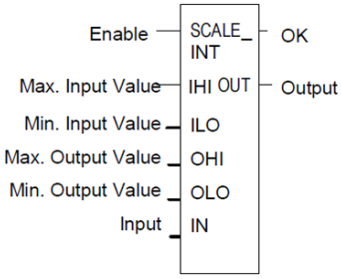


## 10.5.3 Math and Numerical Functions Scaling

The Scaling function scales an input parameter and places the result in an output location.

For integer-type data, all parameters must be integer-based (signed). For wordtype data, all parameters must be word-based (unsigned)

**Figure 127**



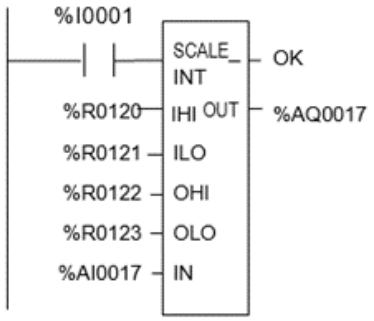

## Parameters of the Scaling Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| IHI ILO | R ,AI, AQ, constant | IHI and ILO contain a constant or reference for the upper and lower limits of the unscaled data. These limits, together with the values for OHI and OLO, are used to calculate the scaling factor that will be applied to the input value IN. |
| OHI OLO | R ,AI, AQ, constant | OHI and OLO contain a constant or reference for the upper and lower limits of the scaled data. |
| IN | R, AI, AQ, constant | IN contains a constant or reference for the actual value to be scaled. |
| ok | flow, none | The OK output is energized when the function is performed without overflow. |
| OUT | R, AI, AQ | Output OUTcontains the scaled equivalent of the input value. |

## Example

In the example, the registers %R0120 through %R0123 are used to store the high and low scaling values. The input value to be scaled is analog input %AI0017. The scaled output data is used to control analog output %AQ0017. The scaling is performed whenever %I0001 is ON.

**Figure 128**




## 10.5.4 Math and Numerical Functions Square Root

The Square Root function finds the square root of a value. When the function receives power flow, the value of output Q is set to the integer portion of the square root of the input IN. The output Q must be the same data type as IN.

The Square Root function operates on these types of data:

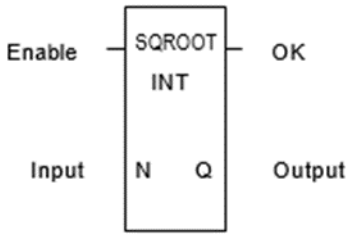
| | |
|---|---|
| INT | Signed integer |
| DINT | Double precision signed integer |
| REAL | Floating Point |

OK is set ON if the function is performed without overflow, unless one of these invalid REAL operations occurs:

- IN < 0
- IN is NaN (Not a Number)

Otherwise, OK is set OFF.

**Figure 129**




### Parameters of the Square Root Function

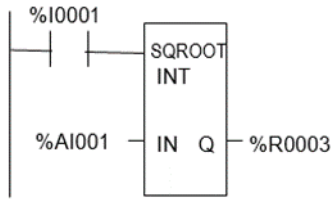
| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |

| Input/ Output | Choices | Description |
|---|---|---|
| IN | All data types: R, AI, AQ, constant<br>INT data type only: I, Q, M, T, G | A constant or reference for the value whose square root is to be calculated. If IN is less than zero, the function will not pass power flow. Range for constants is minimum/maximum DINT for double-precision signed integer operations. |
| ok | flow, none | The OK output is energized when the function is performed without overflow, unless an invalid operation occurs. |
| Q | All data types: R, AI, AQ<br>INT data type only: I, Q, M, T, G | Output Q contains the square root of IN. |

## Example

In the example, the square root of the integer number located at %AI001 is placed into the result located at %R0003 whenever %I0001 is ON.

**Figure 130**



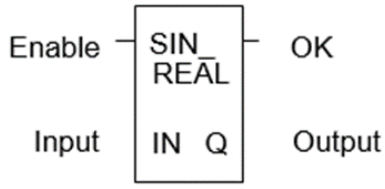

## 10.5.5 Math and Numerical Functions Trigonometric Functions

There are six Trigonometric functions: Sine, Cosine, Tangent, Inverse Sine, Inverse Cosine, and Inverse Tangent.

## Sine, Cosine, and Tangent

When a Sine, Cosine, or Tangent function receives power flow, it operates on IN, whose units are radians, and stores the result in output Q. Both IN and Q are floating-point values.

**Figure 131**




The Sine, Cosine, and Tangent functions accept a broad range of input values, where

$$-2^{63} < IN < +2^{63}, (2^{63} = 9.22x10^{18})$$

## Inverse Sine, Cosine, and Tangent

When an Inverse Sine, Cosine, or Tangent function receives power flow, it operates on IN and stores the result in output Q, whose units are radians. Both IN and Q are floating-point values.

The Inverse Sine and Cosine functions accept a narrow range of input values, where

$$-1 \leq IN \leq 1.$$

Given a valid value for the IN parameter, the Inverse Sine Real function produces a result Q such that:

$$ASIN\ (IN) = \pi/2 \leq Q \leq \pi/2$$

The Inverse Cosine Real function produces a result Q such that:

$$ACOS\ (IN) = 0 \leq Q \leq \pi$$

The Inverse Tangent function accepts the broadest range of input values, where

$$-\infty \leq IN \leq +\infty.$$

Given a valid value for the IN parameter, the Inverse Tangent Real function produces a result Q such that:

$$ATAN\ (IN) = \pi/2 \leq Q \leq \pi/2$$

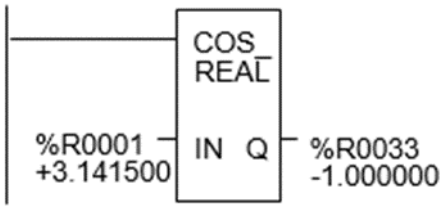
## Parameters of the Trigonometric Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| IN | R, AI, AQ, constant | IN contains the constant or reference real value to be operated on. |
| ok | flow, none | OK is energized when the function is performed without overflow, unless an invalid operation occurs and/or IN is NaN. |
| Q | R, AI, AQ | Output Q contains the trigonometric value of IN. |

## Example

In the example, the Cosine of the value in %R0001 is placed in %R0033.

**Figure 132**



## 10.5.6 Math and Numerical Functions Logarithmic / Exponential Functions

When a Logarithmic or Exponential function receives power flow, it performs the appropriate logarithmic/exponential operation on the Real value in input IN and places the result in output Q.

- For the Base 10 Logarithm (LOG) function, the base 10 logarithm of IN is placed in Q.
- For the Natural Logarithm (LN) function, the natural logarithm of IN is placed in Q.
- For the Power of E (EXP) function, e is raised to the power specified by IN and the result is placed in Q.
- For the Power of X (EXPT) function, the value of input I1 is raised to the power specified by the value I2 and the result is placed in output Q. (The EXPT function has three input parameters and two output parameters.)

The OK output receives power flow unless the input is NaN (Not a Number) or is negative.

**Figure 133**

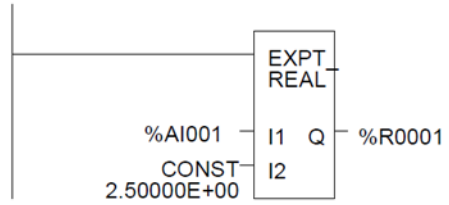

### Parameters of the Logarithmic/Exponential Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| IN or I1, I2 | R, AI, AQ, constant | For EXP, LOG, and LN, IN contains the real value to be operated on. The EXPT function has two inputs, I1 and I2. For EXPT, I1 is the base value and I2 is the exponent. |
| ok | flow, none | OK is energized when the function is performed without overflow, unless an invalid operation occurs and/or IN is NaN or is negative. |
| Q | R, AI, AQ | Output Q contains the logarithmic/exponential value of IN. |

### Example of the EXPT Function

In the example, the value of %AI001 is raised to the power of 2.5 and the result is placed in %R0001.

**Figure 134**

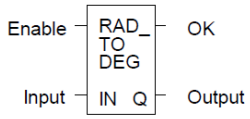



## 10.5.7 Math and Numerical Functions Radian Conversion Functions

When Degree/Radian Conversion function receives power flow, the appropriate conversion (radians to degrees or degrees to radians) is performed on the Real value in input IN and the result is placed in output Q.

The OK output will receive power flow unless IN is NaN (Not a Number).

**Figure 135**



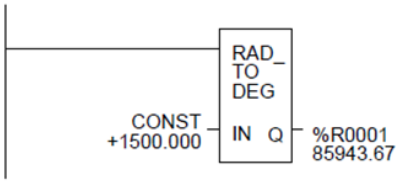

### Parameters of the Radian Conversion Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| IN | R, AI, AQ, constant | IN contains the real value to be operated on. |
| ok | flow, none | The OK output is energized when the function is performed without overflow, unless IN is NaN. |
| Q | R, AI, AQ | Output Q contains the converted value of IN. |

### Example

In the example, +1500 is converted to DEG and is placed in %R0001.

**Figure 136**



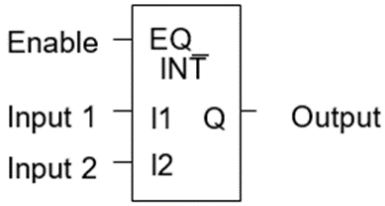
# 10.6 Relational Functions

The Relational functions can be used to compare two numbers and to determine whether a number lies within a specified range.

| | |
|---|---|
| Equal | Test two numbers for equality |
| Not Equal | Test two numbers for non-equality |
| Greater Than | Test whether one number is greater than another |
| Greater Than or Equal | Test whether one number is greater than or equal to another |
| Less Than | Test whether one number is less than another |
| Less Than or Equal | Test whether one number is less than or equal to another |
| Range | Tests whether one number lies between two other numbers |

When the function receives power flow, it compares input IN1 to input IN2. These parameters must be the same data type.

**Figure 137**




If inputs IN1 and IN2 match the specified relational condition, output Q receives power flow and is set ON (1); otherwise, it is set OFF (0).

### Data Types for Relational Functions

Relational functions operate on these types of data:

| | |
|---|---|
| INT | Signed integer |
| DINT | Double precision signed integer |
| REAL | Floating Point |

The %S0020 bit is set ON when a relational function using Real data executes successfully. It is cleared when either input is NaN (Not a Number).

## 10.6.1 Relational Functions Equal, Not Equal, Less Than, Less/Equal, Greater Than, Greater/Equal

### Parameters for the Relational Functions

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| IN1 | R, AI, AQ, constant<br>For INT data only: I, Q, M, T, G | IN1 contains a constant or reference for the first value to be compared. IN1 must be a valid number. Constants must be integers for double precision signed integer operations.<br>IN1 is on the left side of the relational equation, as in IN1 < IN2. |
| IN2 | R, AI, AQ, constant<br>For INT data only: I, Q, M, T, G | IN2 contains a constant or reference for the second value to be compared. IN2 must be a valid number. Constants must be integers for double precision signed integer operations.<br>IN2 is on the right side of the relational equation, as in IN1 < IN2. |
| Q | flow, none | Output Q is energized when IN1 and IN2 match the specified relation. |

### Example

In the example, two double precision signed integers are tested for equality. When the relay %I0001 passes power flow to the LE (Less or Equal) function, the value presently in the reference nicknamed PWR_MDE is compared to the value presently in the reference BIN_FUL. If the value in PWR_MDE is less than or equal to the value in BIN_FUL, coil %Q0002 is turned on.

**Figure 138**


## 10.6.2 Relational Functions Range

The Range function determines if a value is within the range of two numbers.

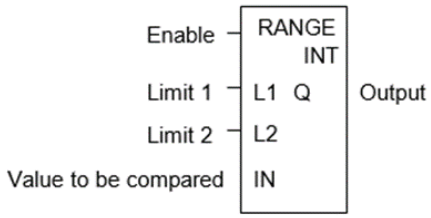
### Data Types for the Range Function

The Range function operates on these types of data:

| | |
|---|---|
| INT | Signed integer (default). |
| DINT | Double precision signed integer. |
| WORD | Word data type. |

When the Range function is enabled, it compares the value of input IN against the range specified by limits L1 and L2. Either L1 or L2 can be the high or low limit. When the value is within the range specified by L1 and L2, inclusive, output parameter Q is set ON (1). Otherwise, Q is set OFF (0).

**Figure 139**



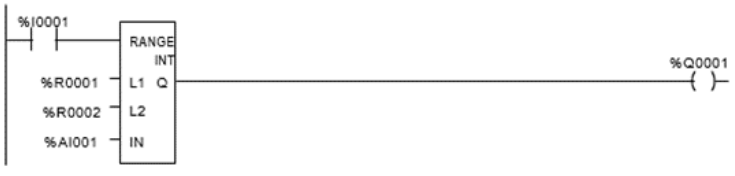

## Parameters for the Range Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| L1 | R, AI, AQ, constant<br>INT and WORD only: I, Q, M, T, G | L1 contains the start point of the range.<br>Constants must be integer values for double precision signed integer operations. |
| L2 | R, AI, AQ, constant<br>INT and WORD only: I, Q, M, T, G | L2 contains the end point of the range.<br>Constants must be integer values for double precision signed integer operations. |
| IN | R, AI, AQ<br>INT and WORD only: I, Q, M, T, G | IN contains the value to be compared against the range specified by L1 and L2. |
| Q | flow, none | Output Q is energized when the value in IN is within the range specified by L1 and L2, inclusive. |

## Example

In this example, when the Range function receives power flow from relay %I0001, the function determines whether the value in %AI001 is within the range 0 to 100. %R0001 contains the value 100. %R2 contains the value 0.

**Figure 140**




Output coil %Q0001 is On only if the value presently in %AI0001 is within the range 0 to 100.

| IN Value %AI001 | Q State %Q0001 |
|---|---|
| < 0 | OFF |
| 0 — 100 | ON |
| > 100 | OFF |

# 10.7 Relay Functions

- Normally Open Contact –| |–
- Normally Closed Contact –|/|–
- Normally Open Coil –( )–
- Retentive SET Coil –(SM)–
- Retentive RESET Coil –(RM)–
- Negated Retentive Coil –(/M)–
- Negated Coil –(/)–
- Retentive Coil –(M)–
- SET Coil –(S)–
- RESET Coil –(R)–
- Positive Transition Coil –(↑)–
- Negative Transition Coil –(↓)–
- Vertical Link vert |
- Horizontal Link horz –
- Continuation Coil –––<+>
- Continuation Contact <+>–––

Each relay contact and coil has one input and one output. Together, they provide logic flow through the contact or coil.

Input→ ——| |—— ← Output

## 10.7.1 Relay Functions Normally-open, Normally-closed, Continuation Contacts

A contact is used to monitor the state of a reference. Whether the contact passes power flow depends on the state or status of the reference being monitored and on the contact type. A reference is ON if its state is 1; it is OFF if its state is 0.

| Type of Contact | Display | Contact Passes Power to Right: |
|---|---|---|
| Normally Open | -\| \|- | When reference is ON. |
| Normally Closed | -\|/\|- | When reference is OFF. |
| Continuation Contact | <+>--- | If the preceding continuation coil is set ON. |

### Normally Open Contact -| |-

A normally open contact acts as a switch that passes power flow if the associated reference is ON (1).

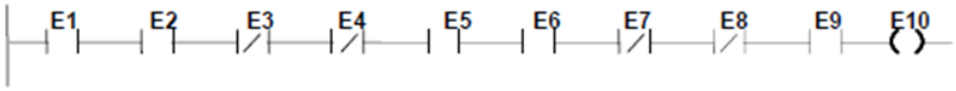
### Normally Closed Contact -|/|-

A normally closed contact acts as a switch that passes power flow if the associated reference is OFF (0).

### Example

The example shows a rung with 10 elements having nicknames from E1 to E10. Coil E10 is ON when reference E1, E2, E5, E6, and E9 are ON and references E3, E4, E7, and E8 are OFF.

**Figure 141**



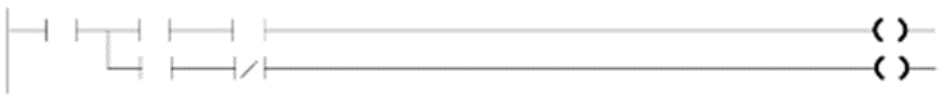

### Continuation Coils and Contacts

Continuation coils and continuation contacts are used to continue relay ladder rung logic beyond the last column. The state of the last executed continuation coil is the flow state used on the next executed continuation contact. If the flow of logic does not execute a continuation coil before it executes a continuation contact, the state of the contact is no flow. There can be only one continuation coil and contact per rung; the continuation contact must be in column 1, and the continuation coil must be in the last column.

## 10.7.2 Relay Functions Coils

Coils are used to control discrete references. Conditional logic must be used to control the flow of power to a coil. Coils cause action directly; they do not pass power flow to the right. If additional logic in the program should be executed as a result of the coil condition, an internal reference for the coil, or a continuation coil/contact combination may be used. Coils are always located at the rightmost position of a line of logic:

**Figure 142**



## References and Coil Checking

When the level of coil checking is set to “single”, you can use a specific %M or %Q reference with only one Coil, but you can use it with one Set Coil and one Reset Coil simultaneously. When the level of coil checking is “warn multiple” or “multiple”, each reference can be used with multiple Coils, Set Coils, and Reset Coils. With multiple usage, a reference could be turned On by either a Set Coil or a normal Coil and could be turned Off by a Reset Coil or by a normal Coil.

## Power Flow and Retentiveness

The following table summarizes how power flow to different types of coils affects their reference. The states of retentive coils are saved when power is cycled or when the PLC goes from Stop to Run mode. The states of non-retentive coils are set to zero when power is cycled or the PLC goes from Stop to Run mode.

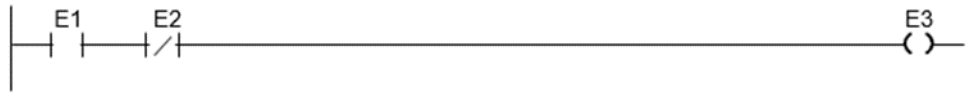
| Type of Coil | Symbol | Power to Coil | Result |
|---|---|---|---|
| Normally Open | -( )- | ON OFF | Sets reference ON, non-retentive.<br>Sets reference OFF, non-retentive. |
| Negated | -(/)- | ON OFF | Sets reference OFF, non-retentive.<br>Sets reference ON, non-retentive. |
| Retentive | -(M)- | ON OFF | Sets reference ON, retentive.<br>Sets reference OFF, retentive. |
| Negated Retentive | -(/M)- | ON OFF | Sets reference OFF, retentive.<br>Sets reference ON, retentive. |
| Positive Transition | -(P)- | OFF→ON | If power flow into the coil was OFF the previous sweep and is ON this sweep, sets the coil ON. |
| Negative Transition | -(N)- | ON→OFF | If power flow into the coil was ON the previous sweep and is OFF this sweep, sets the coil ON. |
| SET | -(S)- | ON OFF | Sets reference ON until reset OFF by ( R ), non-retentive.<br>Does not change the coil state, non-retentive. |
| RESET | -(R)- | ON OFF | Sets reference OFF until set ON by (S), non-retentive.<br>Does not change the coil state, non-retentive. |
| Retentive SET | -(SM)- | ON OFF | Sets reference ON until reset OFF by (RM), retentive.<br>Does not change the coil state. |
| Retentive RESET | -(RM)- | ON OFF | Sets reference OFF until set ON by (SM)-, retentive.<br>Does not change the coil state. |
| Continuation Coil | ----<+> | ON OFF | Sets next continuation contact ON.<br>Sets next continuation contact OFF. |

A coil sets a discrete reference ON while it receives power flow. It is non-retentive; therefore, it cannot be used with system status references (%SA, %SB, %SC, or %G).

### Example

In the example, coil E3 is ON when reference E1 is ON and reference E2 is OFF.

**Figure 143**




## Negated Coil

A negated coil sets a discrete reference ON when it does not receive power flow. It is not retentive, so it cannot be used with system status references (%SA, %SB, %SC, or %G).

### Example

In the example, coil E3 is ON when reference E1 is OFF.

**Figure 144**


## Retentive Coil

Like a normally open coil, the retentive coil sets a discrete reference ON while it receives power flow. The state of the retentive coil is retained across power failure. Therefore, it cannot be used with references from strictly non-retentive memory (%T).

## Negated Retentive Coil

The negated retentive coil sets a discrete reference ON when it does not receive power flow. The state of the negated retentive coil is retained across power failure. Therefore, it cannot be used with references from strictly non-retentive memory (%T).

## Positive Transition Coil

If the reference associated with a positive transition coil was OFF , when the coil receives power flow it is set to ON until the next time the coil is executed. (If the rung containing the coil is skipped on subsequent sweeps, it will remain ON.) This coil can be used as a one-shot.

Do not write from external devices (e.g., PCM, programmer, ADS, etc.) to references used on positive transition coils since it will destroy the one-shot nature of these coils.

Transitional coils can be used with references from either retentive or non-retentive memory (%Q, %M, %T, %G, %SA, %SB, or %SC).

## Negative Transition Coil

If the reference associated with this coil is OFF, when the coil stops receiving power flow the reference is set to ON until the next time the coil is executed.

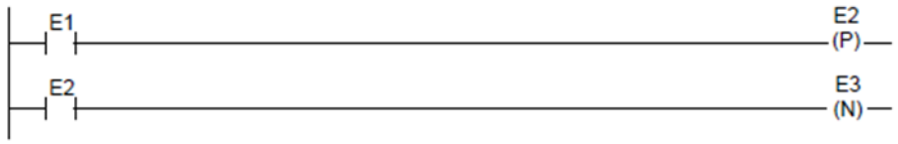
Do not write from external devices to references used on negative transition coils since it will destroy the one-shot nature of these coils.

Transitional coils can be used with references from either retentive or non-retentive memory (%Q, %M, %T, %G, %SA, %SB, or %SC).

## Example

In the example, when reference E1 goes from OFF to ON, coils E2 and E3 receive power flow, turning E2 ON for one logic sweep. When E2 goes from ON to OFF, power flow is removed from E2 and E3, turning coil E3 ON for one sweep.

**Figure 145**




## SET Coil

SET and RESET are non-retentive coils that can be used to keep (“latch”) the state of a reference either ON or OFF. When a SET coil receives power flow, its reference stays ON (whether or not the coil itself receives power flow) until the reference is reset by another coil.

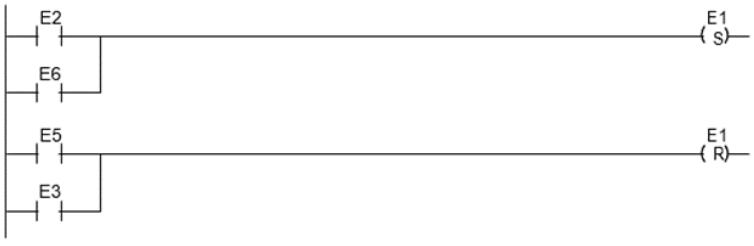
## RESET Coil

The RESET coil sets a discrete reference OFF if the coil receives power flow. The reference remains OFF until the reference is set by another coil. The last-solved SET coil or RESET coil of a pair takes precedence.

### Example

In the example, the coil represented by E1 is turned ON whenever reference E2 or E6 is ON. The coil represented by E1 is turned OFF whenever reference E5 or E3 is ON.

**Figure 146**



### Retentive SET Coil

Retentive SET and RESET coils are similar to SET and RESET coils, but they are retained across power failure or when the PLC transitions from to Run mode. A retentive SET coil sets a discrete reference ON if the coil receives power flow. The reference remains ON until reset by a retentive RESET coil.

### Retentive RESET Coil

This coil sets a discrete reference OFF if it receives power flow. The reference remains OFF until set by a retentive SET coil. The state of this coil is retained across power failure or when the PLC transitions from Stop to Run mode.

## 10.8 Table Functions

The Table functions are used to:

- Copy array data: ARRAY MOVE
- Search for values in an array

The maximum length allowed for these functions is 32,767 for any type.

#### Data Types for the Table Functions

Table functions operate on these types of data:

| | |
|---|---|
| INT | Signed integer |
| DINT | Double precision signed integer |
| BOOL * | Bit data type |
| BYTE | Byte data type |
| WORD | Word data type |

* Applies to Array Move only.

### 10.8.1 Table Functions Array Move

The Array Move function copies a specified number of elements from a source array to a destination array. When the function receives power flow, it copies the number of elements specified from the input array, starting at the indexed location. The function then writes the copied elements to the output array starting with the indexed location.

For bit data, when word-oriented memory is selected for the parameters of the source array and/or destination array starting address, the least significant bit of the specified word is the first bit of the array.

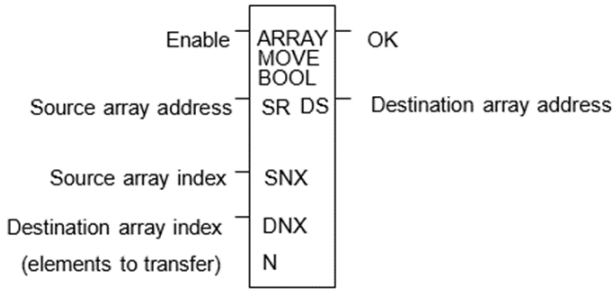
The indices in an Array Move instruction are 1-based. In using an Array Move, no element outside either the source or destination arrays (as specified by their starting address and length) may be referenced.

The OK output receives power flow unless one of the following occurs:

- Enable is OFF.
- (N + SNX – 1) is greater than (length).

- (N + DNX – 1) is greater than (length).

**Figure 147**



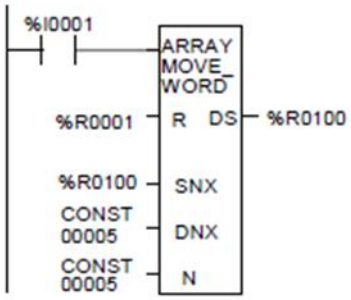

## Parameters for the Array Move Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the operation is performed. |
| SR | For all: R, AI, AQ<br>For INT, BIT, BYTE, WORD: I, Q, M, T, G,<br>For BIT, BYTE, WORD: SA, SB, SC | SR contains the starting address of the source array. For ARRAY_MOVE_ BOOL, any reference may be used; it does not need to be byte aligned. |
| SNX | I, Q, M, T, G, R, AI, AQ, constant | SNX contains the index of the source array. |
| DNX | I, Q, M, T, G, R, AI, AQ, constant | DNX contains the index of the destination array. |
| N | I, Q, M, T, G, R, AI, AQ, constant | N provides a count indicator. |
| ok | flow, none | OK is energized whenever enable is energized. |
| DS | For all: SA, SB, SC, R, AI, AQ<br>For INT, BIT, BYTE, WORD: I, Q, M, T, G | The starting address of the destination array. For ARRAY_MOVE_ BOOL, any reference may be used; it does not need to be byte aligned. |
| length | | The number of elements starting at SR and DS that make up each array. It is defined as the length of SR+DS. |

## Example 1:

In this example, if %R100=3 then %R0003 - %R0007 of the array %R0001 - %R0016 is read and is written into %R0104 - %R0108 of the array %R0100 - %R0115. (%R001 and %R0100 are declared as type WORD of length 16.)

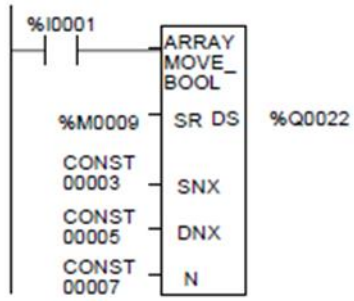
**Figure 148**




## Example 2:

Using bit memory for SR and DS, %M0011 - %M0017 of the array %M0009 - %M0024 is read and then written to %Q0026 - %Q0032 of the array %Q0022 - %Q0037. (%M009 and %Q0022 are declared as type BOOL of length 16).

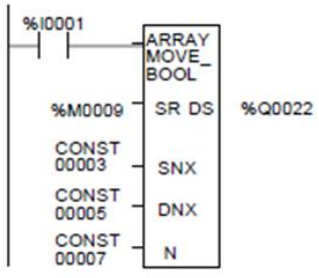
**Figure 149**




## Example 3:

Using word memory, for SR and DS, the third least significant bit of %R0001 through the second least significant bit of %R0002 of the array containing all 16 bits of %R0001 and four bits of %R0002 is read and then written into the fifth least significant bit of %R0100 through the fourth least significant bit of %R0101 of the array containing all 16 bits of % R0100 and four bits of %R0101. 0001 and %R0100 are declared as type BOOL of length 20).

**Figure 150**



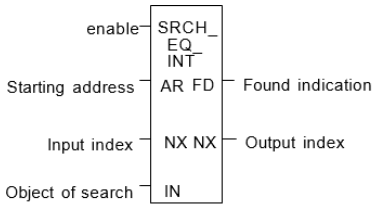
## 10.8.2 Table Functions Search for Array Values

Use the Search functions listed below to search for values in an array.

- Search Equal — Equal to a specified value.
- Search Not Equal — Not equal to a specified value.
- Search Greater Than — Greater than a specified value.
- Search Greater Than or Equal — Greater than or equal to a specified value.
- Search Less Than — Less than a specified value.
- Search Less Than or Equal — Less than or equal to a specified value.

When the Search function receives power, it searches the specified array. Searching begins at the starting address (AR) plus the index value (NX).

**Figure 151**




The search continues until the array element of the search object (IN) is found or until the end of the array is reached. If an array element is found, the Found Indication (FD) is set ON and the Output Index (output NX) is set to the relative position of this element within the array. If no array element is found before the end of the array is reached, the Found Indication (FD) is set OFF and the Output Index (output NX) is set to zero.

Valid values for input NX are 0 to (length)- 1. NX should be set to zero to begin searching at the first element. This value increments by one at the time of execution. Therefore, the values of output NX are 1 to (length). If the value of input NX is out-of-range, (< 0 or > length), its value defaults to zero.

### Parameters of the Search Functions

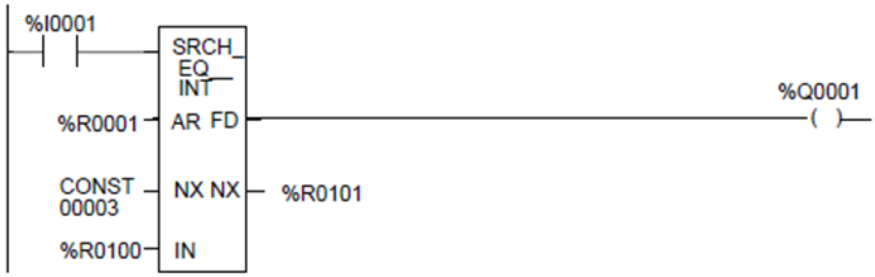
| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is enabled, the search is performed. |
| AR | For all: R, AI, AQ<br>For INT, BYTE, WORD: I, Q, M, T, G,<br>For BYTE, WORD: S | Contains the starting address of the array. |
| Input NX | I, Q, M, T, G, R, AI, AQ, constant | Contains the zero-based index into the array at which to begin the search |
| IN | For all: R, AI, AQ, constant<br>For INT, BYTE, WORD: I, Q, M, T, G,<br>For BYTE, WORD: S | IN contains the object of the search. |

| Input/ Output | Choices | Description |
|---|---|---|
| Output NX | I, Q, M, T, G, R, AI, AQ | Holds the one-based position within the array of the search target. |
| FD | flow, none | FD indicates that an array element has been found and the function was successful. |
| length | 1 to 32,767 bytes or words. | The number of elements starting at AR that make up the array to be searched. |

## Example 1:

The array AR is defined as memory addresses %R0001 - %R0005. When EN is ON, the portion of the array between %R0004 and %R0005 is searched for an element whose value is equal to IN. If %R0001 = 7, %R0002 = 9, %R0003 = 6, %R0004 = 7, %R0005 =7, and %R0100 = 7, then the search will begin at %R0004 and conclude at %R0004 when FD is set ON and a 4 is written to %R0101

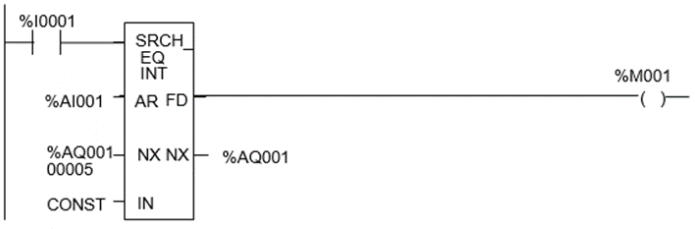
**Figure 152**




## Example 2:

Array AR is defined as memory addresses %AI001 - %AI016. The values of the array elements are 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0, and 500. Initially, % AQ001 is 5. When EN is ON, each sweep will search the array looking for a match to the IN value of 0. The first sweep will start searching at %AI006 and find a match at % AI007, so FD is ON and %AQ001 is 7. The second sweep will start searching at %AI008 and find a match at %AI015, so FD remains ON and %AQ001 is 15. The next sweep will start at %AI016. Since the end of the array is reached without a match, FD is set OFF and %AQ001 is set to zero. The next sweep will start searching at the beginning of the array

**Figure 153**



# 10.9 Timer and Counter Functions

This section describes the timing and counting functions of the Instruction Set. The data associated with these functions is retentive through power cycles.

- On-delay Stopwatch Timer
- Off-delay Timer
- On-delay Timer
- Up Counter
- Down Counter

### Time-tick Contacts

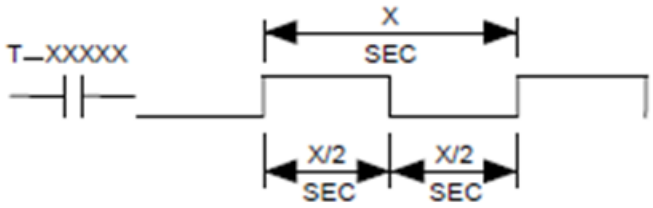
In addition to the Timer functions of the Instruction Set, the VersaMax PLC has four time-tick contacts. These contacts can be used to provide regular pulses of power flow to other program functions. The four time-tick contacts have time durations of 0.01 second, 0.1 second, 1.0 second, and 1 minute.

The state of these contacts does not change during the execution of the sweep. These contacts provide a pulse having an equal on and off time duration.

The contacts are referenced as T_10MS (0.01 second), T_100MS (0.1 second), T_SEC (1.0 second), and T_MIN (1 minute).

The following timing diagram represents the on/off time duration of these contacts.

**Figure 154**




These time-tick contacts represent specific locations in %S memory.

### Function Block Data Required for Timers and Counters

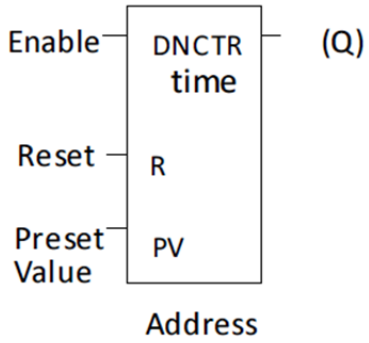
Each timer or counter uses three words (registers) of %R memory to store the following information:

| | |
|---|---|
| current value (CV) | word 1 |
| preset value (PV) | word 2 |
| control word | word 3 |

When you enter a timer or counter, you must enter a beginning address for these three words (registers). Do not use consecutive registers for the 3 word timer/counter blocks.

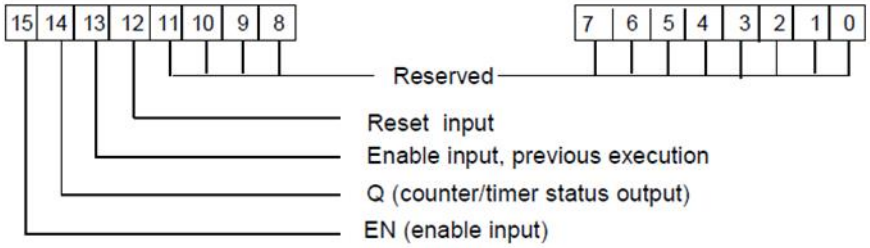
Timers and counters will not work if you place the current value of a block on top of the preset for the previous block.

**Figure 155**




The control word stores the state of the boolean inputs and outputs of its associated function block in the following format:

**Figure 156**




Bits 0 through 11 are used for timer accuracy; not for counters.

If the Preset Value (PV) is not a constant, PV is normally set to a different location than the second word. Some applications use the second word address for the PV, such as using %R0102 when the bottom data block starts at %R0101. It is then possible to change the Preset Value while the timer or counter is running. The first (CV) and third (Control) words can be read but should not be written, or the function will not work.

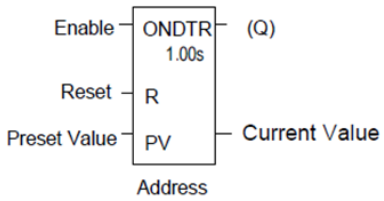
## 10.9.1 Timer and Counter Functions on Delay Stopwatch Timer

A retentive On-Delay Stopwatch Timer (ONDTR) increments while it receives power flow and holds its value when power flow stops. Time may be counted in tenths (0.1), hundredths (0.01), or thousandths (0.001) of a second. The range is 0 to +32,767-time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

When this function first receives power flow, it starts accumulating time (current value).

When this timer is encountered in the ladder logic, its Current Value is updated.

**Figure 157**




When the Current Value equals or exceeds the Preset Value PV, output Q is energized. As long as the timer continues to receive power flow, it continues accumulating until the maximum value is reached. Once the maximum value is reached, it is retained and output Q remains energized regardless of the state of the enable input.

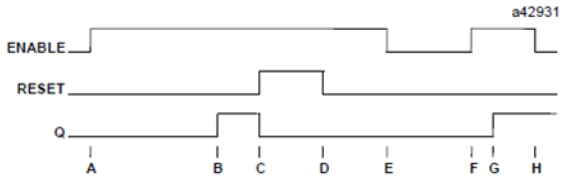
If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the current values of the timers will be the same.

## Parameters of the On-Delay Stopwatch Timer Function

| Input/ Output | Choices | Description |
|---|---|---|
| address | R | The function uses three consecutive words (registers) of %R memory to store the following:<br>• Current value (CV) =word 1.<br>• Preset value (PV) = word 2.<br>• Control word = word 3.<br>Do not use this address with other instructions.<br>Careful: Overlapping references cause erratic timer operation. |
| enable | flow | When enable receives power flow, the timer's Current Value increments. |
| R | flow | When R receives power flow, it resets the Current Value to zero. |
| PV | I ,Q, M, T, G, R, AI, AQ, constant, none | The Preset Value, which is used when the timer is enabled or reset. |
| Q | flow, none | Output Q is energized when the current value of the timer is greater than or equal to the Preset Value. |
| time | tenths, hundredths, or thousandths of seconds | Time increment for the low bit of the PV preset and CV current value. |

## Operation of the On Delay Timer Function

**Figure 158**




A. ENABLE goes high; timer starts accumulating

B. Current value reaches preset value PV; Q goes high

C. RESET goes high; Q goes low, accumulated time is reset (CV=0) D. RESET goes low; timer then starts accumulating again

E. ENABLE goes low; timer stops accumulating. Accumulated time stays the same

F. ENABLE goes high again; timer continues accumulating time

G. Current value becomes equal to preset value PV; Q goes high. Timer continues to accumulate time until ENABLE goes low, RESET goes high or current value becomes equal to the maximum time

H. ENABLE goes low; timer stops accumulating time.

When power flow to the timer stops, the current value stops incrementing and is retained. Output Q, if energized, will remain energized. When the function receives power flow again, the current value again increments, beginning at the retained value. When reset R receives power flow, the current value is set back to zero and output Q is de-energized unless PV equals zero.

## Example

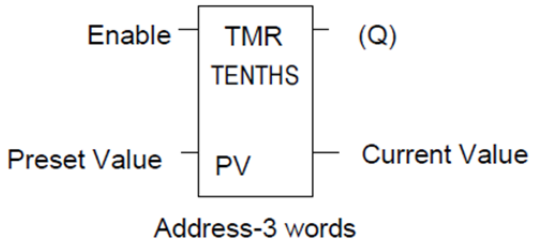
In the example, a retentive on-delay timer is used to create a signal (%Q0011) that turns on 8.0 seconds after %Q0010 turns on, and turns off when %Q0010 turns off.

**Figure 159**

## 10.9.2 Timer and Counter Functions On Delay Timer

The On-Delay Timer (TMR) increments while it receives power flow and resets to zero when power flow stops. Time may be counted in tenths of a second (the default selection), hundredths of a second, or thousandths of a second. The range is 0 to +32,767-time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

**Figure 160**




When the On Delay Timer function receives power flow, the timer starts accumulating time (Current Value). The Current Value is updated when it is encountered in the logic to reflect the total elapsed time the timer has been enabled since it was last reset.

If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the Current Values of the timers will be the same.

This update occurs as long as the enabling logic remains ON. When the current value equals or exceeds the Preset Value PV, the function begins passing power flow to the right. The timer continues accumulating time until the maximum value is reached. When the enabling parameter transitions from ON to OFF, the timer stops accumulating time and the Current Value is reset to zero.

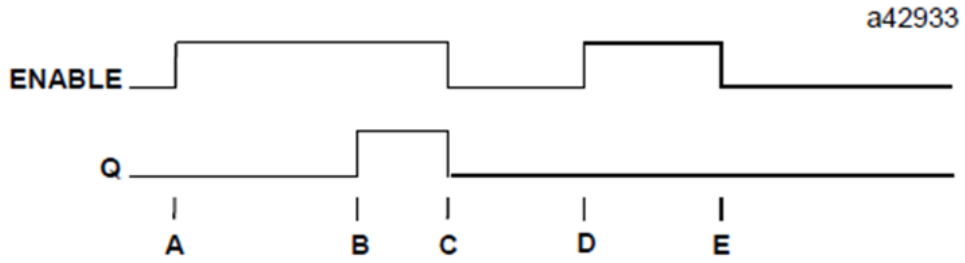
### Parameters for the On Delay Timer Function

| Input/ Output | Choices | Description |
|---|---|---|
| address | R | The function uses three consecutive words (registers) of %R memory to store the following:<br>• Current value (CV)= word 1.<br>• Preset value (PV) = word 2.<br>• Control word = word 3.<br>Do not use this address with other instructions.<br>Careful: Overlapping references cause erratic operation of |
| enable | flow | When enable receives power flow, the timer's current value is incremented. When the TMR is not enabled, the current value is reset to zero and Q is turned off. |
| PV | I Q, M, T, G, R, AI, AQ, constant, none | PV is the value to copy into the timer's preset value when the timer is enabled or reset. |

| Input/ Output | Choices | Description |
|---|---|---|
| Q | flow, none | Output Q is energized when TMR is enabled and the current value is greater than or equal to the preset value. |
| time | tenths (0.1), hundredths (0.01), or thousandths (0.001) of seconds | Time increment for the low bit of the PV preset and CV current value. |

## Operation of the On-Delay Timer Function

**Figure 161**




A. ENABLE goes high; timer begins accumulating time.

B. Current value reaches preset value PV; Q goes high, and timer continues accumulating time. C. ENABLE goes low; Q goes low; timer stops accumulating time and current time is cleared.

D. ENABLE goes high; timer starts accumulating time.

E. ENABLE goes low before current value reaches preset value PV; Q remains low; timer stops accumulating time and is cleared to zero (CV=0).

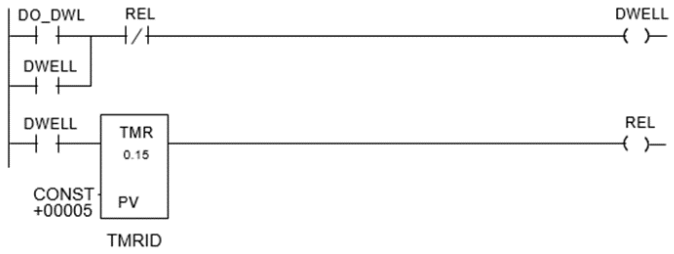
## Example

In the example, a delay timer (with address) TMRID is used to control the length of time that coil is on. This coil has been assigned the Nickname DWELL . When the normally open (momentary) contact with the Nickname DO_DWL is on, coil DWELL is energized.

The contact of coil DWELL keeps coil DWELL energized (when contact DO_DWL is released), and also starts the timer TMRID. When TMRID reaches its preset value of one-half second, coil REL energizes, interrupting the latched-on condition of coil DWELL.

The contact DWELL interrupts power flow to TMRID, resetting its current value and de-energizing coil REL. The circuit is then ready for another momentary activation of contact DO_DWL

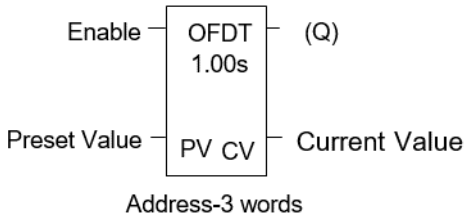
**Figure 162**




## 10.9.3 Timer and Counter Functions Off Delay Timer

The Off-Delay Timer increments while power flow is off and resets to zero when power flow is on. Time may be counted in tenths (0.1), hundredths (0.01), or thousandths (0.001) of a second. Range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

**Figure 163**




When the Off-Delay Timer first receives power flow, it passes power to the right, and the Current Value (CV) is set to zero. The function uses word 1 [register] as its CV storage location. The output remains on as long as the function receives power flow. If the function stops receiving power flow from the left, it continues to pass power to the right, and the timer starts accumulating time in the Current Value. The Off-Delay Timer does not pass power flow if the Preset Value is zero or negative.

If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the Current Values of the timers will be the same.

Each time the function is invoked with the enabling logic set to OFF, the Current Value is updated to reflect the elapsed time since the timer was turned off. When the Current Value (CV) is equal to the Preset Value (PV), the function stops passing power flow to the right and the timer stops accumulating. When the function receives power flow again, the current value resets to zero. When this timer is used in a program block that is not called every sweep, it accumulates time between calls to the program block unless it is reset. That means it functions like a timer in a program with a much slower sweep than the timer in the main program block. For program blocks that are inactive for a long time, the timer should be programmed to allow for this catch-up. For example, if a timer in a program block is reset and the program block is inactive for four minutes, when the program block is called, four minutes of time will have accumulated. This time is applied to the timer when enabled unless the timer is first reset.

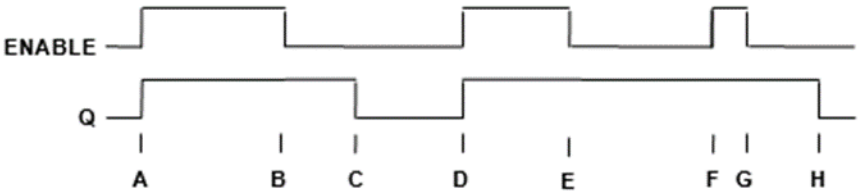
## Example

In the example, an Off-Delay Timer is used to turn off an output (%Q00001) whenever an input (%I00001) turns on. The output is turned on again 0.3 seconds after the input goes off.

**Figure 164**


## Operation of the Off-Delay Timer Function

**Figure 165**




A. ENABLE and Q both go high; timer is reset (CV = 0).

B. ENABLE goes low; timer starts accumulating time.

C. CV reaches PV; Q goes low, and timer stops accumulating time.

D. ENABLE goes high; timer is reset (CV = 0).

E. ENABLE goes low; timer starts accumulating time.

F. ENABLE goes high; timer is reset (CV = 0).

G. ENABLE goes low; timer begins accumulating time.

H. V reaches PV; Q goes low, and timer stops accumulating time.

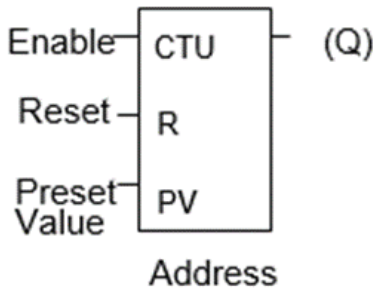
## Parameters of the Off-Delay Timer Function

| Input/ Output | Choices | Description |
|---|---|---|
| address | R | The function uses three consecutive words (registers) of %R memory to store the following:<br>• Current value (CV) = word 1.<br>• Preset value (PV) = word 2.<br>• Control word = word 3.<br>Do not use this address with other instructions.<br>Careful: Overlapping references cause erratic operation of the timer. |
| enable | flow | When enable receives power flow, the timer's current value is incremented. |
| PV | I Q, M, T, G, R, AI, AQ, constant, none | PV is the value to copy into the timer's preset value when the timer is enabled or reset. For a register (%R) OV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R0001 would use %R0002 as the PV parameter. |
| Q | flow, none | Output Q is energized when the current value is less than the preset value. The Q state is retentive on power failure; no automatic initialization occurs at power-up. |
| time | tenths, hundredths, or thousandths of seconds | Time increment for the low bit of the PV preset and CV current value. |

## 10.9.4 Timer and Counter Functions Up Counter

The Up-Counter function counts up to a designated value. The range is 0 to +32,767 counts. When the Up-Counter reset is ON, the Current Value of the counter resets to 0. Each time the enable input transitions from OFF to ON, the Current Value increments by 1. The current value can be incremented past the Preset Value PV. The output is ON whenever the Current Value is greater than or equal to the Preset Value. The state of the CTU is retentive on power failure; no automatic initialization occurs at power-up.

**Figure 166**



## Parameters of the Up Counter Function

| Input/ Output | Choices | Description |
|---|---|---|
| address | R | The function uses three consecutive words (registers) of %R memory to store the following:<br>• Current value (CV) = word 1.<br>• Preset value (PV) = word 2.<br>• Control word = word 3.<br>Do not use this address with another up counter, down counter, or any other instruction or improper operation will result.<br>Careful: Overlapping references cause erratic operation of the counter. |
| enable | flow | On a positive transition of enable, the current count is incremented by one. |
| R | flow | When R receives power flow, it resets the current value back to zero. |
| PV | I, Q, M, T, G, R, AI, AQ, constant, none | PV is the value to copy into the counter's preset value when the counter is enabled or reset. |
| Q | flow, none | Output Q is energized when the Current Value is greater than or equal to the Preset Value. |

## Example of the Up-Counter Function

In the example, every time input %I0012 transitions from OFF to ON, up counter PRT_CNT counts up by 1; internal coil %M0001 is energized when 100 parts have been counted. When %M0001 is ON, the accumulated count is reset to zero.

**Figure 167**

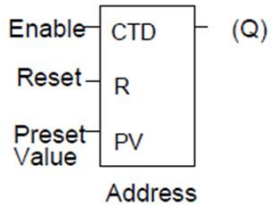

# 10.9.5 Timer and Counter Functions Down Counter

The Down Counter function counts down from a preset value. The minimum Preset Value is zero; the maximum present value is +32,767 counts. The minimum Current Value is –32,768. When reset, the Current Value of the counter is set to the Preset Value PV. When the enable input transitions from OFF to ON, the Current Value is decremented by one.

The output is ON whenever the Current Value is less than or equal to zero.

The Current Value of the Down Counter is retentive on power failure; no automatic initialization occurs at power-up.

**Figure 168**



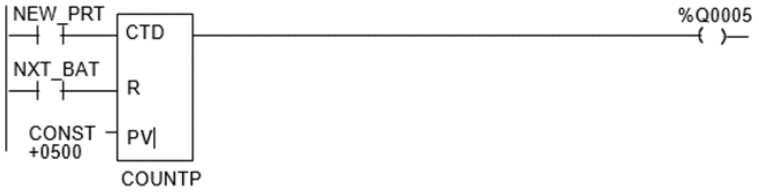

## Parameters of the Down Counter Function

| Input/ Output | Choices | Description |
|---|---|---|
| address | R | The function uses three consecutive words (registers) of %R memory to store the following:<br>• Current value (CV) = word 1.<br>• Preset value (PV) = word 2.<br>• Control word = word 3.<br>Do not use this address with another down counter, up counter, or any other instruction or improper operation will result.<br>**Careful:** Overlapping references will result in erratic counter operation. |
| enable | flow | On a positive transition of enable, the Current Value is decremented by one. |
| R | flow | When R receives power flow, it resets the Current Value to the Preset Value. |
| PV | I, Q, M, T, G, R, AI, AQ, constant, none | PV is the value to copy into the counter's Preset Value when the counter is enabled or reset. |
| Q | flow, none | Output Q is energized when the Current Value is less than or equal to zero. |

## Example 1:

In the example, the down counter identified as COUNTP counts 500 new parts before energizing output %Q0005.

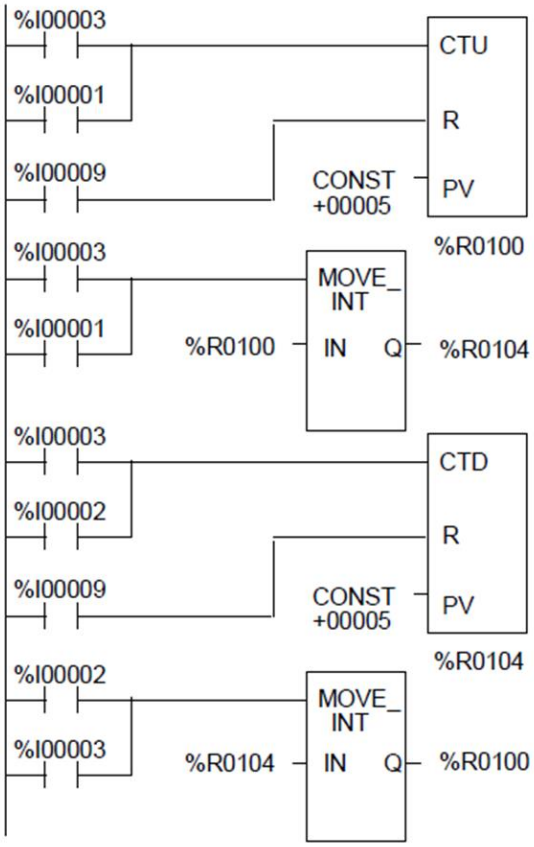
**Figure 169**



## Example 2: Keeping Track of Parts in a Temporary Storage Area

The following example shows how the PLC can keep track of the number of parts in a temporary storage area. It uses an up/down counter pair with a shared register for the accumulated or current value. When parts enter the storage area, the up counter increases the current value of the parts in storage by 1. When a part leaves the storage area, the down counter decrements by 1, decreasing the inventory storage value by 1. The two counters use different register addresses. When a register counts, its current value must be moved to the current value register of the other counter.

**Figure 170**




Refer to the sections on Math functions for an example of using the Addition and Subtraction functions to provide storage tracking.

# Chapter 11: The Service Request Function

This chapter explains the Service Request (SVCREQ) function, which requests a special PLC service. It describes SVCREQ parameters for the VersaMax™ CPU.

- SVCREQ Function Numbers
- Format of the SVCREQ Function
- SVCREQ 1: Change/Read Constant Sweep Timer
- SVCREQ 2: Read Window Times
- SVCREQ 3: Change Programmer Communications Window Mode
- SVCREQ 4: Change System Communications Window Mode
- SVCREQ 6: Change/Read Number of Words to Checksum
- SVCREQ 7: Read or Change the Time-of-Day Clock
- SVCREQ 8: Reset Watchdog Timer
- SVCREQ 9: Read Sweep Time from Beginning of Sweep
- SVCREQ 10: Read Folder Name
- SVCREQ 11: Read PLC ID
- SVCREQ 13: Shut Down (Stop) PLC
- SVCREQ 14: Clear Fault
- SVCREQ 15: Read Last-Logged Fault Table Entry
- SVCREQ 16: Read Elapsed Time Clock
- SVCREQ 18: Read I/O Override Status
- SVCREQ 23: Read Master Checksum
- SVCREQ 24: Reset Ethernet Daughter Board
- SVCREQ 26/30: Interrogate I/O

## 11.1 SVCREQ Function Numbers

Each Service Request has its own function number, as listed in the following table.

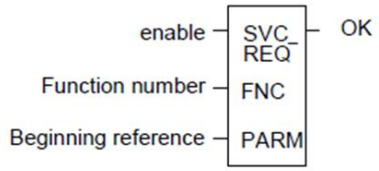
| Function # | Description |
|---|---|
| 1 | Change/Read Constant Sweep Timer |
| 2 | Read Window Times |
| 3 | Change Programmer Communications Window Mode and Time |
| 4 | Change System Communications Window Mode and Time |
| 5 | reserved |
| 6 | Change/Read Number of Words to Checksum |
| 7 | Change/Read Time-of-Day Clock |
| 8 | Reset Watchdog Timer |

| Function # | Description |
|---|---|
| 9 | Read Sweep Time from Beginning of Sweep |
| 10 | Read Folder Name |
| 11 | Read PLC ID |
| 12 | reserved |
| 13 | Shut Down the PLC |
| 14 | Clear Fault Tables |
| 15 | Read Last-Logged Fault Table Entry |
| 16 | Read Elapsed Time Clock |
| 17 | reserved |
| 18 | Read I/O Override Status |
| 19-22 | reserved |
| 23 | Read Master Checksum |
| 26/30 | Interrogate I/O |
| 27, 28 | reserved |
| 29 | Read Elapsed Power Down Time |
| 31-255 | reserved |

# 11.2 Format of the SVCREQ Function

The SVCREQ function has three inputs and one output.

**Figure 171**




When the SVCREQ receives power flow, the PLC is requested to perform the function number FNC indicated. Parameters for the function are located beginning at the reference given for PARM. This is the beginning of the “parameter block" for the function. The number of 16-bit references required depends on the SVCREQ function being used.

Parameter blocks may be used as both inputs for the function and the location where data may be output after the function executes. Therefore, data returned by the function is accessed at the same location specified for PARM.

The SVCREQ function passes power flow unless an incorrect function number, incorrect parameters, or out-of-range references are specified. Specific SVCREQ functions have additional causes for failure.

### 11.2.1 Parameters of the SVCREQ Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When enable is energized, the service request is performed. |
| FNC | I, Q M, T, G, R, AI, AQ, constant | Contains the constant or reference for the requested service. |
| PARM | I, Q M, T, G, R, AI, AQ | Contains the beginning reference for the parameter block for the requested service. |
| ok | flow, none | OK is energized when the function is performed without error. |

### 11.2.2 Example of the SVCREQ Function

In the example, when the enabling input %I0001 is ON, SVCREQ function number 7 is called, with the parameter block located starting at %R0001. Output coil %Q0001 is set ON if the operation succeeds.

**Figure 172**


## 11.3 SVCREQ 1: Change/Read Constant Sweep Timer

Use SVCREQ 1 to enable or disable Constant Sweep Time mode, change the length of the Constant Sweep Time, read whether Constant Sweep Time is currently enabled, or read the Constant Sweep Time length.

### 11.3.1 Input Parameter Block for SCVREQ 1

For this function, the parameter block has a length of two words.

#### Disable Constant Sweep Mode

To disable Constant Sweep mode, enter SVCREQ function #1 with this parameter block:

**Figure 173**

| | |
|---|---|
| address | 0 |
| address + 1 | ignored |

## Enable Constant Sweep Mode

To enable Constant Sweep mode, enter SVCREQ function #1 with this parameter block:

**Figure 174**

| | |
|---|---|
| address | 1 |
| address + 1 | 0 or timer value |

*Note:* *If the timer should use a new value, enter it in the second word. If the timer value should not be changed, enter 0 in the second word. If the timer value does not already exist, entering 0 causes the function to set the OK output to OFF.*

## Change the Constant Sweep Time

To change the timer value **without** changing the selection for sweep mode state, enter SVCREQ function #1 with this parameter block:

**Figure 175**

| | |
|---|---|
| address | 2 |
| address + 1 | new timer value |

## Read the Constant Sweep State and Time

To read the current timer state and value without changing either, enter SVCREQ function #1 with this parameter block:

**Figure 176**

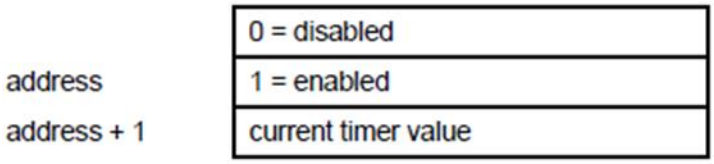
| | |
|---|---|
| address | 3 |
| address + 1 | ignored |

Successful execution will occur, unless:

1. A number other than 0, 1, 2, or 3 is entered as the requested operation.
2. The sweep time value is greater than 500ms (0.5 seconds).
3. Constant sweep time is enabled with no timer value programmed or with an old value of 0 for the timer.

After the function executes, the function returns the timer state and value in the same parameter block references:

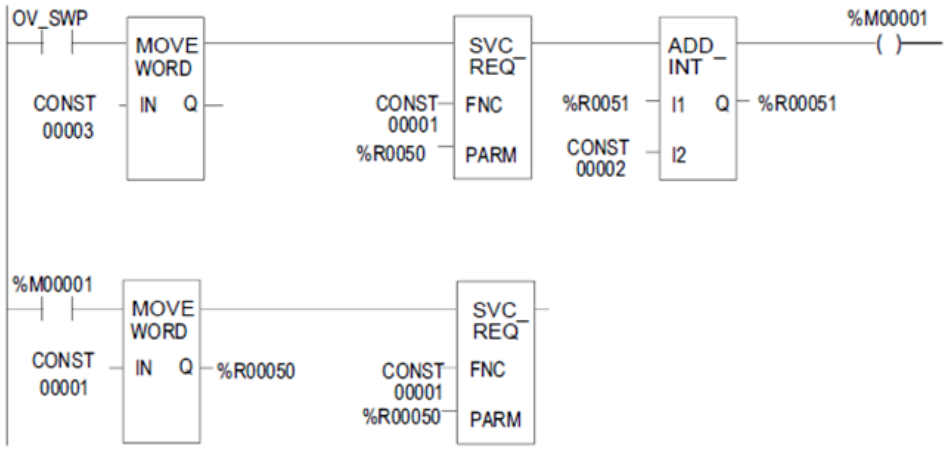
**Figure 177**




## Example of SVCREQ 1

In this example, if contact OV_SWP is set, the Constant SweepTimer is read, the timer is increased by two milliseconds, and the new timer value is sent back to the PLC. The parameter block is in local memory at location %R0050. Because the MOVE and ADD functions require three horizontal contact positions, the example logic uses discrete internal coil %M00001 as a temporary location to hold the successful result of the first rung line. On any sweep in which OV_SWP is not set, %M00001 is turned off.

**Figure 178**




# 11.4 SVCREQ 2: Read Window Times

SVCREQ 2 can be used to read the times of the programmer communications window and the system communications window. These windows can operate in Limited or Run to Completion Mode.

| Mode Name | Value | Description |
|---|---|---|
| Limited Mode | 0 | The execution time of the window is limited to 6ms. The window terminates when it has no more tasks to complete or after 6ms elapses. |
| Run to Completion Mode | 2 | Regardless of the time assigned to a window, it runs until all tasks within that window are completed (up to 400ms). |

A window is disabled when the time value is zero.

### 11.4.1 Output Parameter Block for SVCREQ 2

The parameter block has a length of three words:

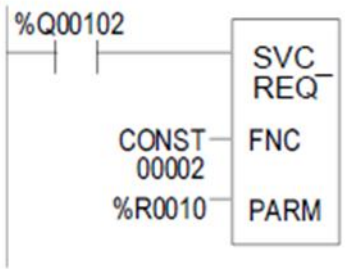
| | High Byte | Low Byte | |
|---|---|---|---|
| address | Mode | Value in ms | Programmer Window |
| address + 1 | Mode | Value in ms | System Communications Window |
| address + 2 | must be zero | must be zero | reserved |

All parameters are output parameters. It is not necessary to enter values in the parameter block to program this function.

#### Example of SVCREQ 2

In the following example, when enabling output %Q00102 is set, the CPU places the current time values of the windows in the parameter block starting at location %R0010.

**Figure 179**




## 11.5 SVCREQ 3: Change Programmer Communications Window Mode

Use SVCREQ 3 to change the programmer communications window mode (Limited or Run-to-Completion). The change occurs during the next CPU sweep after the function is called. The time of the window cannot be changed; it is always 6ms.

SVCREQ 3 passes power flow to the right unless a mode other than 0 (Limited) or 2 (Run-to-Completion) is selected.

The parameter block has a length of one word.

### 11.5.1 Changing the Programmer Communications Window Mode

To change the programmer window, enter SVCREQ 3 with this parameter block:

### Example of SVCREQ 3

In the following example, when enabling input %I006 goes ON, the programmer communications window is enabled and assigned a value of 6ms. The parameter block is in reference memory location %R0051.

**Figure 180**


## 11.6 SVCREQ 4: Change System Communications Window Mode

Use SVCREQ 4 to change the system communications window mode (Limited or Run-to-Completion). The change occurs during the next CPU sweep after the function is called. The time of the window cannot be changed; it is always 6ms.

SVCREQ 4 passes power flow to the right unless a mode other than 0 (Limited) or 2 (Run-to-Completion) is selected.

The parameter block has a length of one word.

### 11.6.1 Changing the System Communications Window Mode

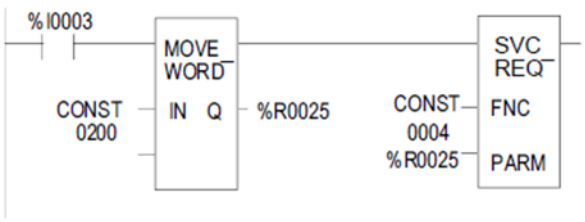
To change the programmer window, enter SVCREQ 4 with this parameter block:

| | High Byte | Low Byte |
|---|---|---|
| address | Mode | 6 |

### Example of SVCREQ 4

In the following example, when enabling input %I0003 is ON the system communications window is changed to Run-to-Completion mode. The parameter block is at location %R0025.

**Figure 181**



# 11.7 Change/Read Number of Words to Checksum

Use SVCREQ 6 to read or change the number of words in the program to be checksummed. The function is successful unless some number other than 0 or 1 is entered as the requested operation.

## 11.7.1 Parameter Block Formats for SVCREQ 6

The parameter block has a length of 2 words.

To read the word count, the first word of the parameter block must contain a zero:

**Figure 182**

| | |
|---|---|
| address | 0 (read word count) |
| address + 1 | ignored |

The function returns the current word count in the second word of the parameter block.

**Figure 183**

| | |
|---|---|
| address | 0 |
| address + 1 | current word count |

To change the word count, the first word of the parameter block must contain a one:

**Figure 184**

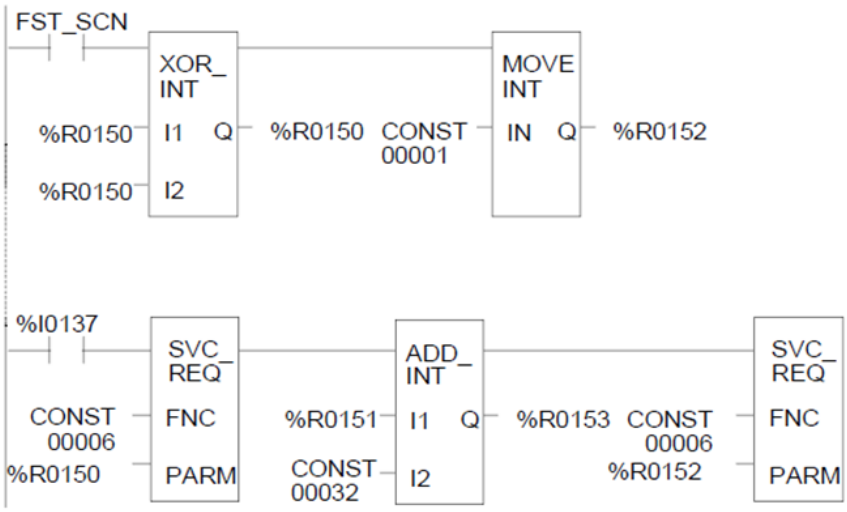
| | |
|---|---|
| address | 1 (change word count) |
| address + 1 | new word count (0 to 32) |

The PLC will change the number of words to be checksummed to the new value.

### Example of SVCREQ 6

In the example, when enabling contact FST_SCN is set, the parameter blocks for the checksum function are built. Later in the program, if input %I0137 turns on, the SVCREQ reads the number of words being checksummed. The parameter block for the Read function is located at %R0150-151. The ADD function adds 32 to the current word count in %R0151 and places the result in %R0153. The parameter block for the Change function is located at %R00152-153. The second SVCREQ then changes to the new word count specified in %R0153.

**Figure 185**




# 11.8 SVCREQ 7: Read or Change the Time-of-Day Clock

Use SVCREQ 7 to read or change the time of day clock in the PLC. The data can be either BCD or ASCII. Either 2-digit-year or 4-digit-year format is available. The function is successful unless some number other than 0 (read) or 1 (change) is entered for the requested operation, or an invalid data format is specified, or data is provided in an unexpected format.

## 11.8.1 Parameter Block Format for SVCREQ 7

For the date/time functions, the length of the parameter block depends on the data format.

The data block is either BCD or ASCII. BCD format requires 6 words; packed ASCII requires 12 words (13 words for 4-digit year). For both data types:

- Hours are stored in 24-hour format.
- Day of the week is a numeric value from 1 (Sunday) to 7 (Saturday).

| | 2-Digit Year Format | 4-Digit Year Format |
|---|---|---|
| address | 0 = read time and date | 0 = read time and date |
| | 1 = set time and date | 1 = set time and date |
| address + 1 | 1 = BCD format | 81h = BCD format |
| | 3 = packed ASCII format | 83h = packed ASCII format |
| address + 2 to end | data | data |

Words 3 to the end of the parameter block contain output data returned by a read function, or new data being supplied by a change function. In both cases, format of these data words is the same. When reading the date and time, words (address + 2) to the end of the parameter block are ignored on input.

## 11.8.2 SVCREQ 7 Parameter Block Content: BCD Format

In BCD format, each time and date item occupies one byte, so the parameter block has six words.

### 2-Digit Year

The last byte of the sixth word is not used. When setting the date and time, this byte is ignored; when reading date and time, the function returns 00.

| **Parameter Block Format: High Byte: Low Byte** | | | **Example: Read Date and Time in BCD format (Sun., July 3, 1998, at 2:45:30 p.m.)** | |
|---|---|---|---|---|
| 1 = change or 0 = read | | address | 0 (read) | |
| 1 (BCD format) | | address + 1 | 1 (BCD format) | |
| month | year | address + 2 | 07 (July) | 98 (year) |
| hours | day of month | address + 3 | 14 (hours) | 03 (day) |
| seconds | minutes | address + 4 | 30 (seconds) | 45 (minutes) |
| (null) | day of week | address + 5 | 0 | 06 (Friday) |

### 4-Digit Year

The parameter block has six words. All bytes are used.

| **Parameter Block Format: High Byte: Low Byte** | | | **Example: Read Date and Time in BCD format (Sun., July 3, 1998, at 2:45:30 p.m.)** | |
|---|---|---|---|---|
| 1 = change or 0 = read | | address | 0 | 00 (read) |
| 81h (BCD format, 4-digit) | | address + 1 | 0 | 81h (BCD, 4-digit) |
| month | year | address + 2 | 19 (year) | 98 (year) |
| day of month | month | address + 3 | 03 (day) | 07 (July) |
| minutes | hours | address + 4 | 45 (minutes) | 14 (hours) |
| day of week | seconds | address + 5 | 06 (Friday) | 30 (seconds) |

## 11.8.3 SVCREQ 7 Parameter Block Content: Packed ASCII Format

In Packed ASCII format, each digit of the time and date items is an ASCII formatted byte. Spaces and colons are embedded into the data to format it for printing or display. ASCII format requires 12 words in the parameter block (13 words for 4-digit year).

### 2-Digit Year

| **Parameter Block Format: High Byte: Low Byte** | | | **Example: Read Date and Time in BCD format (Sun., July 3, 1998, at 2:45:30 p.m.)** | |
|---|---|---|---|---|
| 1 = change or 0 = read | | address | 0 (read) | |
| 3 (ASCII format) | | address + 1 | 3 (ASCII format) | |
| year | year | address + 2 | 38 (8) | 39 (9) |
| month | (space) | address + 3 | 31 (1) | 20 (space) |
| (space) | month | address + 4 | 20 (space) | 30 (0) |
| day of month | day of month | address + 5 | 35 (5) | 30 (leading 0) |
| hours | (space) | address + 6 | 31 (1) | 20 (space) |
| : | hours | address + 7 | 3A ( : ) | 31 (1) |
| minutes | minutes | address + 8 | 33 (3) | 31 (1) |
| seconds | : | address + 9 | 30 (0) | 3A ( : ) |
| (space) | seconds | address + 10 | 20 (space) | 30 (0) |
| day of week | day of week | address + 11 | 32 (2: Mon.) | 30 (leading 0) |

### 4-Digit Year

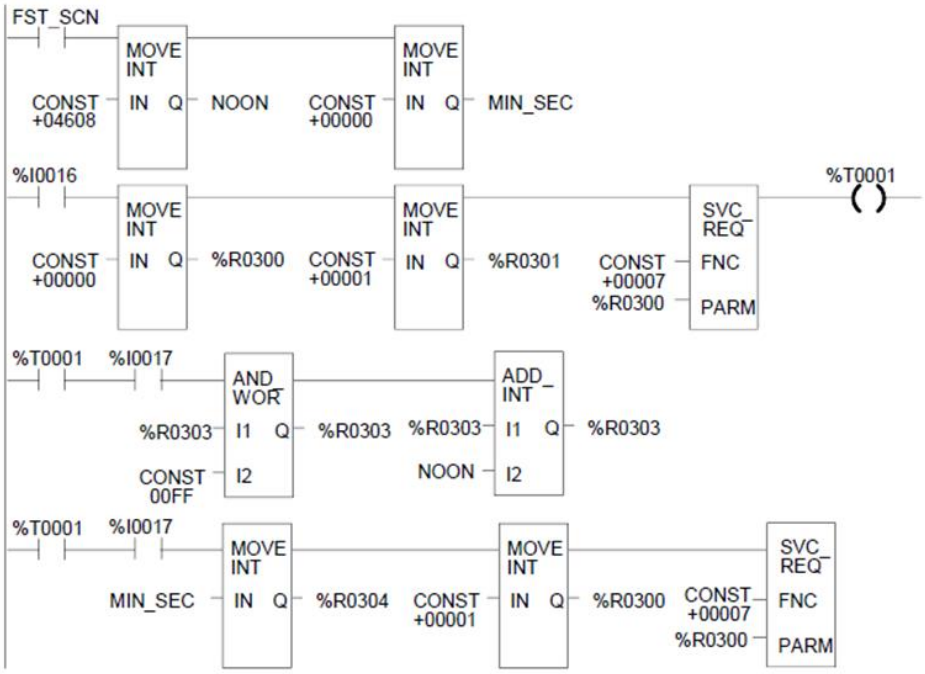
| **Parameter Block Format: High Byte: Low Byte** | | | **Example: Read Date and Time in BCD format (Sun., July 3, 1998, at 2:45:30 p.m.)** | |
|---|---|---|---|---|
| 1 = change or 0 = read | | address | 0 (read) | |
| 83h (ASCII 4 digit) | | address + 1 | 83h (ASCII 4 digit) | |
| year (hundreds) | year (hundreds) | address + 2 | 39 (9) | 31 (1) |
| year (ones) | year (tens) | address + 3 | 38 (8) | 39 (9) |
| month (tens) | (space) | address + 4 | 31 (1) | 20 (space) |
| (space) | month (ones) | address + 5 | 20 (space) | 30 (0) |
| day of month (ones) | day of month (ones) | address + 6 | 35 (5) | 30 (leading 0) |
| hours (tens) | (space) | address + 7 | 31 (1) | 20 (space) |
| : (colon) | hours (ones) | address + 8 | 3A ( : ) | 31 (1) |

| **Parameter Block Format:**<br>**High Byte: Low Byte** | | | **Example:**<br>**Read Date and Time in BCD format (Sun., July 3, 1998, at 2:45:30 p.m.)** | |
|---|---|---|---|---|
| Minutes (ones) | minutes (tens) | address + 9 | 33 (3) | 31 (1) |
| Seconds (tens) | : (colon) | address + 10 | 30 (0) | 3A ( : ) |
| (space) | seconds (ones) | address + 11 | 20 (space) | 30 (0) |
| day of week (ones) | day of week (tens) | address + 12 | 32 (2: Mon.) | 30 (leading 0) |

## Example of SVCREQ 7

In the example, when called for by previous logic, a parameter block for the time-of-day clock is built. It requests the current date and time, then sets the clock to 12 noon using BCD format. The parameter block is located at location %R0300. Array NOON has been set up elsewhere in the program to contain the values 12, 0, and 0. (Array NOON must also contain the data at %R0300.) BCD format requires six contiguous memory locations for the parameter block.

**Figure 186**



# 11.9 SVCREQ 8: Reset Watchdog Timer

Use SVCREQ 8 to reset the watchdog timer during the sweep. Ordinarily, when the watchdog timer expires, the PLC shuts down without warning. SVCREQ 8 allows the timer to keep going during a time-consuming task (for example, while waiting for a response from a communications line)

**⚠CAUTION**

Be sure that resetting the watchdog timer does not adversely affect the controlled process.

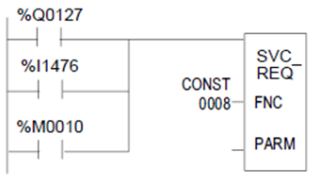
## 11.9.1 Parameter Block Format for SVCREQ 8

This function has no associated parameter block.

## 11.9.2 Example of SVCREQ 8

In this example, power flow through enabling output %Q0027 or input %I1476 or internal coil %M00010 causes the watchdog timer to be reset.

**Figure 187**




# 11.10 SVCREQ 9: Read Sweep Time from Beginning of Sweep

Use SVCREQ 9 to read the time in milliseconds since the start of the sweep. The data format is unsigned 16-bit integer.

## 11.10.1 Output Parameter Block Format for SVCREQ 9

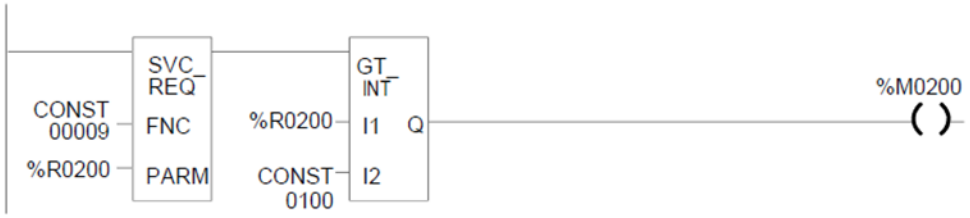
The parameter block is an output parameter block only; it has a length of one word.

| | |
|---|---|
| address | time since start of sweep |

### Example of SVCREQ 9

In the following example, the elapsed time from the start of the sweep is always read into location %R0200. If it is greater than 100ms, internal coil %M0200 is turned on.

**Figure 188**




# 11.11 SVCREQ 10: Read Folder Name

Use SVCREQ 10 to read the name of the currently-executing folder.

## 11.11.1 Output Parameter Block Format for SVCREQ 10

The output parameter block has a length of four words. It returns eight ASCII characters; the last is a null character (00h). If the program name has fewer than seven characters, null characters are added to the end.

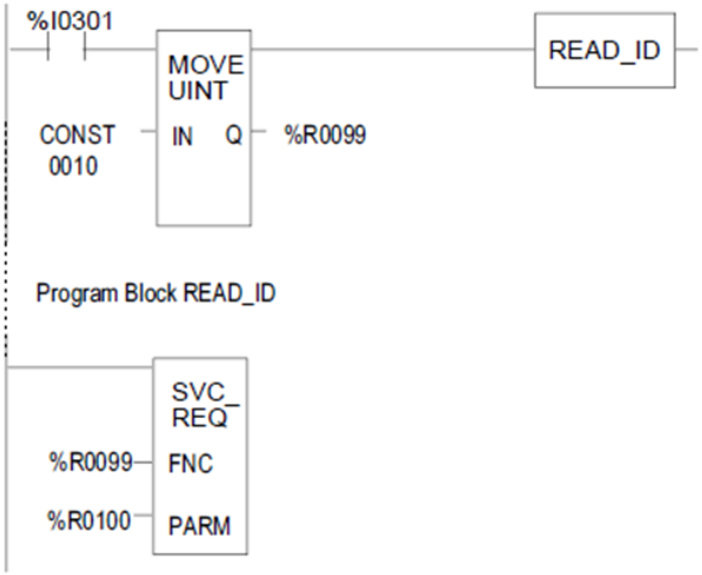
| | Low Byte | High Byte |
|---|---|---|
| address | character 1 | character 2 |
| address + 1 | character 3 | character 4 |
| address + 2 | character 5 | character 6 |
| address + 3 | character 7 | 0 |

## 11.11.2 Example of SVCREQ 10

In this example, when enabling input %I0301 goes OFF, register location %R0099 is loaded with the value 10, which is the function code for the Read Folder Name function.

The Program Block READ_ID is then called to retrieve the folder name. The parameter block is located at address %R0100.

**Figure 189**




# 11.12 SVCREQ 11: Read PLC ID

Use SVCREQ 11 to read the name of the PLC executing the program.

## 11.12.1 Output Parameter Block Format for SVCREQ 11

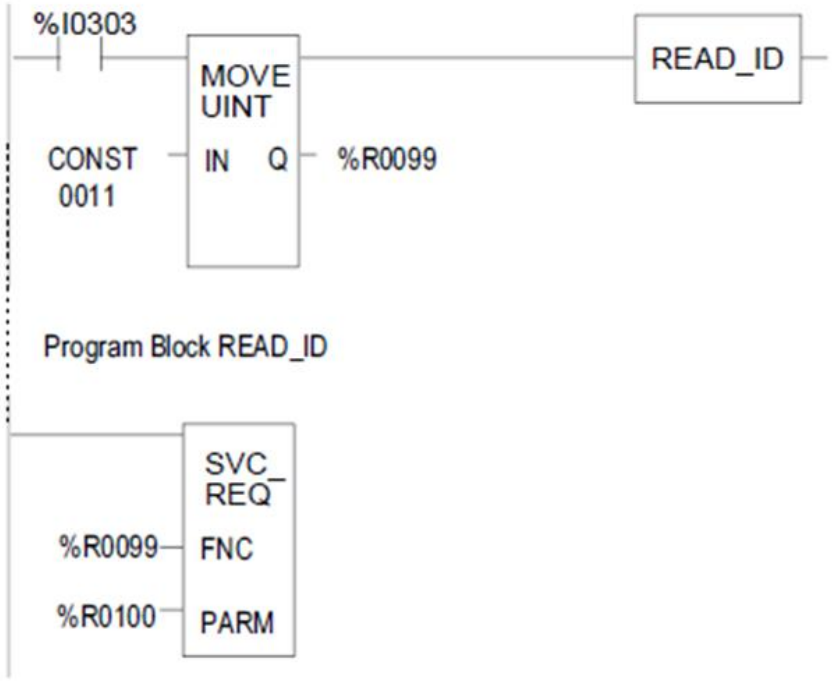
The output parameter block has a length of four words. It returns eight ASCII characters; the last is a null character (00h). If the PLC ID has fewer than seven characters, null characters are added to the end.

| | Low Byte | High Byte |
|---|---|---|
| address | character 1 | character 2 |
| address + 1 | character 3 | character 4 |
| address + 2 | character 5 | character 6 |
| address + 3 | character 7 | 0 |

## 11.12.2 Example of SVCREQ 11

In this example, when enabling input %I0302 goes OFF, register location %R0099 is loaded with the value 11, which is the function code for the Read PLC ID function. The program block READ_ID is then called to retrieve the ID. The parameter block is located at address %R0100.

**Figure 190**




# 11.13 SVCREQ 13: Shut Down (Stop) PLC

Use SVCREQ 13 to stop the PLC at the end of the next sweep. All outputs go to their designated default states at the start of the next PLC sweep. An informational Shut Down PLC fault is placed in the PLC Fault Table. The I/O scan continues as configured.

## 11.13.1 Parameter Block for SVCREQ 13

This function has no parameter block.

## 11.13.2 Example of SVCREQ 13

In the example, when a “Loss of I/O Module" fault occurs, SVCREQ 13 executes. The PARM input is not used.

This example uses a JUMP to the end of the program to force a shutdown if the Shutdown PLC function executes successfully. This JUMP and LABEL are needed because the transition to Stop mode does not occur until the end of the sweep in which the function executes.

**Figure 191**


# 11.14 SVCREQ 14: Clear Fault

Use SVCREQ 14 to clear either the PLC fault table or the I/O fault table. The SVCREQ output is set ON unless some number other than 0 or 1 is entered as the requested operation.

## 11.14.1 Input Parameter Block for SVCREQ 14

For this function, the parameter block has a length of 1 word. It is an input parameter block only. There is no output parameter block.

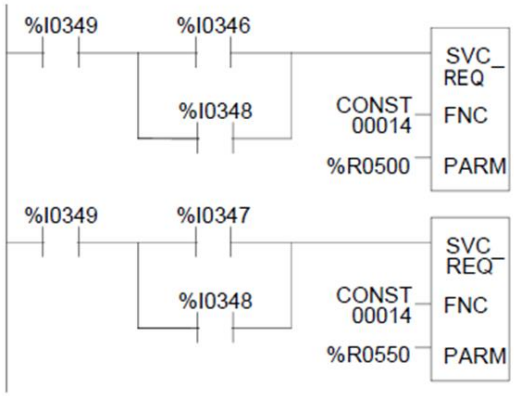
| |
|---|
| 0 = clear PLC fault table<br>1 = clear I/O fault table |

## 11.14.2 Example of SVCREQ 14

In the example, when input %I0346 is on and input %I0349 is on, the PLC fault table is cleared. When input %I0347 is on and input %I0349 is on, the I/O fault table is cleared. When input %I0348 is on and input %I0349 is on, both are cleared.

The parameter block for the PLC fault table is located at %R0500; for the I/O fault table the parameter block is located at %R0550. Both parameter blocks are set up elsewhere in the program.

**Figure 192**




# 11.15 SVCREQ 15: Read Last-Logged Fault Table Entry

Use SVCREQ 15 to read the last entry logged in either the PLC fault table or the I/O fault table. The SVCREQ output is set ON unless some number other than 0 or 1 is entered as the requested operation or the fault table is empty.

## 11.15.1 Input Parameter Block for SVCREQ 15

For this function, the parameter block has a length of 22 words. The input parameter block has this format:

| | 2-Digit Year Format | 4-Digit Year Format |
|---|---|---|
| address | 0 = Read PLC fault table. | 8 = Read PLC fault table. |
| | 1 = Read I/O fault table. | 9 = Read I/O fault table. |

The format of the output parameter block depends on whether the function reads data from the PLC fault table or the I/O fault table.

| **Parameter Block Format:** | | | **I/O Fault Table Output Format** | |
|---|---|---|---|---|
| **High Byte:** | **Low Byte** | | **High Byte:** | **Low Byte** |
| 0 | | | 1 | |
| spare | long/short | address + 1 | memory type | long/short |
| spare | spare | address + 2 | | offset |
| slot | rack | address + 3 | slot | rack |
| | task | address + 4 | block | bus |
| fault action | fault group | address + 5 | | point |
| | error code | address + 6 | fault action | fault group |
| | fault specific data | address + 7 | fault type | fault category |
| | | address + 8 | fault specific data | fault description |
| | | to | | |
| | | address + 18 | | |
| minutes | seconds | address + 19 | minutes | seconds |
| day of month | hour | address + 20 | day of month | hour |

**2-Digit Year Format**

| | | | | |
|---|---|---|---|---|
| year | month | address + 21 | year | month |

**4-Digit Year Format**

| | | | | |
|---|---|---|---|---|
| spare | month | address + 21 | spare | month |
| year | | address + 22 | year | |

## 11.15.2 Long/Short Value

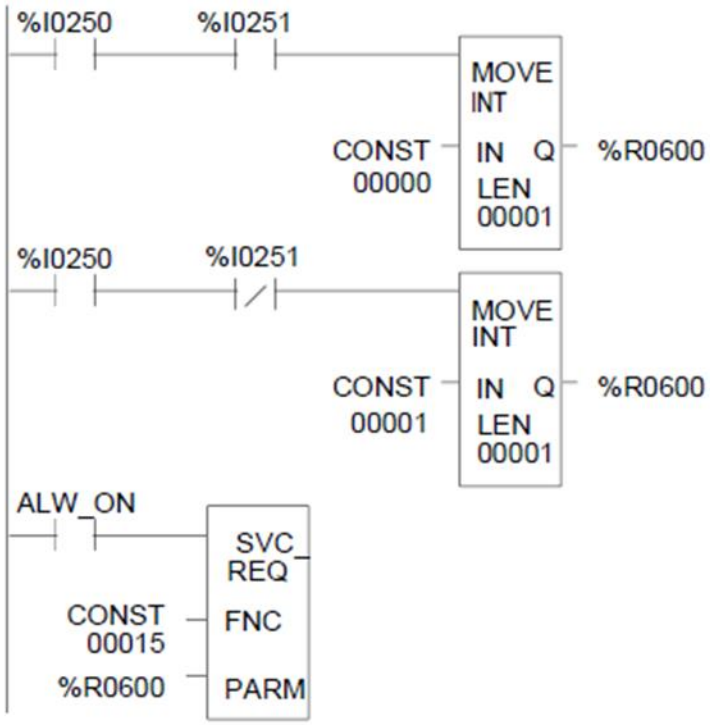
The first byte of word address +1 contains a number that indicates the length of the fault-specific data in the fault entry. These possible values are:

| | |
|---|---|
| PLC fault table | 00 = 8 bytes (short)<br>01 = 24 bytes (long) |
| I/O fault table | 02 = 5 bytes (short)<br>03 = 21 bytes (long) |

## 11.15.3 Example of SVCREQ 15

When inputs %I0250 and %I0251 are both on, the first Move function places a zero (read PLC fault table) into the parameter block for SVCREQ 15.When input %I0250 is on and input %I0251 is off, the Move instruction instead places a one (read I/O fault table) in the SVCREQ parameter block. The parameter block is located at location %R0600.

**Figure 193**




# 11.16 SVCREQ 16: Read Elapsed Time Clock

Use SVCREQ 16 to read the system's elapsed time clock. The elapsed time clock measures the time in seconds since the PLC was powered on.

## 11.16.1 Output Parameter Block for SVCREQ 16

This function has an output parameter block only. Its length is 3 words.

| | |
|---|---|
| address | seconds from power on (low order) |
| address+1 | seconds from power on (high order) |
| address+2 | 100 microsecond ticks |

The first two words are the elapsed time in seconds. The last word is the number of 100 microsecond ticks in the current second.

## 11.16.2 Example of SVCREQ 16

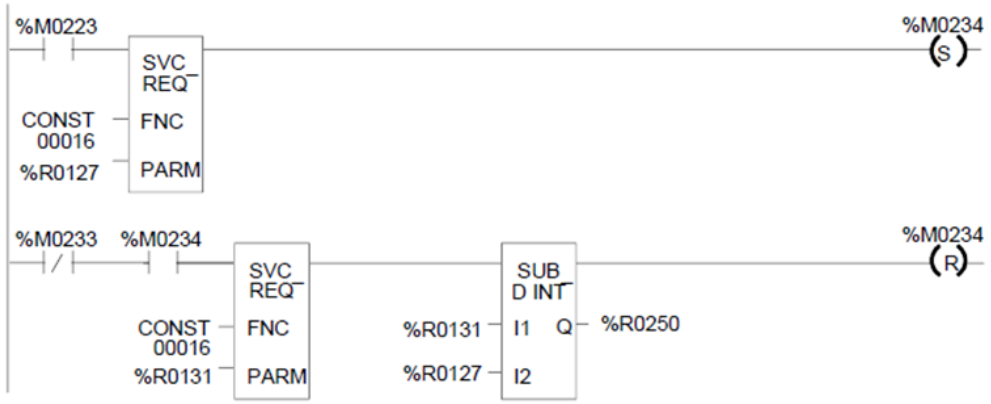
In the example, when internal coil %M0233 is on, the SVCREQ with a parameter block located at %R0127 reads the system's elapsed time clock and sets internal coil %M0234.

When coil %M0233 is off, the SVCREQ with a parameter block at %R0131 reads the elapsed time clock again.

The subtraction function finds the difference between the first and second readings, which have been stored in the SVCREQ parameter blocks. The subtraction ignores the hundred microsecond ticks.

The difference between the two readings is placed in memory location %R0250.

**Figure 194**




# 11.17 SVCREQ 18: Read I/O Override Status

Use SVCREQ 18 to check for any overrides in the CPU's %I and %Q memories.

## 11.17.1 Output Parameter Block for SVCREQ 18

This function has an output parameter block only. Its length is 1 word.

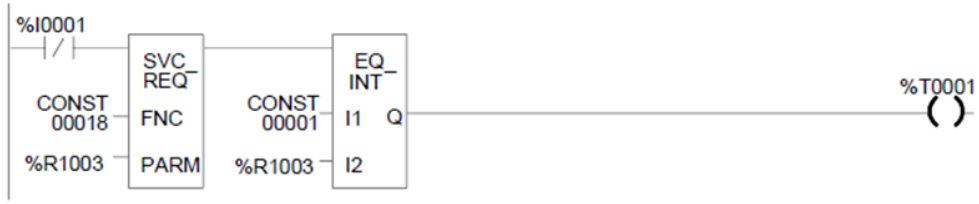
| | |
|---|---|
| address | 0 = No overrides are set<br>1 = Overrides are set. |

## 11.17.2 Example of SVCREQ 18

The following SVCREQ reads the status of I/O overrides memory into location %R1003.

The equality function checks %R1003 to see if it is equal to (the constant) 1. If it is, the equality function turns on output %T0001.

**Figure 195**



# 11.18 SVCREQ 23: Read Master Checksum

Use SVCREQ 23 to read the master checksums of the application program and the configuration. The SVCREQ output is always ON if the function is enabled.

## 11.18.1 Output Parameter Block for SVCREQ 23

For this function, the output parameter block has a length of 12 words with this format:

The first two items in the output parameter block indicate when the program and configuration checksums are valid. (Program checksums may not be valid during a Run Mode Store.)

| | |
|---|---|
| address | Master Program Checksum Valid (0 = not valid, 1 = valid) |
| address + 1 | Master Configuration Checksum Valid (0 = not valid, 1 = valid) |
| address + 2 | Number of Program Blocks (including _MAIN) |
| address + 3 | Size of User Program in Bytes (DWORD data type) |
| address + 5 | Program Additive Checksum |
| address + 6 | Program CRC Checksum (DWORD data type) |
| address + 8 | Size of Configuration Data in Bytes |
| address + 9 | Configuration Additive Checksum |
| address + 10 | Configuration CRC Checksum (DWORD data type) |

## 11.18.2 Example of SVCREQ 23

In the example, when input %I0251 is ON, the master checksum information is placed into the parameter block at %R0050 and the output coil (%Q0001) is turned on.

**Figure 196**

# 11.19 SVCREQ 24: Reset Ethernet Daughter Board

Service request #24 is added for user to reset Ethernet daughterboard from ladder. The FNC number “24” is the instruction number to reset Ethernet daughterboard on IC200CPUE05. The first WORD in the parameter block for the SVC_REQ #24 should be “0” to reset Ethernet daughterboard. Other values are ignored and SVC_REQ #24 won’t take effect or pass power flow.

**Figure 197**


*Note:* *It is important to invoke SVC_REQ #24 for Ethernet daughterboard for only one sweep at a time. Each time this function executes, the target daughterboard will be reset regardless of whether it has finished starting up from a previous reset. If multiple reset requests (for example, push button to reset Ethernet) occur simultaneously, Ethernet daughterboard will be reset only once rather than multiple times.*

SVC_REQ #24 is useful to recover Ethernet interface from failures caused by heavy traffic or high bit-error-rate. PLC fault table and LAN status bits below can be used to determine whether the Ethernet interface on IC200CPUE05 is healthy or faulted. The meaning of each individual status bit can be found in chapter 13 of GFK-1503 VersaMax PLC User Manual. Users are advised to use SVC_REQ #24 to reset Ethernet interface when any of the following events occurs:

- In PLC fault table a “loss of Ethernet interface” fault is logged.
- In PLC fault table a “loss of Ethernet interface” fault is logged.
- When bit 15 (LAN Interface Lockup) is set, Ethernet has encountered a lockup problem. A reset is needed.

| Status Bits | Brief Description |
|---|---|
| 1–2 | Reserved, always 0 |
| 3 | Full-duplex |
| 4-12 | Reserved, always 0 |
| 13 | LAN OK |
| 14 | Resource problem |
| 15 | LAN Interface lockup |
| 16 | LAN Interface OK |
| 17-80 | Reserved |

# 11.20 SVCREQ 26/30: Interrogate I/O

Use SVCREQs 26 and 30 to check whether the installed modules match the software configuration. If not, these SVCREQs place appropriate addition, loss, and mismatch faults in the PLC and/or I/O fault tables. SVCREQs 26 and 30 both perform the same function.

The more configuration faults there are, the longer it takes these SVCREQs longer to execute.

These SVCREQs have no parameter block. They always output power flow.

## 11.20.1 Example of SVCREQ 26

In the example, when input %I0251 is ON, the SVCREQ checks the installed modules and compares them to the software configuration. Output %Q0001 is turned on after the SVCREQ is complete.

**Figure 198**


# 11.21 SVCREQ 29: Read Elapsed Power Down Time

Use SVCREQ 29 to read the amount of time elapsed between the last power-down and the most recent powerup. If the watchdog timer expired before power-down, the PLC is not able to calculate the power down elapsed time, so the time is set to 0.

The SVCREQ output is always ON.

## 11.21.1 Output Parameter Block for SVCREQ 29

This function has an output parameter block only. The parameter block has a length of 3 words.

| | |
|---|---|
| address | Power-Down Elapsed Seconds (low order) |
| address + 1 | Power-Down Elapsed Seconds (high order) |
| address + 2 | zero |

The first two words are the power-down elapsed time in seconds. The last word is always 0.

## 11.21.2 Example of SVCREQ 29

In the example, when input %I0251 is ON, the Elapsed Power-Down Time is placed into the parameter block that starts at %R0050. The output coil (%Q0001) is turned on.

**Figure 199**

# Chapter 12: Serial I/O / SNP / RTU Protocols

This chapter describes the VersaMax™ CPU's Serial I/O feature, which can be used to control the read/write activities of one of the CPU ports directly from the application program.

This chapter also contains instructions for using COMMREQs to configure the CPU serial ports for SNP, RTU, or Serial I/O protocol.

- Configuring Serial Ports Using the COMMREQ Function
- Configuring Serial Ports Using the COMMREQ Function
  - RTU Slave/SNP Slave Operation with a Programmer Attached
  - COMMREQ Command Block for Configuring SNP Protocol
  - COMMREQ Data Block for Configuring RTU Protocol
  - COMMREQ Data Block for Configuring Serial I/O
- Serial I/O COMMREQ Commands
  - Initialize Port
  - Set Up Input Buffer
  - Flush Input Buffer
  - Read Port Status
  - Write Port Control
  - Cancel Operation
  - Autodial
  - Write Bytes
  - Read Bytes
  - Read String

Details of RTU and SNP protocol are described in the Serial Communications User Manual (GFK-0582).

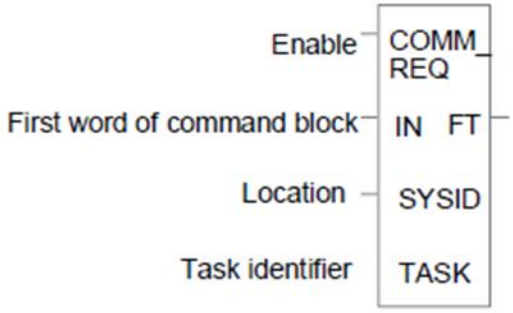
## 12.1 Format of the Communication Request Function

Serial I/O is implemented through the use of Communication Request (COMMREQ) functions. The operations of the protocol, such as transmitting a character through the serial port or waiting for an input character, are implemented through the COMMREQ function block. In CPUE05, Serial I/O is not available for Port 1 when that port is configured or forced for Station Manager operation.

The COMMREQ requires that all its command data be placed in the correct order (in a command block) in the CPU memory before it is executed. The COMMREQ should then be executed by a contact of a one-shot coil to prevent sending the data multiple times. A series of Block Move (BLKMV) commands should be used to move the words to create a command block in the Register tables.

The COMMREQ function has three inputs and one output. When the function receives power flow, a command block of data is sent to the specified module

**Figure 200**




## 12.1.1 Parameters of the COMMREQ Function

| Input/ Output | Choices | Description |
|---|---|---|
| enable | flow | When the function is energized, the communications request is performed. |
| IN | R, AI, AQ | IN contains the first word of the command block. |
| SYSID | I, Q, M, T, G, R, AI, AQ, constant | SYSID contains the rack number (most significant byte) and slot number (least significant byte) of the target device. For the CPU, SYSID must specify rack/slot 0. |
| TASK | R AI, AQ, constant | TASK specifies the port for which the operation is intended:<br>task 19 for port 1 task 20 for port 2 |
| FT | flow, none | FT is energized if an error is detected processing the COMMREQ:<br>• The specified target address is not present (SYSID).<br>• The specified task is not valid for the device (TASK).<br>• The data length is 0.<br>• The device's status pointer address (in the command block) does not exist. |

## 12.1.2 Command Block for the COMMREQ Function

The Command Block starts at the reference specified in COMMREQ parameter IN. The length of the Command Block depends on the amount of data sent to the device.

The Command Block contains the data to be communicated to the other device, plus information related to the execution of the COMMREQ. The Command Block has the following structure:

| | |
|---|---|
| address | Length (in words) |
| address + 1 | Wait/No Wait Flag |
| address + 2 | Status Pointer Memory |
| address + 3 | Status Pointer Offset |
| address + 4 | Idle Timeout Value |
| address + 5 | Maximum Communication Time |
| address + 6 to address + 133 | Data Block |

## 12.1.3 Example of the COMMREQ Function

In the example, when %M0021 is ON, a Command Block located starting at %R0032 is sent to port 2 (communications task 20) of the CPU (rack 0, slot 0). If an error occurs processing the COMMREQ, %Q0110 is set.

**Figure 201**

# 12.2 Configuring Serial Ports Using the COMMREQ Function

The following tables list the command block values required for setting up a Serial Port for SNP, RTU, and Serial I/O. All values are in hexadecimal unless otherwise indicated.

The BLKMV commands that are used to create the command block are described in the example.

It is important to note that 2 parameters have been added to the RTU and Serial IO port configuration COMMREQ, receive to transmit delay and RTS drop delay. When these parameters are included in a COMMREQ the data block length must be set to 12H. If a value of 10H is used, the COMMREQ will still be processed however the receive to transmit and RTS drop delays would not be recognized. It is also important to note that if a COMMREQ containing the receive to transmit delay and RTS drop delay is sent to a CPU that does not support these delay features the CPU will accept and process the COMMREQ but will ignore the receive to transmit, RTS drop delay, and turnaround delay (i.e. turn around delay will be ignored only for the RTU and Serial IO protocols in this case).

Note Either the old form (length 10H) or the new form (length 12H) of the COMMREQ can be used. Only the new form supports the new parameters.

***Note:*** *Either the old form (length 10H) or the new form (length 12H) of the COMMREQ can be used. Only the new form supports the new parameters.*

## 12.2.1 Timing

If a port configuration COMMREQ is sent to a serial port that currently has an SNP/SNPX master (for example, the programmer) connected to it, the serial port configuration specified by the COMMREQ does not take effect until the CPU detects a loss of the SNP/SNPX master. This occurs the configured T3' time after the master disconnects. The COMMREQ status word for the port configuration COMMREQ is updated as soon as the CPU verifies that the specified configuration is valid. That means a COMMREQ Successful value may be returned by the Port Configuration COMMREQ before the specified configuration is actually installed.

## 12.2.2 Sending Another COMMREQ to the Same Port

The application program must wait at least 2 seconds plus the configured T3' time after a new serial port protocol is installed before sending any COMMREQs specific to that protocol to the port. This applies to a new protocol installed by Storing a new hardware configuration or by a port configuration COMMREQ. If the port is configured for Serial I/O, this waiting period must also follow any Stop to Run mode transition of the CPU.

## 12.2.3 Invalid Port Configuration Combinations

The configurations of both ports must be compatible. One port must be available for PLC programmer connection.

The CPU rejects the following combinations:

| Port 1 | Port 2 |
|---|---|
| Disabled | Disabled |
| Disabled | Serial I/O<br>(CPU Run/Stop switch disabled) |
| Serial I/O<br>(CPU Run/Stop switch disabled) | Disabled |
| Serial I/O<br>(CPU Run/Stop switch disabled) | Serial I/O<br>(CPU Run/Stop switch disabled) |
| Station Manager | Disabled |
| Station Manager | Serial I/O<br>(CPU Run/Stop switch disabled) |

## 12.2.4 12.2.4 RTU Slave/SNP Slave Operation With Programmer Attached

A programmer (an SNP/SNPX device) can be attached to port 1 or port 2 while RTU Slave mode is active on the port. For multi-drop connections, the CPU must have been configured to use an appropriate PLC ID. Note that for a multi-drop SNP connection with the port currently configured for RTU, the SNP ID associated with the CPU settings must match the multi-drop ID.

The programmer must use the same serial communications parameters (baud rate, parity, stop bits, etc.) as the currently-active RTU Slave protocol for it to be recognized.

When the CPU recognizes the programmer, the CPU removes the RTU Slave protocol from the port and installs SNP Slave as the currently-active protocol. The SNP ID, modem turnaround time, and default idle time for this new SNP Slave session are obtained from the configured CPU settings, not the port 1 or port 2 configurations.

Connection should be established within 12 seconds. When the programmer connection has been enabled, normal programmer communications can take place. (Failure of the programmer to establish communications within 12 seconds is treated as a Loss of Programmer Communications).

The programmer may send a new protocol via configuration or a Serial Port Setup COMMREQ. (COMMREQs not supported by SNP Slave protocol are rejected). If a new protocol is received, it will not take effect until the programmer is disconnected.

After the programmer is removed, there is a slight delay (equal to the configured SNP T3' timeout) before the CPU recognizes its absence. During this time, no messages are processed on the port. The CPU detects removal of the programmer as an SNP Slave protocol timeout. Therefore, it is important to be careful when disabling timeouts used by the SNP Slave protocol.

When the CPU recognizes the disconnect, it reinstalls RTU Slave protocol unless a new protocol has been received. In that case, the CPU installs the new protocol instead.

### Example

1. Port 1 is running RTU Slave protocol at 9600 baud.
2. A programmer is attached to port 1. The programmer is using 9600 baud.
3. The CPU installs SNP Slave on port 1 and the programmer communicates normally.
4. The programmer stores a new configuration to port 1. The new configuration sets the port for SNP Slave at 4800 baud (it will not take effect until the port loses communications with the programmer).
5. When the CPU loses communications with the programmer, the new configuration takes effect.

## 12.2.5 Example COMMREQ Command Block for Configuring SNP Protocol

| | Values | Meaning |
|---|---|---|
| Address | 10H | Data Block Length |
| Address + 1 | 0 = No Wait | WAIT/NOWAIT Flag |
| Address + 2 | 0008 = %R, register memory | Status Word Pointer Memory Type |
| Address + 3 | Zero-based number that gives the address of the COMMREQ status word (for example, a value of 99 gives an address of 100 for the status word) | Status Word Pointer Offset |
| Address + 4 | 0 (Only used in Wait/No Wait mode) | Idle Timeout Value |
| Address + 5 | 0 (Only used in Wait/No Wait mode) | Maximum Communication Time |
| Address + 6 | FFF0H | Command Word (serial port setup) |
| Address + 7 | 0001 | Protocol: 1=SNP |
| Address + 8 | 0000=Slave | Port Mode |

| | Values | Meaning |
|---|---|---|
| Address + 9 | 7=38400, 6=19200, 5=9600, 4=4800 | Data Rate |
| Address + 10 | 0 = None, 1 = Odd, 2 = Even | Parity |
| Address + 11 | 1 = None | Flow Control |
| Address + 12 | 0 = None, 1 = 10ms, 2 = 100ms, 3 = 500ms | Turnaround Delay |
| Address + 13 | 0 = Long, 1 = Medium, 2 = Short, 3 = None | Timeout |
| Address + 14 | 1 = 8 bits | Bits Per Character |
| Address + 15 | 0 = 1 Stop Bit, 1 = 2 Stop bits | Stop Bits |
| Address + 16 | not used | Interface |
| Address + 17 | not used | Duplex Mode |
| Address + 18 | user-provided* | Device identifier bytes 1 and 2 |
| Address + 19 | user-provided* | Device identifier bytes 3 and 4 |
| Address + 20 | user-provided* | Device identifier bytes 5 and 6 |
| Address + 21 | user-provided* | Device identifier bytes 7 and 8 |

* The device identifier for SNP Slave ports is packed into words with the least significant character in the least significant byte of the word. For example, if the first two characters are “A” and “B,” the Address + 18 will contain the hex value 4241.

## 12.2.6 Example COMMREQ Data Block for Configuring RTU Protocol

| | Values | Meaning |
|---|---|---|
| First 6 words | | Reserved for COMMREQ use. |
| Address + 6 | FFF0H | Command |
| Address + 7 | 003 | Protocol: 0003=RTU |
| Address + 8 | 000 | Port Mode: 0000=Slave |
| Address + 9 | 2=1200, 3-2400, 4=4800, 5=9600, 6=19200,7=38400*, 8=57600**<br>*CPU models IC200CPU005 and CPUE05 only | Data Rate |

| | Values | Meaning |
|---|---|---|
| Address + 10 | 0 = None, 1 = Odd, 2 = Even | Parity |
| Address + 11 | 0 = Hardware, 1 = None | Flow Control |
| Address + 12 | 0-255 (units of 10ms, e.g. 10=100ms) | Turnaround delay |
| Address + 13 | not used | Timeout |
| Address + 14 | not used | Bits per Character |
| Address + 15 | not used | Stop Bits |
| Address + 16 | not used | Interface |
| Address + 17 | 0 = 2-wire, 1 = 4-wire | Duplex Mode |
| Address + 18 | Station Address (1-247) | Device Identifier |
| Address + 19–21 | not used | Device Identifier |
| Address + 22* | 0-255 (units of 10ms, e.g. 10=100ms) | Receive to transmit delay |
| Address + 22* | 0-255 (units of 10ms, e.g. 10=100ms) | RTS drop delay |

**Note:** *The data block length (Address + 0) for a COMMREQ that includes the Receive to transmit delay and RTS drop delay should be 12H not 10H. Both forms (Length 10H and 12H) are supported.*

**Note:** *If RTU is configured for 115.2K baud. a major error code 12 (0cH) and a minor error code 2 (02H) is returned in the COMMREQ status word. This will occur for any unsupported baud rate.*

## 12.2.7 Example COMMREQ Data Block for Configuring Serial I/O Protocol

| | Values | Meaning |
|---|---|---|
| First 6 words | | Reserved for COMMREQ use. |
| Address + 6 | FFF0H | Command |
| Address + 7 | 005 | Protocol: 0005=Serial IO |
| Address + 8 | 0 = Slave | Port Mode |
| Address + 9 | 4=4800, 5=9600, 6=19200, 7= 38400*, 8=57600*** | Data Rate |
| | *CPU models IC200CPU005 and CPUE05 only | |
| Address + 10 | 0 = None, 1 = Odd, 2 = Even | Parity |
| Address + 11 | 0 = Hardware, 1 = None | Flow Control |
| Address + 12 | 0-255 (units of 10ms, e.g. 10=100ms) | Turnaround delay |
| Address + 13 | 0 = Long | Timeout |
| Address + 14 | 0=7 bits, 1=8 bits | Bits per Character |
| Address + 15 | 0 = 1 stop bit, 1 = 2 stop bits | Stop Bits |
| Address + 16 | not used | Interface |
| Address + 17 | 0 = 2-wire, 1 = 4-wire | Duplex Mode |
| Address + 18-21 | not used | Device Identifier |
| Address + 22* | 0-255 (units of 10ms, e.g. 10=100ms) | Receive to transmit delay |
| Address + 22* | 0-255 (units of 10ms, e.g. 10=100ms) | RTS drop delay |

***Note:*** *The data block length (Address + 0) for a COMMREQ that includes the Receive to transmit delay and RTS drop delay should be 12H not 10H. Both forms (Length 10H and 12H) are supported.*

***Note:*** *If Serial I/O is configured for 115.2K baud. a major error code 12 (0cH) and a minor error code 2 (02H) is returned in the COMMREQ status word. This will occur for any unsupported baud rate.*

# 12.3 Calling Serial I/O COMMREQs from the PLC Sweep

Implementing a serial protocol using Serial I/O COMMREQs may be restricted by the PLC sweep time. For example, if the protocol requires that a reply to a certain message from the remote device be initiated within 5mS of receiving the message, this method may not be successful if the PLC sweep time is 5mS or longer, since timely response is not guaranteed.

Since the Serial I/O is completely driven by the application program, in STOP mode a port configured as Serial I/O automatically reverts to SNP slave, to facilitate programmer communication. Therefore, while in Stop mode, Serial I/O protocol is not active; it is only active when the PLC is in Run mode.

When the port reverts back to SNP Slave, the same serial communications parameters (baud rate, parity, stop bits ...) as the currently-active Serial I/O protocol are used.

Therefore the programmer must use the same parameters for it to be recognized. If any of the parameter values associated with the Serial I/O protocol are not supported by the SNP Slave protocol, the programmer will not be able to communicate with the PLC via that port.

## 12.3.1 Compatibility

The COMMREQ function blocks supported by Serial I/O are not supported by other currently-existing protocols (such as SNP slave, SNP master, and RTU slave). Errors are returned if they are attempted for a port configured for one of those protocols.

## 12.3.2 Status Word for Serial I/O COMMREQs

A value of 1 is returned in the COMMREQ status word upon successful completion of the COMMREQ. Any other value returned is an error code where the low byte is a major error code and the high byte is a minor error code.

| Major Error Code | Description | |
|---|---|---|
| 1 (01h) | Successful Completion (this is the expected completion value in the COMMREQ status word). | |
| 12 (0Ch) | Local error —Error processing a local command. The minor error code identifies the specific error. | |
| | 1 (01h) | Wait-type command is not permitted. Use No-Wait command. |
| | 2 (02h) | COMMREQ command is not supported. |
| | 5 (05h) | Error writing COMMREQ status word to PLC memory. |
| | 6 (06h) | Invalid PLC memory type specified. |
| | 7 (07h) | Invalid PLC memory offset specified. |
| | 8 (08h) | Unable to access PLC memory. |
| | 9 (09h) | Data length exceeded. |
| | 12 (0Ch) | COMMREQ data block length too small. |
| | 14 (0Eh) | COMMREQ data is invalid. |
| | 15 (0Fh) | Could not allocate system resources to complete COMMREQ. |
| 13 (0Dh) | Remote error — Error processing a remote command. The minor error code identifies the error. | |
| | 2 (02h) | Number of bytes requested to read is greater than input buffer size OR number bytes requested to write is zero or greater than 250 bytes. |
| | 3 (03h) | COMMREQ data block length is too small. String data is missing or incomplete. |
| | 4 (04h) | Receive timeout awaiting serial reception of data |
| | 8 (08h) | Unable to access PLC memory. |
| | 12 (0Ch) | COMMREQ data block length too small. |
| | 48 (30h) | Serial output timeout. The serial port was unable to transmit the string. (Could be due to missing CTS signal when the serial port is configured to use hardware flow control.) |
| | 50 (32h) | COMMREQ timeout. The COMMREQ did not complete within a 20-second time limit. |
| 14 (0Eh) | Autodial Error — An error occurred while attempting to send a command string to an attached external modem. The minor error code identifies the specific error. | |
| | 1 (01h) | Not used. |
| | 2 (02h) | The modem command string length exceeds end of reference memory type. |
| | 3 (03h) | COMMREQ Data Block Length too small. Output command string data missing or incomplete. |
| | 4 (04h) | Serial output timeout. The serial port was unable to transmit the modem autodial output. |
| | 5 (05h) | Response was not received from modem. Check modem and cable. |

| Major Error Code | Description | |
|---|---|---|
| | 6 (06h) | Modem responded with BUSY. Modem is unable to complete the requested connection. The remote modem is already in use; retry the connection request later. |
| | 7 (07h) | Modem responded with NO CARRIER. Modem is unable to complete the requested connection. Check the local and remote modems and the telephone line. |
| | 8 (08h) | Modem responded with NO DIALTONE. Modem is unable to complete the requested connection. Check the modem connections and the telephone line. |
| | 9 (09h) | Modem responded with ERROR. Modem is unable to complete the requested command. Check the modem command string and modem. |
| | 10 (0Ah) | Modem responded with RING, indicating that the modem is being called by another modem. Modem is unable to complete the requested command. Retry the modem command later. |
| | 11 (0Bh) | Unknown response received from the modem. Modem unable to complete the request. Check the modem command string and modem. Response should be CONNECT or OK. |
| | 50 (32h) | COMMREQ timeout. The COMMREQ did not complete within a 20-second time limit. |

## 12.4 Serial I/O COMMREQ Commands

The following COMMREQs are used to implement Serial I/O:

- Local COMMREQs - do not receive or transmit data through the serial port.
  - Initialize Port (4300)
  - Set Up Input Buffer (4301)
  - Flush Input Buffer (4302)
  - Read Port Status (4303)
  - Write Port Control (4304)
  - Cancel Operation (4399)
- Remote COMMREQs - receive and/or transmit data through the serial port.
  - Autodial (4400)
  - Write bytes (4401)
  - Read bytes (4402)
  - Read String (4403)

## 12.4.1 Overlapping COMMREQs

Some of the Serial I/O COMMREQs must complete execution before another COMMREQ can be processed. Others can be left pending while others are executed.

## 12.4.2 COMMREQS that Must Complete Execution

- Autodial (4400)
- Initialize Port (4300)
- Set Up Input Buffer (4301)
- Flush Input Buffer (4302)
- Read Port Status (4303)
- Write Port Control (4304)
- Cancel Operation (4399)
- Serial Port Setup (FFF0)

## 12.4.3 COMMREQs that Can be Pending While Others Execute

The table below shows whether Write Bytes, Read Bytes and Read String COMMREQs can be pending when other COMMREQs are executed.

| **Currently pending COMMREQs** | **NEW COMMREQ** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Autodial (4400)** | **Write Bytes (4401)** | **Initialize Port (4300)** | **Set Up Input Buffer (4301)** | **Flush Input Buffer (4302)** | **Read Port Status (4303)** | **Write Port Control (4304)** | **Read Bytes (4402)** | **Read String (4403)** | **Cancel Operation (4399)** | **Serial Port Setup (FFF0)** |
| Write Bytes (4401) | No | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No |
| Read Bytes (4402) | No | Yes | Yes | No | No | Yes | Yes | No | No | Yes | No |
| Read String (4403) | No | Yes | Yes | No | No | Yes | Yes | No | No | Yes | No |

## 12.4.4 Initialize Port Function (4300)

This function causes a reset command to be sent to the specified port. It also cancels any COMMREQ currently in progress and flushes the internal input buffer. RTS is set to inactive.

## Example Command Block for the Initialize Port Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0001 | 0001 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4300 | 10CC | Initialize port command |

## Operating Notes

*Note:* *COMMREQs that are cancelled due to this command executing do not have their respective COMMREQ status words updated.*

**⚠CAUTION**

If this COMMREQ is sent when a Write Bytes (4401) COMMREQ is transmitting a string from a serial port, transmission is halted. The position within the string where the transmission is halted is indeterminate. In addition, the final character received by the device the CPU is sending to is also indeterminate.

# 12.4.5 Set Up Input Buffer Function (4301)

This function can be used to change the size of the internal memory buffer where input data will be placed as it is received. By default, the buffer is set to a maximum of 2K bytes. As data is received from the serial port it is placed in the input buffer. If the buffer becomes full, any additional data received from the serial port is discarded and the Overflow Error bit in the Port Status word (See Read Port Status Function) is set.

## Retrieving Data from the Buffer

Data can be retrieved from the buffer using the Read String or Read Bytes function. It is not directly accessible from the application program.

If data is not retrieved from the buffer in a timely fashion, some characters may be lost.

### Example Command Block for the Set Up Input Buffer Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0002 | 0002 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4301 | 10CD | Setup input buffer command |
| address +7 | 0064 | 0040 | Buffer length (in words) |

### Operating Notes

It is not possible to set the buffer length to zero. If zero is entered as the buffer length, the buffer size will be set to the 2K bytes default.

If a length greater than 2K bytes is specified, an error is generated.

## 12.4.6 Flush Input Buffer Function (4302)

This operation empties the input buffer of any characters received through the serial port but not yet retrieved using a read command. All such characters are lost.

### Example Command Block for the Flush Input Buffer Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0001 | 0001 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4302 | 10CE | Setup input buffer command |

## 12.4.7 Read Port Status Function (4303)

This function returns the current status of the port. The following events can be detected:

- A read request was initiated previously, and the required number of characters has now been received or the specified time-out has elapsed.
- A write request was initiated previously and transmission of the specified number of characters is complete or a time-out has elapsed.

The status returned by the function indicates the event or events that have completed.

More than one condition can occur simultaneously, if both a read and a write were initiated previously.

### Example Command Block for the Read Port Status Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0003 | 0003 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4303 | 10CF | Setup input buffer command |
| address +7 | 0076 | 004C | Port status memory type (%M) |
| address +8 | 0101 | 065 | Port status memory offset (%M101) |

### Port Status

The port status consists of a status word and the number of characters in the input buffer that have not been retrieved by the application (characters which have been received and are available).

| | |
|---|---|
| word 1 | Port status word (see below) |
| word 2 | Characters available in the input buffer |

The Port Status Word can be:

| Bit | Name | Definition | Meaning | |
|---|---|---|---|---|
| 15 | RI | Read In progress | Set | Read Bytes or Read String invoked |
| | | | Cleared | Previous Read bytes or String has timed out, been canceled, or finished |
| 14 | RS | Read Success | Set | Read Bytes or Read String has successfully completed |

| Bit | Name | Definition | Meaning | |
|---|---|---|---|---|
| | | | Cleared | New Read Bytes or Read String invoked |
| 13 | RT | Read Time-out | Set | Receive timeout occurred during Read Bytes or Read String |
| | | | Cleared | New Read Bytes or Read String invoked |
| 12 | WI | Write In progress | Set | New Write Bytes invoked |
| | | | Cleared | Previously-invoked Write Bytes has timed out, been canceled, or finished |
| 11 | WS | Write Success | Set | Previously-invoked Write Bytes has successfully completed |
| | | | Cleared | New Write Bytes invoked |
| 10 | WT | Write Time-out | Set | Transmit timeout occurred during Write Bytes |
| | | | Cleared | New Write Bytes invoked |
| 9 | CA | Character Available | Set | Unread characters are in the buffer |
| | | | Cleared | No unread characters in the buffer |
| 8 | OF | OverFlow error | Set | Overflow error occurred on the serial port or internal buffer |
| | | | Cleared | Read Port Status invoked |
| 7 | FE | Framing Error | Set | Framing error occurred on the serial port |
| | | | Cleared | Read Port Status invoked |
| 6 | PE | Parity Error | Set | Parity error occurred on the serial port |
| | | | Cleared | Read Port Status invoked |
| 5 | CT | CTS is active | Set | CTS line on the serial port is active or the serial port does not have a CTS line |
| | | | Cleared | CTS line on the serial port is not active |
| 4 - 0 | U | not used, should be 0 | | |

## 12.4.8 Write Port Control Function (4304)

This function forces RTS for the specified port:

### Example Command Block for the Write Port Control Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0002 | 0002 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4304 | 10D0 | Write port control command |
| address +7 | xxxx | xxxx | Port control word |

## Port Control Word

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTS | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

The Port Control Word can be:

15 RTS Commanded state of the RTS output

1 = Activates RTS

0 = Deactivates RTS

0-14 U Unused (should be zero)

## Operating Note

For CPU port 2 (RS-485), the RTS signal is also controlled by the transmit driver. Therefore, control of RTS is dependent on the current state of the transmit driver. If the transmit driver is not enabled, asserting RTS with the Write Port Control COMMREQ will not cause RTS to be asserted on the serial line. The state of the transmit driver is controlled by the protocol and is dependent on the current Duplex Mode of the port. For 2-wire and 4-wire Duplex Mode, the transmit driver is only enabled during transmitting. Therefore, RTS on the serial line will only be seen active on port 2 (configured for 2-wire or 4-wire Duplex Mode) when data is being transmitted. For point-to-point Duplex Mode, the transmit driver is always enabled. Therefore, in point-to-point Duplex Mode, RTS on the serial line will always reflect what is chosen with the Write Port Control COMMREQ.

# 12.4.9 Cancel Commreq Function (4399)

This function cancels the current operations in progress. It can be used to cancel both read operations and write operations.

If a read operation is in progress and there are unprocessed characters in the input buffer, those characters are left in the input buffer and available for future reads. The serial port is not reset.

## Example Command Block for the Cancel Operation Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0002 | 0002 | Data block length (2) |
| address +1 | 0000 | 0000 | NOWAIT mode |

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4399 | 112F | Cancel operation command |
| address +7 | 0001 | 0001 | Transaction type to cancel<br>• 1 All operations<br>• 2 Read operations<br>• 3 Write operations |

## Operating Notes

This function does not update the status of words of the cancelled COMMREQs.

**⚠CAUTION**

Failure to do so may result in injury to personnel or cause damage to the equipment. If this COMMREQ is sent in either Cancel All or Cancel Write mode when a Write Bytes (4401) COMMREQ is transmitting a string from a serial port, transmission is halted. The position within the string where the transmission is halted is indeterminate. In addition, the final character received by the device the CPU is sending to is also indeterminate.

# 12.4.10 Autodial Function (4400)

This feature allows the VersaMax CPU to automatically dial a modem and send a specified byte string.

To implement this feature, the port must be configured for Serial I/O.

For example, pager enunciation can be implemented by three commands, requiring three COMMREQ command blocks:

**Autodial**: 04400 (1130h) Dials the modem.

**Write Bytes:** 04401 (1131h) Specifies an ASCII string, from 1 to 250 bytes in length, to send from the serial port.

**Autodial:** 04400 (1130h) It is the responsibility of the PLC application program to hang up the phone connection. This is accomplished by reissuing the autodial command and sending the hang up command string.

## Autodial Command Block

The Autodial command automatically transmits an Escape sequence that follows the Hayes convention. If you are using a modem that does not support the Hayes convention, you may be able to use the Write Bytes command to dial the modem.

Examples of commonly used command strings for Hayes-compatible modems are listed below:

| Command String | Length | Function |
|---|---|---|
| ATDP15035559999<CR> | 16 (10h) | Pulse dial the number 1-503-555-9999 |
| ATDT15035559999<CR> | 16 (10h) | Tone dial the number 1-503-555-9999 |
| ATDT9,15035559999<CR> | 18 (12h) | Tone dial using outside line with pause |
| ATH0<CR> | 5 (05h) | Hang up the phone |
| ATZ <CR> | 4 (04h) | Restore modem configuration to internally saved values |

## Example Autodial Command Block

This example COMMREQ command block dials the number 234-5678 using a Hayes-compatible modem.

| Word | Definition | Values |
|---|---|---|
| 1 | 0009h | CUSTOM data block length (includes command string) |
| 2 | 0000h | NOWAIT mode |
| 3 | 0008h | Status word memory type (%R) |
| 4 | 0000h | Status word address minus 1 (Register 1) |
| 5 | 0000h | not used |
| 6 | 0000h | not used |
| 7 | 04400 (1130h) | Autodial command number |
| 8 | 00030 (001Eh) | Modem response timeout (30 seconds) |
| 9 | 0012 (000Ch) | Number of bytes in command string |
| 10 | 5441h | A (41h), T (54h) |
| 11 | 5444h | D (44h), T (54h) |
| 12 | 3332h | Phone number: 2 (32h), 3 (33h) |
| 13 | 3534h | 4 (34h), 5 (35h) |
| 14 | 3736h | 6 (36h), 7 (37h) |
| 15 | 0D38h | 8 (38h) <CR> (0Dh) |

# 12.4.11 Write Bytes Function (4401)

This operation can be used to transmit one or more characters to the remote device through the specified serial port. The character(s) to be transmitted must be in a word reference memory . They should not be changed until the operation is complete.

Up to 250 characters can be transmitted with a single invocation of this operation. The status of the operation is not complete until all of the characters have been transmitted or until a timeout occurs (for example, if hardware flow control is being used and the remote device never enables the transmission).

## Example Command Block for the Write Bytes Function

| | VALUE (decimal) | VALUE (hexadecimal | MEANING |
|---|---|---|---|
| address | 0006 | 0006 | Data block length (includes characters to send) |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4401 | 1131 | Write bytes command |
| address +7 | 0030 | 001E | Transmit time-out (30 seconds). See note below. |
| address +8 | 0005 | 0005 | Number of bytes to write |
| address +9 | 25960 | 6568 | 'h' (68h), 'e' (65h) |
| address +10 | 27756 | 6C6C | 'l' (6Ch), 'l' (6Ch) |
| address +11 | 0111 | 006F | 'o' (6Fh) |

Although printable ASCII characters are used in this example, there is no restriction on the values of the characters which can be transmitted.

## Operating Notes

*Note:* *Specifying zero as the Transmit time-out sets the time-out value to the amount of time actually needed to transmit the data, plus 4 seconds.*

**⚠CAUTION**

If an Initialize Port (4300) COMMEQ is sent or a Cancel Operation (4399) COMMREQ is sent in either Cancel All or Cancel Write mode while this COMMREQ is transmitting a string from a serial port, transmission is halted. The position within the string where the transmission is halted is indeterminate. In addition, the final character received by the device the CPU is sending to is also indeterminate.

# 12.4.12 Read Bytes Function (4402)

This function causes one or more characters to be read from the specified port. The characters are read from the internal input buffer and placed in the specified input data area.

The function returns both the number of characters retrieved and the number of unprocessed characters still in the input buffer. If zero characters of input are requested, only the number of unprocessed characters in the input buffer is returned.

If insufficient characters are available to satisfy the request and a non-zero value is specified for the number of characters to read, the status of the operation is not complete until either sufficient characters have been received or the time-out interval expires. In either of those conditions, the port status indicates the reason for completion of the read operation. The status word is not updated until the read operation is complete (either due to timeout or when all the data has been received).

If the time-out interval is set to zero, the COMMREQ remains pending until it has received the requested amount of data, or until it is cancelled.

If this COMMREQ fails for any reason, no data is returned to the buffer. Any data that was already in the buffer remains and can be retrieved with a subsequent read request.

## Example Command Block for the Read Bytes Function

| | VALUE (decimal) | VALUE (hexadecimal | MEANING |
|---|---|---|---|
| address | 0005 | 0005 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4402 | 1132 | Read bytes command |
| address +7 | 0030 | 001E | Read time-out (30 seconds) |
| address +8 | 0005 | 0005 | Number of bytes to read |
| address +9 | 0008 | 0008 | Input data memory type (%R). |
| address +10 | 0100 | 0064 | Input data memory address (%R0100) |

## Return Data Format for the Read Bytes Function

The return data consists of the number of characters actually read, the number of characters still available in the input buffer after the read is complete (if any), and the actual input characters.

| | |
|---|---|
| address | Number of characters actually read |
| address +1 | Number of characters still available in the input buffer, if any |
| address +2 | first two characters (first character is in the low byte) |
| address +3 | third and fourth characters (third character is in the low byte) |
| address +n | subsequent characters |

## Operating Note

If the input data memory type parameter is specified to be a word memory type, and if an odd number of bytes are actually received, then the high byte of the last word to be written with the received data is set to zero.

As data is received from the serial port it is placed in the internal input buffer. If the buffer becomes full, then any additional data received from the serial port is discarded and the Overflow Error bit in the Port Status word (See Read Port Status Function) is set

## 12.4.13 Read String Function (4403)

This function causes characters to be read from the specified port until a specified terminating character is received. The characters are read from the internal input buffer and placed in the specified input data area.

The function returns both the number of characters retrieved and the number of unprocessed characters still in the input buffer. If zero characters of input are requested, only the number of unprocessed characters in the input buffer is returned.

If the terminating character is not in the input buffer, the status of the operation is not complete until either the terminating character has been received or the time-out interval expires. In either of those conditions, the port status indicates the reason for completion of the read operation.

If the time-out interval is set to zero, the COMMREQ remains pending until it has received the requested string, terminated by the specified end character.

If this COMMREQ fails for any reason, no data is returned to the buffer. Any data that was already in the buffer remains and can be retrieved with a subsequent read request.

### Example Command Block for the Read String Function

| | VALUE (decimal) | VALUE (hexadecimal) | MEANING |
|---|---|---|---|
| address | 0005 | 0005 | Data block length |
| address +1 | 0000 | 0000 | NOWAIT mode |
| address +2 | 0008 | 0008 | Status word memory type (%R) |
| address +3 | 0000 | 0000 | Status word address minus 1 (%R0001) |
| address +4 | 0000 | 0000 | Not used |
| address +5 | 0000 | 0000 | Not used |
| address +6 | 4403 | 1133 | Read string command |
| address +7 | 0030 | 001E | Read time-out (30 seconds) |
| address +8 | 0013 | 000D | Terminating character (carriage return): must be between 0 and 255 (0xFF), inclusive |
| address +9 | 0008 | 0008 | Input data memory type (%R) |
| address +10 | 0100 | 0064 | Input data memory address (%R0100) |

## Return Data Format for the Read String Function

The return data consists of the number of characters actually read, the number of characters still available in the input buffer after the read is complete (if any), and the actual input characters:

| | |
|---|---|
| address | Number of characters actually read |
| address +1 | Number of characters still available in the input buffer, if any |
| address +2 | first two characters (first character is in the low byte) |
| address +3 | third and fourth characters (third character is in the low byte) |
| address +n | subsequent characters |

## Operating Note

If the input data memory type parameter is specified to be a word memory type, and if an odd number of bytes are actually received, then the high byte of the last word to be written with the received data is set to zero

As data is received from the serial port it is placed in the internal input buffer. If the buffer becomes full, then any additional data received from the serial port is discarded and the Overflow Error bit in the Port Status word (See Read Port Status Function) is set.

# Chapter 13: Ethernet Communications

This chapter describes the Ethernet communications features of VersaMax® CPU model IC200CPUE05.

- Overview of the Ethernet interface
- IP Addressing
- Routers
- Ethernet Global Data
- Checking the status of an Ethernet Global Data exchange

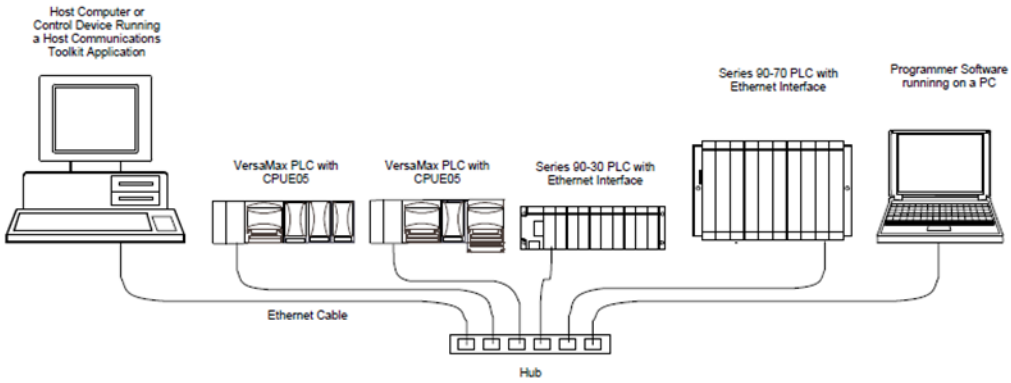
  Diagnostic Tools
- Troubleshooting Common Ethernet Difficulties

## 13.1 Overview of the Ethernet Interface

VersaMax CPU model IC200CPUE05 has a built-in Ethernet interface that makes it possible to communicate on a 10BaseT network in either halfduplex or full-duplex mode.

Using 10/100 hubs allows CPUE05 to communicate on a network containing 100Mb devices.

**Figure 202**




Use the Ethernet interface to:

- **Send and receive Ethernet Global Data**. Ethernet Global Data can be used for highly efficient periodic data transfer on the LAN.
- **Access data from CPUE05 using a Host computer**. Computer applications can access data from CPUE05 through its SRTP server capability.
- **Communicate simultaneously to multiple devices**. The multiplexing capabilities of Ethernet interface, along with Ethernet network's high capacity, allow CPUE05 to communicate with several other devices at the same time.

- **Indirectly attach to other Local Area Networks and/or wide area networks via third party IP routers.** CPUE05 can communicate with remote PLCs and other nodes via an IP Router.
- **Communicate with remote computers via Serial Line Protocol (SLIP) using modems and/or serial lines**. Using third party SLIP devices, a remote host computer can be attached to a TCP/IP network. Once attached, the serial communications can be routed over the Ethernet interface to the CPUE05.
- **Maintain compatibility with other devices**. CPUE05 is compatible with the Emerson Series 90-30 Ethernet Interface, Series 90-30 CPU364 Embedded Ethernet Interface, and Series 90-70 Ethernet Interface (Type 2). It is also compatible with Emerson programming packages supporting TCP/IP Ethernet communications.

### 13.1.1 Ethernet Global Data

CPUE05 also supports up to 32 simultaneous Ethernet Global Data exchanges. Ethernet Global Data exchanges are configured using the PLC programming software, then stored to the PLC. Both Produced and Consumed exchanges may be configured. CPUE05 supports up to 1200 data ranges across all Ethernet Global Data exchanges, and can be configured for selective consumption of Ethernet Global Data exchanges.

### 13.1.2 SRTP Server

CPUE05 supports up to eight simultaneous SRTP Server connections for use by other devices on the Ethernet network, such as the PLC programmer, CIMPLICITY HMI, SRTP channels for Series 90 PLCs, and Host Communications Toolkit applications. No PLC programming is required for server operation.

### 13.1.3 SRTP Channels

SRTP Channels can be used by a Series 90-30 or Series 90-70 PLC to communicate with CPUE05. The CPUE05 cannot initiate SRTP channels.

### 13.1.4 Attachment to the Ethernet LAN

The Ethernet port uses a twisted pair cable of up to 100 meters in length between each node and a hub or repeater. Typical hubs or repeaters support 4 to 12 nodes connected in a star wiring topology.

### 13.1.5 The Station Manager Software

CPUE05 provides built-in Station Manager support. It accommodates online diagnostic and supervisory access through either the Station Manager port or via Ethernet. Station Manager services include:

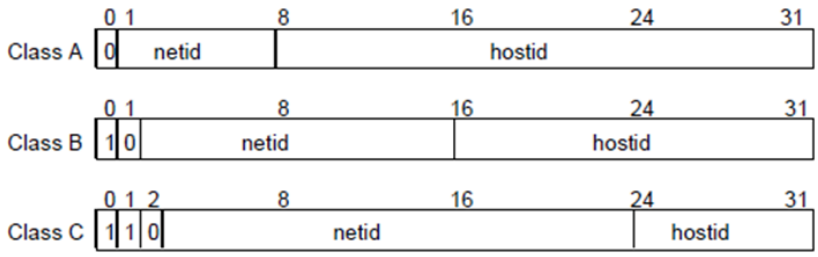
- An interactive set of commands for interrogating and controlling the station.
- Unrestricted access to observe internal statistics, an exception log, and configuration parameters.
- Password security for commands that change station parameters or operation.
- Access to the Station Manager requires a user-provided computer terminal or terminal emulator.

## 13.2 IP Addressing

The CPUE05 must have a unique IP address that identifies it on the Ethernet network.

The IP Address is assigned using the configuration software, as described in chapter 6. The IP address is 32 bits long and has a netid part and a hostid part. The format of the IP address depends on the network class:

**Figure 203**




Each IP address on a network has:

- The same class. Each network is a Class A, Class B or Class C network. A Class A network can support 16,777,214 hosts, Class B: 65,534 hosts, and Class C: 254 hosts.
- The same netid, which is generally assigned by the Internet authorities.
- A different hostid, giving it a unique IP address. The hostid is generally assigned by your local network administrator.

IP addresses are written in “dotted-decimal” format as four decimal integers (0-255) separated by periods. Each integer represents one byte of the IP address. For example, the 32-bit IP address

00001010 00000000 00000000 00000001

is written as

10.0.0.1

The class of an IP address is indicated by the first decimal integer:

| Range of first integer | Class |
|---|---|
| 0 - 127 | A |
| 128 - 191 | B |
| 192 - 223 | C |
| 224-239 | D (Reserved for Multicast Use) |
| 240 - 255 | E (Reserved for Experimental Use) |

RFC 1918 reserves IP addresses in the following ranges for private networks.

10.0.0.0 – 10.255.255.255 (Class A)

172.16.0.0 – 172.31.255.255 (Class B)

192.168.0.0 – 192.168.255.255 (Class C)

x.y.z.1 is reserved for gateways.

x.y.z.255 is reserved for subnet broadcast

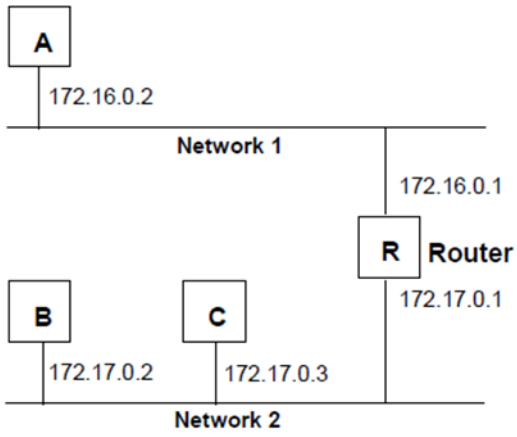
# 13.3 Routers

Routers connect individual physical networks into a system of networks. When a node on one network needs to communicate with a node on another network, a Router transfers the data between the two networks.

### Example: Networks Connected by a Router

The following figure shows Network 1 and Network 2 connected by Router R.

**Figure 204**




Host B can communicate with host C directly because they are on the same network. Their IP addresses have the same netid.

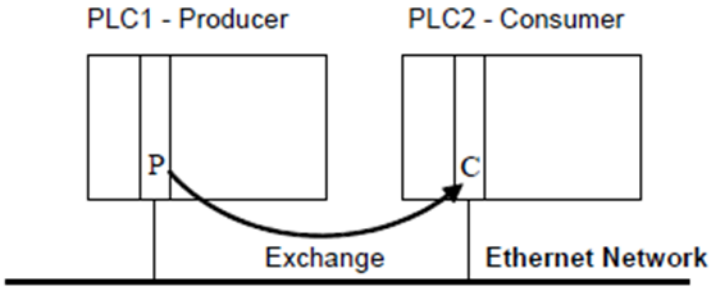
However, to send data to host A, which is on another network (it has a different netid,) host B must send it via the router. The router has two IP addresses (172.16.0.1 and 172.17.0.1). The first is used by hosts on Network 1 and the second is used by hosts on Network 2. In this example, the router's IP address on Network 2 is 172.17.0.1. This address would be configured in host B as its default gateway address.

# 13.4 Ethernet Global Data

Ethernet Global Data is data that is automatically sent from one Ethernet device to one or more others. Once Ethernet Global Data has been configured, the data is sent automatically during system operation. No program interaction is necessary to produce or consume the global data.

The device that sends the Ethernet Global Data is called the producer. Each device that receives Ethernet Global Data is called a consumer. Each unique Ethernet Global Data message is called an exchange.

**Figure 205**




**Ethernet Global Data provides simple, regular communication of data between devices. It should not be used for event notification if possible loss of data would be significant.**

VersaMax CPU IC200CPUE05 can be configured for up to 32 produced Ethernet Global Data exchanges (total of Produced and Consumed) s. Each Ethernet Global Data exchange must be configured individually for each PLC and consists of one or more data ranges. See chapter 6 for configuration information.

## 13.4.1 The Frequency of Sending/Receiving an Exchange

During configuration, the repetition period of each Ethernet Global Data exchange is set up for the producer. The range is 10 milliseconds to 1 hour, which is selectable in increments of 10 mS. It is not necessary to produce and consume data faster than the application requires. This reduces the load on the network and on the devices, providing capacity for other transfers.

## 13.4.2 The Consumer Update Timeout Period

As part of the configuration for each consumed exchange, a “timeout period” can be set up for the exchange. The CPU reports an error if the first or subsequent packet of data has not arrived within the specified time. The range is 0 for no timeout detection, or 10 to 3,600,000 milliseconds. The consumer’s timeout period should be greater than the producer’s repetition period. Emerson recommends that the consumer timeout be set to no lower than twice the production period,

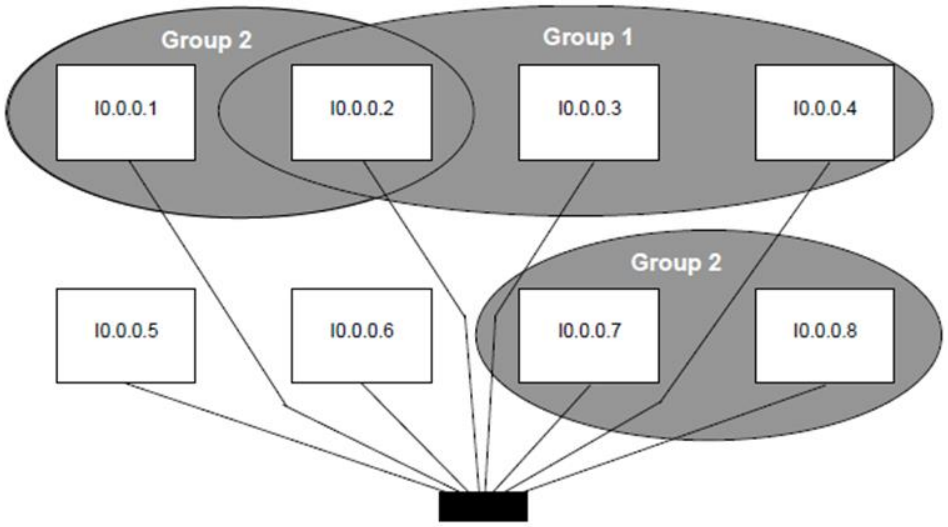
## 13.4.3 Ethernet Global Data Groups

If more than one device on the network should consume a Global Data exchange, those devices can be set up as a group. The network can include up to 32 numbered groups.

Groups allow each sample from the producer to be seen simultaneously by all consumers in the group.

A device can belong to more than one group, as shown in the following illustration.

**Figure 206**




Each device in a group responds to the group's assigned ID number. For CPUE05, the Group IDs are 1 to 32.

Each Group ID corresponds to a Multicast (Class D) IP address reserved by the Internet authorities. The default Multicast IP addresses used by Ethernet Global Data are:

| Group ID | IP Address |
|---|---|
| 1 | 224.0.7.1 |
| 2 | 224.0.7.2 |
| . | . |
| . | . |
| . | . |
| 32 | 224.0.7.32 |

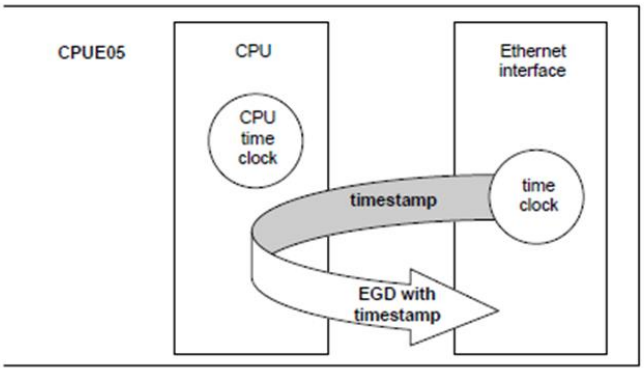
Group Multicast IP Addresses used by Ethernet Global Data should not be changed unless the defaults would cause a network conflict. If necessary, they can be changed within the reserved range of multicast IP addresses (224.0.0.0 through 239.255.255.255). The change must be made using an Advanced User Parameter File.

## 13.4.4 Timestamping of Ethernet Global Data Exchanges

The PLC CPU timestamps each Ethernet Global Data Message it produces. The timestamp indicates when the data was transferred from the producing PLC's CPU to its Ethernet interface for transmission over the network.

The PLC CPU obtains the timestamp data from the time clock in the Ethernet interface. The CPU only uses this timestamp for Ethernet Global Data exchanges. The timestamp from the Ethernet interface does not affect the time of the CPU's internal time clock.

**Figure 207**

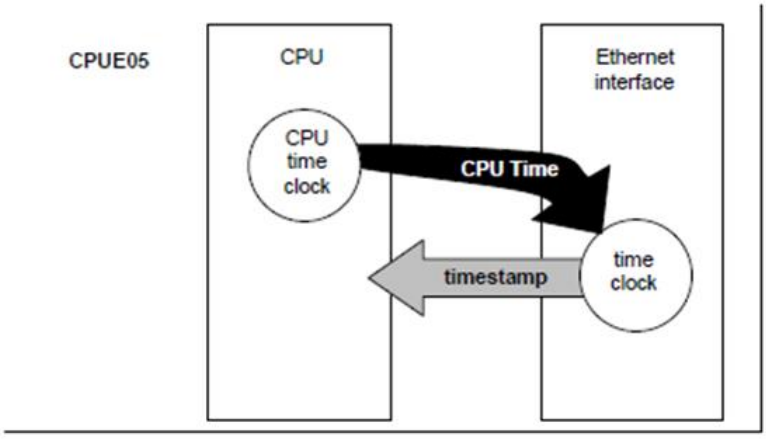



### Synchronizing the Timestamp

The timestamp clock in the Ethernet interface is synchronized to either the clock in the CPU or an external Network Time Protocol (NTP) server. NTP is supported in IC200CPUE05-HK and previous versions only.

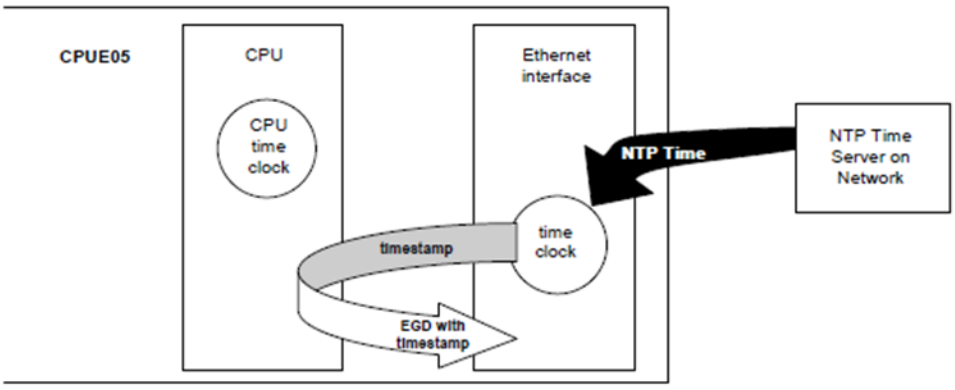
- **The CPU Time Clock:** If no NTP servers are configured, the Ethernet interface's built-in time clock is synchronized once, at power-up or restart, to the clock in the CPU. Because the clocks in the other devices on the network are not synchronized with the CPUE05, their timestamps cannot be compared accurately.

**Figure 208**



- **NTP Server Time Clock:** If time servers are configured and present on the network (see chapter 6 for configuration details), the Ethernet interface's built-in clock is periodically synchronized to the clock from one to three NTP servers on the network. The Ethernet interface periodically requests time from the servers and uses the time from the most accurate server (based on NTP stratum number).

**Figure 209**




All Ethernet interfaces that have been configured to use Network Time Protocol will have updated, synchronized timestamps because they are all controlled by the NTP server clock. Therefore, accurate timing comparisons between exchanged data can be made. For example, if several PLCs sent alarm data, it might be helpful to know the order in which the alarms occurred.

Multiple NTP servers can be used to improve the availability of time servers.

When the time is obtained from an NTP server, dates from January 1, 1970 are supported by the Ethernet Interface.

## 13.4.5 Configuring NTP for the CPUE05 Ethernet Interface

This feature is supported in IC200CPUE05-HK and previous versions only.

To implement Network Time Protocol in the Ethernet interface in CPUE05, the IP address of one to three NTP Time Servers are specified in the PLC Ethernet configuration. Refer to chapter 6, the section, Configuring the Ethernet Interface for details. CPUE05 does not support multicast NTP operation; multiple NTP servers may be specified individually.

The Ethernet interface in CPUE05 always operates in client mode. It will synchronize to an NTP time server, but it will not synchronize other devices on the network.

Time synchronization takes multiple message exchanges to reach maximum precision. Based on the default configuration of poll times, NTP synchronization should occur approximately 2 minutes after a time server has been established.

## 13.4.6 The Content of an Ethernet Global Data Exchange

ranges transmitted as a sequence of 1 to 1400 bytes of data. The content of the data is defined for both the producer and consumers of the data. In this example, a producer sends an 11-byte exchange consisting of the current contents of %R00100 through % R00104 followed by the current contents of %I00257 through %I00264:

| Address | Length | Type | Description |
|---|---|---|---|
| %R00100 | 5 | WORD | Conveyor1 in PLC1 |
| %I00257 | 1 | BYTE | Conveyor1 limit switch in PLC1 |

The same exchange can be configured for each consumer to suit the needs of the application. The size of the exchange must be consistent on all nodes.

## 13.4.7 Data Types for Ethernet Global Data

The table below lists memory types that can be configured for produced and/or consumed Ethernet Global Data.

| Type | Description | Producer or Consumer |
|---|---|---|
| %R | Register memory in word mode | P/C |
| %AI | Analog input memory in word mode | P/C |
| %AQ | Analog output memory in word mode | P/C |
| %I | Discrete input memory in byte mode | P/C |
| %Q | Discrete output memory in byte mode | P/C |
| %T | Discrete temporary memory in byte mode | P/C |
| %M | Discrete momentary memory in byte mode | P/C |
| %SA | Discrete system memory group A in byte mode | P/C |
| %SB | Discrete system memory group B in byte mode | P/C |
| %SC | Discrete system memory group C in byte mode | P/C |
| %G | Discrete global data table in byte mode | P/C |

### The Data Ranges in a Global Data Exchange

The variable ranges in an exchange are defined in the Ethernet Global Data configuration in hardware configuration. There can be:

- Up to 1200 data ranges for all EGD exchanges for one CPUE05.
- Up to 100 data ranges per exchange.
- A length of 1 byte to 1400 bytes per exchange. The total size of an exchange is the sum of the data lengths of all of the data ranges configured for that exchange.

Different exchanges may share some or all of the same data ranges even if the exchanges are produced at different rates. A consumer does not have to consume all of the data from a produced exchange. A consumed exchange may be configured to ignore specified data ranges. (See "Selective Consumption" in chapter 6.)

## 13.4.8 Effect of PLC Modes and Actions on Ethernet Global Data

The usual PLC mode for Ethernet Global Data operation is Run with I/O enabled. In that mode, Ethernet Global Data remains configured and exchanges are both produced and consumed. If the PLC mode is set to Stop with I/O disabled, the Producer ID remains configured, but production and consumption stops. The samples of the consumed exchanges received while the PLC is stopped continue to be processed by the Ethernet interface. The latest received data from the network will be available to the application when the PLC returns to an I/O enabled state.

The table below summarizes what happens to the configuration and operation of Ethernet Global Data in different PLC modes.

| PLC Mode or Action | Exchanges continue to be… | |
|---|---|---|
| | Produced | Consumed |
| RUN-Outputs Enabled | yes | yes |
| STOP-I/O Enabled | yes | yes |
| STOP-I/O Disabled | no | no * |
| * The latest data from the network is available to the application when the PLC transitions from Stop to Run mode. | | |

## 13.4.9 EGD Synchronization

Ethernet Global Data attempts to provide the most up-to-date process data, consistent with the configured schedule. The Ethernet interface maintains a timer for each produced exchange. When the timer for the exchange expires, the Ethernet interface requests that the data for the exchange be transferred from reference memory during the output scan portion of the next CPU sweep. Once the data has been transferred by the CPU sweep, the Ethernet interface immediately formulates a sample and transfers the sample on the network. As soon as a sample for a consumed exchange is received, it is transferred to the CPU during the next input scan portion of the CPU sweep.

The result of this scheduling method for Ethernet Global Data is a variability of up to one producer CPU sweep time in the interval between samples produced on the network. This variability in the time between samples is present to assure that the most up-to-date data is being transferred.

In general, it is not useful or necessary to configure the production period to be less than the CPU sweep time. If the producer period for an exchange is set lower than the CPU sweep time, the Ethernet interface will send a “stale” sample (a sample containing the same data as previously sent) at the configured interval. When the fresh CPU data becomes available at the end of the sweep, the Ethernet interface will immediately send another sample with the fresh data. The timer of the produced exchange is not reset when this sample is sent. This can result in more samples in the network than would be expected from the configured period.

## Timing Examples

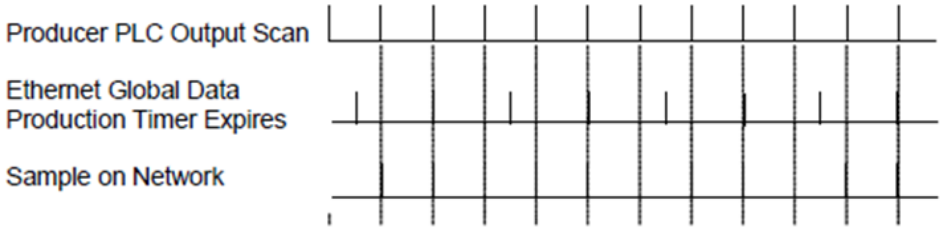
The following illustrations show the relationship between the PLC output scan time, the produced exchange timer, and data samples on the network.

### Example 1

Only one sample is produced on the network per producer period expiration. The variability between samples can be up to producer CPU sweep time.

**Producer Period = 1.5 Times CPU Sweep**

**Figure 210**

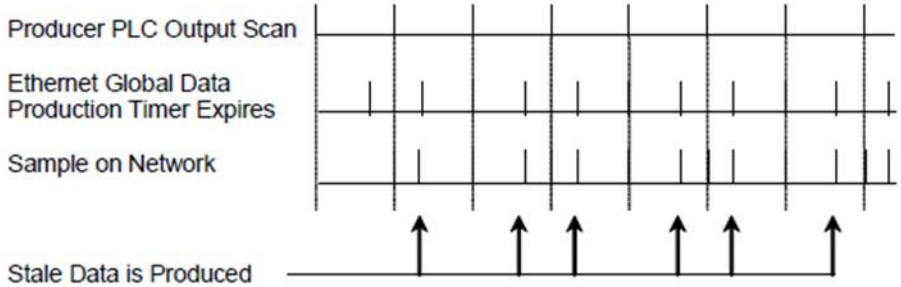



### Example 2

More than one sample can be produced per producer period expiration and stale samples are produced to the network.

**Producer Period = 2/3 Time of CPU Sweep**

**Figure 211**



# 13.5 Diagnostic Tools

There are several tools to assist you in diagnosing problems that may occur with Ethernet operations and Ethernet Global Data.

- Check the **Ethernet LEDs**, as detailed in the following sections, to troubleshoot a problem on power-up of the Ethernet Interface. The LEDs provide an immediate visual summary of the operating state of the Interface.
- Use the **PLC Fault Table**, also explained in this chapter. The PLC Fault Table records exceptions logged by the PLC, the Ethernet interface, and other modules. The PLC Fault Table is accessed through the PLC programming software.
- The application program can use special status data to monitor Ethernet operations.
  - The Ethernet interface status address, selected during PLC configuration, contains information about the operating status of the Ethernet interface.
  - The Exchange Status words, selected during Ethernet Global Data configuration, contain information about the status of exchange operations.
- • Use the **Station Manager** function to troubleshoot a problem with the Ethernet Interface, with the network, with PLC backplane communication, or with your application. The LOG, TALLY, and STAT Station Manager commands are especially useful. Refer to the VersaMax PLC Ethernet Station Manager Manual(GFK–1876) for information on how to access and use the Station Manager.

## 13.5.1 What to do if you Cannot Solve the Problem

If you still cannot solve your problem, call Emerson – NA, 1- 800-433–2682.

Please have the following information available when you call.

- The Name and Catalog Number marked on the product.
- Description of symptoms of problem. Depending on the problem, you may also be asked for the following information:
  - The ladder logic application program and the PLC sweep length at the time the problem occurred.
  - A listing of the configuration parameters for the Ethernet Interface that failed.
  - A description of the network configuration. This should include the number of PLCs and host computers accessing the network, the type of network cable used (e.g. twisted pair, fiber optic, etc.), length of network cable, and the number and manufacturer of transceivers, hubs, and network switches used.

## 13.5.2 Checking the Ethernet LEDs

After configuring the Interface, follow the steps below to verify that the Ethernet Interface is operating correctly.

1. Turn power OFF to the PLC for 3–5 seconds, then turn the power back ON. This starts a series of diagnostic tests. During powerup diagnostics, after a brief delay the STAT LED on the Ethernet side of the CPU module blinks. Both the LAN and PORT1 LEDs are off. If a fatal diagnostic failure occurs, the failure is indicated by a two-digit pattern in amber on the STAT LED.
2. For IC200CPUE05-IL and earlier versions, the following applies:

   After successful power-up, all three LEDs on the Ethernet side turn on briefly. Then the STAT and LAN LEDs should be green. The LAN LED blinks when there is traffic.

   For version IC200CPUE05-IL and later the following applies: After successful power-up, STAT LED turns ON on the Ethernet side. Then the STAT LED should be green for a brief moment. The LAN LED turns ON when it is connected to network and blinks when there is traffic. LAN LED remains OFF if module is not connected to the network.
3. If the STAT LED is amber, check the PLC Fault Table. With the Station Manager feature, you can also use the LOG command as explained in GFK-1876, The VersaMax PLC Ethernet Station Manager Manual.

   If a problem occurs during power-up, the Ethernet interface may not begin operating. Check the Ethernet LEDs, as explained on the following pages.

| Ethernet LEDs | Indications | | Actions |
|---|---|---|---|
| LAN ◯ Off<br>STAT ◯ Off<br>PORT 1 ◯ Off | Off | | • Make sure the PLC has power<br>• Look in the PLC Fault Table for problems<br>• Recheck configuration<br>• Check module installation<br>• If the problem persists, replace PLC CPU |
| LAN ◯ Off<br>STAT ◔ Blinking amber<br>PORT 1 ◯ Off | • Hardware failure mode. STAT: Blinks 2-digit error code: | | • Note error code<br>• Power cycle or restart Ethernet interface<br>• If problem persists, replace the PLC hardware. |
| | 1 – 2 | unexpected interrupt | |
| | 1 – 3 | timer failure | |
| | 1 – 4 | DMA failure | |
| | 2 – 1 | RAM failure | |
| | 2 – 2 | stack error | |
| | 2 – 3 | shared memory interface error | |
| | 2 – 4 | firmware CRC error | |
| | 3 - 1 | unidentified instruction, or divide by 0 | |

| Ethernet LEDs | Indications | | Actions |
|---|---|---|---|
| | 3 – 2 | unexpected SWI interrupt | |
| | 3 – 3 | prefetch abort error | |
| | 3 - 4 | data abort error | |
| | 3 – 5 | unexpected IRQ request | |
| | 3 – 6 | unexpected FIQ interrupt | |
| | 3 - 7 | reserved exception error | |
| | 4 - 1 | fatal operating system startup or EEPROM error | |
| LAN Off<br>STAT Slow blink green<br>PORT 1 Off | Waiting for Ethernet configuration data from CPU.<br>PORT 1: PLC CPU is controlling Port 1. | | • Use the PLC programmer to update the configuration, then store the configuration to the PLC.<br>• Power cycle the PLC.<br>• Clear faults and press the Restart pushbutton for less than 5 seconds to restart the Ethernet interface. |
| LAN Green / flickering<br>STAT Slow blink green<br>PORT 1 Off | Waiting for IP Address LAN: Ethernet interface is online. Flickers during activity.<br>STAT: IP Address has not been configured.<br>PORT 1: PLC CPU is controlling Port 1. | | IP address has not been configured, or has been configured as 0.0.0.0<br>• Use the PLC programmer to configure a non-zero IP address. |
| LAN Green / flickering<br>STAT Slow blink green<br>PORT 1 Amber | Waiting for IP Address LAN: Ethernet interface is online. Flickers during activity.<br>STAT: IP Address has not been configured.<br>PORT 1: Available for Station Manager use | | |

| Ethernet LEDs | Indications | Actions |
|---|---|---|
| LAN Off<br>STAT Slow blink green<br>PORT Amber | Waiting for IP Address LAN: Ethernet interface is offline. Attempting to recover if possible.<br>STAT: IP Address has not been configured.<br>PORT 1: PLC CPU is controlling Port 1. | |
| LAN Off<br>STAT Slow blink green<br>PORT Amber | Waiting for IP Address LAN: Ethernet interface is offline. Attempting to recover if possible.<br>STAT: IP Address has not been configured.<br>PORT 1: Available for Station Manager use | |
| LAN Green / flickering<br>STAT Green<br>PORT 1 Off | Operational<br>LAN: Ethernet interface is online. Flickers during activity.<br>STAT: No “exception” detected<br>PORT 1: PLC CPU is controlling Port 1. | If LAN is off, the problem may be:<br>• Network cable not connected either at the PLC or at the hub.<br>• Hub disconnected/failed.<br>• Network cable not properly terminated.<br>If STAT is amber, an “exception” condition has occurred. |
| LAN Green / flickering<br>STAT Green<br>PORT 1 Amber | Operational<br>LAN: Ethernet interface is online. Flickers during activity.<br>STAT: No “exception” detected<br>PORT 1: Forced to Station Manager use | |
| LAN Off<br>STAT Green<br>PORT 1 Off | Operational<br>LAN: Ethernet interface is offline. Attempting to recover if possible.<br>STAT: No “exception” detected<br>PORT 1: PLC CPU is controlling Port 1. | |

| Ethernet LEDs | Indications | Actions |
|---|---|---|
| LAN ○ Off<br>STAT ● Green<br>PORT 1 ◐ Amber | Operational<br>LAN: Ethernet interface is offline. Attempting to recover if possible.<br>STAT: No “exception” detected<br>PORT 1: Forced to Station Manager use | |
| LAN ◑ Slow blink green,<br>STAT ◑ LEDS blink in<br>PORT 1 ◑ unison | Software Load<br>Loading new firmware (via CPU serial port) | No action necessary; the Ethernet interface restarts automatically after loading is complete |

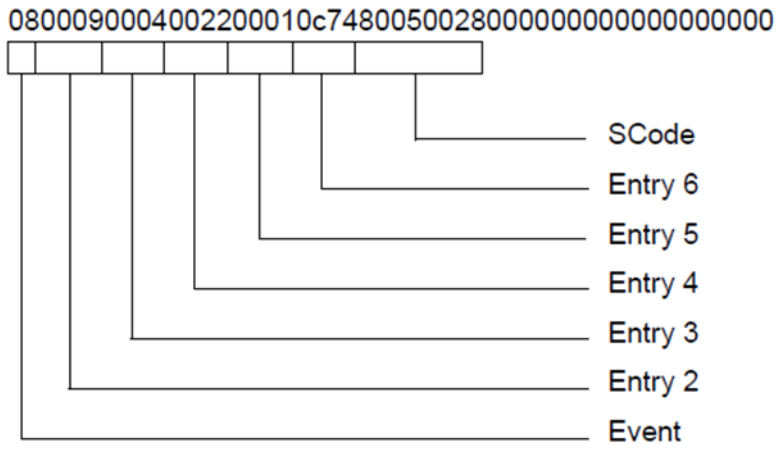
## 13.5.3 Using the PLC Fault Table

Most error conditions involving the Ethernet interface generate faults in the PLC Fault table. The table on the next page lists Ethernet interface faults and corrective actions.

To display fault text for the Ethernet interface, access the PLC Fault Table from the programmer. For the Ethernet interface the leftmost 14 digits of extra fault data show the corresponding log Events (2 digits) and Entries 2,3, 4, 5, and 6 (in that order, 4 digits each), and other optional data.

The following example reports an Event 8, Entry 2=9, Entry 3=4, Entry 4 =22H, Entry 5 = 1, Entry 6=c74H, and SCode = 80050028H.

**Figure 212**




This information can be used to refer directly to detailed fault descriptions included in the Log event table under the LOG command in the VersaMax PLC Station Manager Manual (GFK-1876).

Please note some internal system errors display error messages as ASCII text in the fault extra data.

## PLC Fault Table Descriptions

| PLC Fault | User Action |
|---|---|
| Backplane communications with PLC fault; lost request | Check that PLC CPU is running normally (usually in Run mode) *<br>Check to make sure you are not sending COMMREQs faster than the Ethernet interface can process them. * |
| Bad local application request; discarded request | Check for valid COMMREQ command code. * |
| Bad remote application request; discarded request | Try to validate the operation of the remote node. * |
| Can't locate remote node; discarded request | Error reported when message received where IP address cannot be resolved. Error may indicate that remote host is not operational on the network. Check that remote host is operational on network and its addresses are correct. |
| Comm_req – Bad task ID programmed | Message from PLC for unknown Ethernet interface task. Check COMMREQ function block. |
| Comm_req – Wait mode not allowed | Check COMMREQ to make sure sent in no-wait mode. |
| LAN data memory exhausted – check parms; resuming | The Ethernet interface does not have free memory to process communications. * |
| LAN I/F capacity exceeded; discarded request | Verify that connection limits are not being exceeded. |
| LAN transceiver fault; Off network until fixed | Ethernet interface is not properly connected to the network. Check the connection to the network hub or switch. |
| LAN system-software fault; aborted connection resuming | Internal system error. * |
| LAN system-software fault; restarted LAN IF | |
| LAN system-software fault; resuming | |
| Module software corrupted; requesting reload | Catastrophic internal system error. Contact Emerson Technical Support at https://www.emerson.com/Industrial-Automation-Controls/support. |
| Module state doesn't permit Comm_Req; discarded | COMMREQ received when Ethernet interface cannot process COMMREQ. Make sure Ethernet interface is configured and online. |

| PLC Fault | User Action |
|---|---|
| Unsupported feature in configuration | Attempt has been made to configure a feature not supported by the Ethernet interface. Check CPU revision; order upgrade kit for CPU and/or Ethernet interface. |

* If the problem persists, contact Emerson– NA.

## 13.5.4 Checking the Status of the Ethernet Interface

The application program can monitor the status of the Ethernet interface using the status bits described below. The beginning address of the data is the Status Address entered when configuring the CPU. See "Configuring the Ethernet Interface" in chapter 6 for details.

The Ethernet interface updates these status bits every PLC I/O scan. The Ethernet status bits normally occupy a single block of memory. Most of these bits are reserved. Five are of interest for checking the status of the Ethernet interface:

| Status Bits | Brief Description |
|---|---|
| 1–2 | Reserved, always 0 |
| 3 | Full-duplex |
| | |
| 4-12 | Reserved, always 0 |
| 13 | LAN OK |
| 14 | Resource problem |
| 15 | Reserved, always 0 |
| 16 | LAN Interface OK |
| 17-80 | Reserved |

| | |
|---|---|
| **Bit 3: Full Duplex** | If this bit 3 is 1, CPUE05 is operating in full-duplex Ethernet mode. Full- duplex or half-duplex operation is automatically negotiated between the CPUE05 and its immediately-connected network device, usually a network hub. If this bit is 0, CPUE05 is operating in half-duplex Ethernet mode. This bit is only valid if bit 13 (LAN OK) is 1. |
| **Bit 13: LAN OK** | This bit is 1 while the Ethernet interface is able to communicate on the network. If the network is not accessible due to local or network problems, this bit is 0. When communication resumes, it is automatically set to 1. |
| **Bit 14: Resource Problem** | This bit is 1 whenever the Ethernet interface has a resource problem (i.e., lack of data memory). The bit is reset to 0 on a subsequent PLC sweep. The Ethernet interface may or may not be able to continue functioning, depending on the severity of the problem. Use the PLC Fault Table to |

| | |
|---|---|
| **Bit 16: LAN Interface OK** | identify the problem. The Station Manager STAT B and LOG commands can also provide more information. |

## 13.5.5 Checking the Status of an Ethernet Global Data Exchange

To check the status of any Ethernet Global Data exchange, monitor the value in the Exchange Status word (selected during Ethernet Global Data configuration). The PLC automatically writes exchange status information in this location when:

- a producer/consumer period expires ( the value is set for the entire period).
- an Ethernet Global Data configuration is stored to the PLC.
- the PLC powers up and it has an Ethernet Global Data configuration.
- the Ethernet interface configured for Ethernet Global Data is restarted. If the application program uses the Exchange Status word to check exchange status, it must clear this word to 0 once a non-zero value is written to it. That allows the application program to detect a new exchange status in subsequent sweeps.

The Exchange Status word uses the error codes below to report exchange status. Refer to the section, Troubleshooting Common Ethernet Difficulties later in this chapter.

| Value (Decimal) | Error | Description |
|---|---|---|
| 0 | Exchange status has not been updated | Produced: Initial value until the first producer period refresh occurs.<br>Consumed: The data has not been refreshed and timeout has not expired. |
| 1 | No error | Produced: The produced exchange is producing data. Consumed: The data has been refreshed on schedule. |
| 3 | NTP error | Consumed only: The CPU is configured for network time synchronization, but is not synchronized. (This feature is supported IC200CPUE05-HK and previous versions only.) |
| 4 | Specification error | Produced and Consumed: Error configuring the exchange. For CPUE05, this error does NOT indicate a consumed exchange size mis comparison. |
| 6 | Refresh timeout without data. | Consumed only: The timeout period has expired but data has not been refreshed from the network. |
| 7 | Data after refresh timeout | Consumed only: The data has been refreshed since the previous consumption, but was not refreshed within the timeout period. |
| 10 | IP connection not available | Produced and Consumed: The IP network connection is not available. |

| Value (Decimal) | Error | Description |
|---|---|---|
| 12 | Lack of resource error | Produced and Consumed: Local resources are not available to establish the exchange. Look in the PLC Fault Table for details. |
| 14 | Length error | Consumed only: The packet received did not match the length expected. |
| 18 | Loss of Ethernet interface error | Produced and Consumed: The Ethernet interface is not communicating with the CPU. A loss of module or reset of module PLC Fault Table entry may also be present. If the failure is transient in nature, the status of the exchange may change at a later time. That indicates subsequent transfers on the exchange were successful. |
| 22 | EGD not supported | This error cannot occur with CPUE05. |
| 26 | No response | Produced and Consumed: Ethernet interface failed to establish exchange. |
| 28 | Other error | Produced and Consumed: Error other than 12, 14, 18, or 26 when establishing an exchange. Look in the PLC Fault Table for information. |
| 30 | Exchange deleted | Produced And Consumed: Exchange has been deleted and will no longer be scanned. |

### 13.5.6 Using the Ethernet Station Manager Function

CPUE05 provides local Station Manager operation via Port 1. This port can be configured for either CPU serial communications (SNP, RTU, Serial I/O) or local Station Manager use. While Port 1 is configured as a local Station Manager, it cannot be used for CPU serial communications or firmware loading. However, if the port is configured as a CPU port instead (the default setting), it can temporarily be forced to local Station Manager operation using the Restart pushbutton (or using the "chport1" Station Manager command).

The CPUE05 also supports remote Station Manager operation over the Ethernet network via UDP protocol. With UDP protocol, the remote station is addressed via an IP address. Unlike some Series 90 Ethernet products, CPUE05 cannot send or receive remote Station Manager messages that have been sent to a specified MAC address.

For a detailed description of Station Manager functions, please refer to GFK-1876, the VersaMax PLC Ethernet Station Manager User's Manual.

## 13.6 Troubleshooting Common Ethernet Difficulties

Some common Ethernet errors are described below. Ethernet errors are generally indicated in the PLC Fault Table and the Ethernet exception log. As previously explained in Using the PLC Fault Table, PLC Faults generated by the Ethernet interface contain Ethernet exception events within the extra fault data. See the VersaMax Station Manager Manual, GFK-1876 for detailed descriptions of Ethernet exception events.

## 13.6.1 PLC Timeout Errors

When the SRTP traffic to the CPUE05 exceeds the PLC's ability to process the requests, PLC Timeout errors may occur. PLC Timeout errors will take down an SRTP Server connection; in this case, the remote SRTP client must reestablish a new SRTP connection to the CPUE05.

This error is indicated in the PLC Fault Table as:

Backplane communication with PLC fault; lost request with exception Event = 8, Entry 2 = 8

Backplane communication with PLC fault; lost request (no exception Event)

These errors may also be accompanied by either of the following:

Backplane communication with PLC fault; lost request with exception

Event = 8,

Entry 2 = 6

LAN system-software fault; resuming with exception Event = 8,

Entry 2 = 16

The PLC Timeout condition occurs when the CPUE05 cannot process requests within a specified timeout period. The remedy is to reduce the requests, or increase the processing capacity in the PLC.

| Cause | Corrective Action |
|---|---|
| Heavy SRTP traffic. | Reduce the size, number, or frequency of SRTP requests at the remote SRTP client. |
| Long PLC sweep time. | Modify the PLC application to reduce the PLC sweep time. |
| PLC Communication Window set to LIMITED mode. | Change to RUN-TO-COMPLETION mode. |

If none of the above corrective actions is feasible, the timeout interval may be lengthened. The timeout interval is specified by the "crsp_tmot" Advanced User Parameter. The default timeout value is 15 seconds. See Configuring Advanced User Parameters in chapter 6 to change Advanced User Parameter values.

Note that changing this timeout value does not reduce the actual time for the PLC to process the requests.

## 13.6.2 Unexpected Ethernet Restart or Runtime Errors

Sustained heavy EGD and/or SRTP operation can exceed the data transfer and processing capacity of the CPUE05. This can result in missed EGD exchanges, unexpected automatic restarts of the Ethernet interface within the CPUE05, or runtime fatal errors at the Ethernet interface.

Restart errors are indicated in the PLC Fault Table as one or more of the following:

"Loss of daughterboard" (no exception Event) "

Reset of daughterboard" (no exception Event) "

LAN system-software fault; restarted LAN I/F" with exception Event = 3, Entry 2 = 1, Entry 3 = 5f0fH

After any of the above errors, the Ethernet interface restarts itself automatically without manual intervention.

The above Ethernet restarts may be accompanied by one or more of the following in the PLC Fault Table:

"Backplane communications with PC fault; lost request" (no exception event) "LAN system-software fault; resuming"

with exception Event = 28, Entry 2 = 1, SCode = 95255037H

Runtime errors suspend normal operation and a blink fatal error code in amber at the STAT LED. To recover, manually restart the Ethernet interface. Runtime error codes "31" and "33" have been observed under heavy load. See Checking the Ethernet LEDs section earlier in this chapter for descriptions of runtime diagnostic fatal error codes.

All Ethernet Global Data (EGD) exchanges default to status code 18 (0012H) during a loss or reset of the Ethernet interface. EGD operation will resume after the restart is complete.

These restart and runtime errors occur when the CPUE05 cannot process the attempted volume of EGD and/or SRTP requests. As these errors have been observed only when the CPUE05 is connected to a repeater-type network hub, the primary remedy is to replace the repeater-type hub with a switching-type network hub. A secondary remedy is to reduce the number, size, or frequency of the EGD exchanges and/or transfers over SRTP connections.

## 13.6.3 EGD Configuration Mismatch Errors

When using Ethernet Global Data, the produced exchange (defined at the producer) must agree with the consumed exchange (defined at the consumer). The consumer generates an error when the size of an exchange received from the network differs from the configured size for that consumed exchange.

This error is indicated in the PLC Fault Table as:

LAN system-software fault; resuming

with exception Event = 28, Entry 2 = 1d

As this error is generated each time the mismatched exchange is received, the Ethernet exception log can quickly fill up with mismatch error events.

| Cause | Corrective Action |
|---|---|
| Producer and Consumer exchange definitions are of different size. | Review the conflicting exchange definitions at the producer and at the consumer. Change the incorrect exchange definition so that produced and consumed definitions are the same size. |

If the consumer wishes to ignore certain portions of a consumed exchange, be sure that the length of the ignored portions is correct. The ignored portion is specified as a byte count.

## 13.6.4 Receive Resource Exhaustion Errors

Heavy network traffic can exhaust available memory in the Ethernet interface used for network communications. This most often occurs under heavy Ethernet Global Data (EGD) traffic on a busy network. Since the traffic on the network is unpredictable, this error condition may always occur.

This error is indicated in the PLC Fault Table as:

"LAN system-software fault; resuming"

with exception Event = 28, Entry 2 = 1

| Cause | Corrective Action |
|---|---|
| Heavy EGD traffic exhausts network data buffers. | Modify the application to reduce the number, size, or frequency of produced and consumed EGD exchanges. |
| Bursts of heavy network traffic are received at the CPUE05. | Analyze the broadcast and multicast network traffic received at the CPUE05. Reduce such traffic if possible. |

## 13.6.5 Station Manager Lockout under Heavy Load

Sustained heavy EGD and/or SRTP Server load can utilize all processing resources within the Ethernet interface, effectively locking out the Station Manager function. The Station Manager appears inoperative under either local or remote operation. The Ethernet interface always gives higher priority to data communication functions than to the Station Manager. When the processing load is reduced, the Station Manager becomes operative once again.

This condition is not reported to the PLC Fault Table or Ethernet exception log.

## 13.6.6 PING Restrictions

To conserve network data buffer resources, the CPUE05 process only one ICMP control message at a time. An ICMP Echo (ping) request that arrives while the CPUE05 is processing another ICMP control message is discarded. When multiple remote hosts attempt to ping the CPUE05 at the same time, some individual ping requests may be ignored depending upon the timing of the ping requests on the network.

The CPUE05 may initiate ping requests to another host on the network via the "ping" Station Manager command. The ping request sequence is restricted to one remote host at a time.

Discarded ping requests are not reported to the PLC Fault Table or Ethernet exception log.

## 13.6.7 SRTP Connection Timeout

When a remote SRTP client is abruptly disconnected from a CPUE05 (for example, by disconnecting the Ethernet cable), the underlying TCP connection attempts to re-establish communication. The SRTP connection in the CPUE05 remains open for approximately 5 minutes while TCP attempt to reconnect; during this interval, the SRTP connection is unavailable. If all the SRTP connections in the CPUE05 are in use or otherwise unavailable, a new SRTP client connection must wait until the TCP reconnect time expires on an existing connection.

The SRTP connection timeout is normal expected behavior and is consistent with other Emerson PLC products.

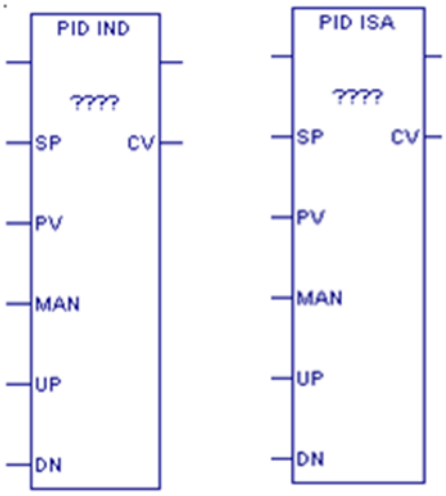
# Chapter 14: PID Built-in Function Block

This chapter describes the PID (Proportional plus Integral plus Derivative) built-in function block, which is used for closed-loop process control.

- Operands of the PID Function
- Reference Array for the PID Function
- PID Algorithm Selection and Gain Calculations
- Determining the Process Characteristics
- Setting and Tuning Loop Gains
- Example

## 14.1 Operands of the PID Function

**Figure 213**




The PID function compares feedback from a process variable (PV) with a desired process set point (SP) and updates a control variable (CV) based on the error

The PID function uses PID loop gains and other parameters stored in a 40-word reference array of 16-bit integer words to solve the PID algorithm at the desired time interval.

### 14.1.1 Parameters of the PID Function Block

| Parameter | Description | Allowed Types | Allowed Operands | Optional |
|---|---|---|---|---|
| Address (????) | Location of the PID reference array, which contains user-configurable and internal parameters, described on Reference Array Parameters later in this chapter. Uses 40 words that cannot be shared. | WORD | R, L, P, W and symbolic | No |
| SP | The control loop or process Set Point. Set using Process Variable counts, the PID function adjusts the output Control Variable so that the Process Variable matches the Set Point (zero error). | INT, BOOL array of length 16 or more | All except S, SA, SB, and SC | No |
| PV | Process Variable input from the process being controlled. Often a %AI input. | INT, BOOL array of length 16 or more | All except S, SA, SB, and SC, and constant | No |
| MAN | When energized to 1 (through a contact), the PID function block is in manual mode. If this input is 0, the PID function block is in automatic mode. | NA | Flow | No |
| UP | If energized along with MAN, increases the Control Variable by 1 CV count per solution of the PID function. | NA | Flow | No |
| DN | If energized along with MAN, decreases the Control Variable by 1 CV count per solution of the PID function. | NA | Flow | No |
| CV | The Control Variable output to the process. Often a %AQ output. | INT, BOOL array of length 16 or more | All except %S and constant | No |

## 14.2 Reference Array for the PID Function

This parameter block for the PID function occupies 40 words of memory, located at the starting Address specified in the PID function block operands. Some of the words are configurable. Other words are used by the CPU for internal PID storage and are normally not changed.

Every PID function call must use a different 40-word memory area, even if all the configurable parameters are the same.

The configurable words of the reference array must be specified before executing the PID function. Zeros can be used for most default values. Once suitable PID values have been chosen, they can be defined as constants in BLKMOV functions so the program can set and change them as needed.

The PID function does not pass power flow if there is an error in the configurable parameters. The function can be monitored using a temporary coil while modifying data.

***Note:*** *If you set the Override bit (bit 0) of the Control Word (word 15 of the reference array) to 1, the next four bits of the control word must be set to control the PID function block input contacts, and the Internal SP (word 16) and Internal PV (word 18) must be set because you have taken control of the PID function block away from the ladder logic.*

## 14.2.1 Scaling Input and Outputs

All parameters of the PID function are 16 bit integer words for compatibility with 16 bit analog process variables. Some parameters must be defined in either PV counts or units or in CV counts or units.

The SP input must be scaled over the same range as the PV, because the PID function calculates error by subtracting these two inputs.

The process PV and control CV counts do not have to use the same scaling. Either may be -32000 or 0 to 32000 to match analog scaling, or from 0 to 10000 to display variables as 0.00% to 100.00%. If the process PV and control CV do not use the same scaling, scale factors are included in the PID gains.

## 14.2.2 Reference Array Parameters

The reference array parameters are described on the following pages. The Machine Edition software allows you to modify the configurable parameters for a PID instruction in real time in online programmer mode. To customize PID parameters, right click the PID function and select Tuning.

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 1<br>(Address + 0) | **Loop Number**<br>Optional number of the PID block. It provides a common identification in the CPU with the loop number defined by an operator interface device. | Integer | 0 to 255 (for user display only) |
| 2 (Address + 1) | **Algorithm**<br>1 = ISA algorithm<br>2 = independent algorithm | - | Set by the CPU |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 3 (Address + 2) | **Sample Period**<br>The shortest time, in 10 ms. increments, between solutions of the PID algorithm. For example, use a 10 for a 100ms. sample period. | 10 ms. | 0 (every sweep) to 65535 (10.9 Min) At least 10ms. |
| 4,5<br>(Address + 3, Address + 4) | **Dead Band +**<br>**Dead Band –**<br>Integral values defining the upper (+) and lower (-) Dead Band limits. If no Dead Band is required, these values must be 0. If the PID Error (SP - PV) or (PV - SP) is above the (-) value and below the (+) value, the PID calculations are solved with an Error of 0. If non- zero, the (+) value must greater than 0 and the (-) value less than 0 or the PID block will not function.<br>Leave these at 0 until the PID loop gains are set up or tuned. A Dead Band might be added to avoid small CV output changes due to variations in error. | PV Counts | Dead Band +: 0 to 32767 (never negative)<br>Dead Band -: -32768 to 0 (never positive) |
| 6<br>(Address + 5) | **PID_IND: Proportional Gain (Kp)**<br>**PID_ISA: Controller gain (Kc = Kp)**<br>PID_IND: Change in the Control Variable in CV Counts for a 100 PV Count change in the Error term. Entered as an integer representing a fixed-point decimal ratio with two decimal places. Displayed as a ratio of percentages with two decimal places.<br>For example, a Kp entered as 450 is displayed as 4.50 and results in a Kp * Error / 100 or 450 * Error / 100 contribution to the PID Output.<br>PID_ISA: Same as PID_IND.<br>Kp is generally the first gain set when adjusting a PID loop. | 0.01<br>CV%/PV%<br>%CV / %PV | 0 to 327.67% |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 7<br>(Address + 6) | **PID_IND: Derivative Gain (Kd)**<br>**PID_ISA: Derivative Time (Td = Kd)**<br>PID_IND: Change in the Control Variable in CV Counts if the Error or PV changes 1 PV Count every 10 ms. Entered as an integer representing a fixed-point decimal time in seconds with two decimal places. The least significant digit represents 0.01 second (10ms.) units. Displayed as seconds with two decimal places.<br>For example, Kd entered as 120 is displayed as 1.20 Sec and results in a Kd * Δ Error / delta time or 120 * 4 / 3 contribution to the PID Output if Error changing by 4 PV Counts every 30ms. Kd can be used to speed up a slow loop response but is very sensitive to PV input noise. This noise sensitivity can be reduced by using the derivative filter, which is enabled by<br>setting bit 5 of the **Config Word** (later in this table)<br>PID_ISA: The ISA derivative time in seconds, Td, is entered and displayed in the same way as Kd. Total derivative contribution to PID Output is<br>Kc * Td * Δ Error / dt. | 0.01<br>seconds | 0 to 327.67<br>sec |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 8 (Address + 7) | **PID_IND: Integral Rate (Ki)**<br>**PID_ISA: Integral Rate (1/Ti = Ki)**<br>PID_IND: Rate of change in the Control Variable in CV Counts per second when the Error is a constant 1 PV Count. Entered as an integer representing a fixed-point decimal rate with three decimal places. The least significant digit represents 0.001 counts per second, or 1 count per 0.001 second. Displayed as Repeats/Sec with three decimal places.<br>For example, Ki entered as 1400 is displayed as 1.400 Repeats/Sec and results in a Ki * Error * dt or 1400 *<br>20 * 50/1000 = 1,400 contribution to PID Output for an Error of 20 PV Counts and a 50 ms. CPU sweep time (Sample Period of 0).<br>PID_ISA: The ISA Integral Time in seconds, Ti, must be inverted and entered, as integral rate, as described for PID_IND. Total integral contribution to PID Output is Kc * Ki * Error * dt. Ki is usually the second gain set after Kp. | Repeats/ 0.001 Sec. | 0 to 32.767 repeats/sec |
| 9 (Address + 8) | **CV Bias/Output Offset**<br>Number of CV Counts added to the PID Output before the rate and amplitude clamps. It can be used to set non-zero CV values when only Kp Proportional gains are used, or for feed-forward control of this PID loop output from another control loop. | CV Counts | -32768 to 32767 (add to PID output) |
| 10, 11 (Address + 9. Address + 10) | **CV Upper Clamp**<br>**CV Lower Clamp**<br>Number of CV Counts that define the highest and lowest value that CV is allowed to take. These values are required. The Upper Clamp must have a more positive value than the Lower Clamp, or the PID block will not work. These are usually used to define limits based on physical limits for a CV output. They are also used to scale the Bar Graph display for CV. The PID block has anti-reset-windup, controlled by bit 4 of the Config Word, to modify the integral term value when a CV clamp is reached. | CV Counts | -32768 to 32767 (Word 10 must be greater than word 11.) |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 12 (Address + 11) | **Minimum Slew Time**<br>Minimum number of seconds for the CV output to move from 0 to full travel of 100% or 32000 CV Counts. It is an inverse rate limit on how fast the CV output can change.<br>If positive, CV cannot change more than 32000 CV Counts times the solution time interval (seconds) divided by Minimum Slew Time.<br>For example, if the Sample Period is 2.5 seconds and the Minimum Slew Time is 500 seconds, CV cannot change more than 32000 * 2.5 / 500 or 160 CV Counts per PID solution.<br>The integral term value is adjusted if the CV rate limit is exceeded.<br>When Minimum Slew Time is 0, there is no CV rate limit. Set Minimum Slew Time to 0 while tuning or adjusting PID loop gains. | Seconds / Full Travel | 0 (none) to 32000 sec to move full CV travel |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 13<br>(Address + 12) | **Config Word**<br>The low 6 bits of this word are used to modify default PID settings. The other bits should be set to 0.<br>**Bit 0:** Error Term Mode. When this bit is 0, the error term is SP - PV. When this bit is 1, the error term is PV - SP. Setting this bit to 1 modifies the standard PID Error Term from the normal (SP – PV) to (PV – SP), reversing the sign of the feedback term. This mode is used for reverse acting controls where the CV must go down when the PV goes up.<br>**Bit 1**: Output Polarity. When this bit is 0, the CV output is the output of the PID calculation. When it is set to 1, the CV output is the negated output of the PID calculation. Setting this bit to 1 inverts the Output Polarity so that CV is the negative of the PID output rather than the normal positive value.<br>**Bit 2:** Derivative action on PV. When this bit is 0, the derivative action is applied to the error term. When it is set to 1, the derivative action is applied to PV only.<br>**Bit 3:** Deadband action. When the Deadband action bit is 0, the actual error value is used for the PID calculation. When the Deadband action bit is 1, deadband action is chosen. If the error value is within the deadband limits, the error used for the PID calculation is forced to be zero. If, however, the error value is outside the deadband limits, the magnitude of the error used for the PID calculation is reduced by the deadband limit (\|error\| = \|error – deadband limit\|).<br>**Bit 4**: Anti-reset windup action. When this bit is 0, the anti-reset-windup action uses a reset (integral term) back-calculation. When the output is clamped, the accumulated integral term is replaced with whatever value is necessary to produce the clamped output exactly. When the bit is 1, the accumulated term is replaced with the value of the integral term at the start of the calculation. In this way, the pre-clamp integral value is retained as long as the output is clamped. This option is not recommended for new applications. See “CV Amplitude and Rate Limits” on page 14-15.<br>**Bit 5**: Enable derivative filtering. When this bit is set to 0, no filtering is applied to the derivative term. When set to 1, a first order filter is applied. This will limit the effects of higher frequency process disturbances, such as measurement noise, on the derivative term.<br>Setting Config Word: Set Config Word to 0 for default operation. Add 1 (16#0001) to set bit 0, add 2 (16#0002) to set bit 1, add 4 (16#0004) to set bit 2, add 8 (16#0008) to set bit 3, add 16 (16#0010) to set bit 4, and add 32 (16#0020) to set bit 5. For example, to set bits 0, 3 and 5 only, set Config Word to 1 + 8 + 32 = 41 (16#0029). Some users will find the Config Word value easier to interpret in hexadecimal (16#) format. | Low 6 bits used | Boolean |

<table>
<tr><th>Words</th><th>Parameter/Description</th><th>Low Bit Units</th><th>Range</th></tr>
<tr><td>14 (Address + 13)</td><td>Manual Command<br>Set to the current CV output while the PID block is in Automatic mode. When the block is switched to Manual mode, this value is used to set the CV output and the internal value of the integral term within the Upper and Lower Clamp and Slew Time limits.</td><td>CV Counts</td><td>Tracks CV in Auto or sets CV in Manual</td></tr>
<tr><td>15 (Address + 14)</td><td>Control Word<br>If the Override bit (bit 0) is set to 1, the Control Word and the internal SP, PV and CV parameters must be used for remote operation of the PID block (see below). This allows a remote operator interface device, such as a computer, to take control away from the PLC program.<br>Caution: If you do not want to allow remote operation of the PID block, make sure the Control Word is set to 0. If the low bit is 0, the next 4 bits can be read to track the status of the PID input contacts as long as the PID Enable contact has power.<br>Control Word is a discrete data structure with the first five bit positions defined in the following format:
<table>
<tr><th>Bit</th><th>Word Value</th><th>Function</th><th>Status or External Action if Override bit is set to 1</th></tr>
<tr><td>0</td><td>1</td><td>Override</td><td>If 0, monitor block contacts below. If 1, set them externally.</td></tr>
<tr><td>1</td><td>2</td><td>Manual / Auto</td><td>If 1, block is in Manual mode. If other numbers, it is in Automatic mode.</td></tr>
</table></td><td>Maintained by the CPU, unless bit 0 (Override) is set to 1.</td><td>Boolean</td></tr>
</table>

| Words | Parameter/Description | | | | Low Bit Units | Range |
|---|---|---|---|---|---|---|
| | 2 | 4 | Enable | Should normally be 1. Otherwise, block is never called. | | |
| | 3 | 8 | UP / Raise | If 1 and Manual (bit 1) is 1, CV is incremented every solution. | | |
| | 4 | 16 | DN / Lower | If 1 and Manual (bit 1) is 1, CV is decremented every solution. | | |
| 16 (Address + 15) | **Internal SP** Tracks the SP input. If Override = 1, must be set externally to solve the PID algorithm using an alternate SP value. The original SP value is maintained until overwritten. | | | | Set and maintained by the CPU, unless bit 10 (Override) of Control Word is set to 1. | Non-configurable, unless bit 10 (Override) of Control Word is set to 1. |
| 17 (Address +16) | **Internal CV** Tracks CV output. | | | | Set and maintained by the CPU. | Non-configurable. |
| 18 (Address +17) | **Internal PV** Tracks PV input. Must be set externally if Override bit is set to 1. | | | | Set and maintained by the CPU, unless bit 10 (Override) of Control Word is set to 1. | Non-configurable unless bit 10 (Override) of Control Word is set to 1. |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 19<br>(Address +18) | **Output**<br>Signed word value representing the output of the function block before the optional inversion. If the output polarity bit in the Config Word is set to 0, this value equals the CV output. If the output polarity bit is set to 1, this value equals the negative of the CV output. | Set and maintained by the CPU. | Non-configurable. |
| 20<br>(Address +19) | **Derivative Term Storage**<br>Used internally for storage of intermediate values. Do not write to this location. | | |
| 21, 22<br>(Address + 20. Address + 21) | **Integral Term Storage**<br>Used internally for storage of intermediate values. Do not write to these locations. | | |
| 23 (Address +22) | **Slew Term Storage**<br>Used internally for storage of intermediate values. Do not write to this location. | | |
| 24 – 26<br>(Address +<br>23 – Address<br>+ 25) | **Previous Solution Time**<br>Internal storage of (time of last PID solution. Normally do not write to these locations. Some special circumstances may justify writing to these locations.<br>Note: If you call the PID block in Automatic mode after a long delay, you might want to use SVC_REQ #16 or SVC_REQ #51 to load the current CPU elapsed time clock into Word 24 to update the last PID solution time to avoid a step change of the integral term. | Set and maintained by the CPU. | Non-configurable. |
| 27 (Address + 26) | **Integral Remainder Storage**<br>Holds remainder from integral term scaling. | Set and maintained by the CPU. | Non-configurable. |
| 28, 29 (Address +<br>27, Address<br>+ 28) | **SP, PV Lower Range**<br>**SP, PV Upper Range**<br>Optional integer values in PV Counts that define high and low display values for SP and PV. (Word 29 must be greater than word 28.) | PV Counts | -32768 to 32767 |
| 30 (Address + 29) | **Reserved**<br>Word 30 is reserved. Do not use this location. | N/A | Non-configurable. |

| Words | Parameter/Description | Low Bit Units | Range |
|---|---|---|---|
| 31, 32 (Address + 30, Address + 31) | **Previous Derivative Term Storage** Used in calculations for derivative filter. Do not write to these locations. | Set and maintained by the CPU. | Non-configurable. |
| 33 –40 (Address + 32 –Address + 39) | **Reserved** Words 32-39 are reserved. Do not use these references. | N/A | Non-configurable |

# 14.3 Operation of the PID Function

## 14.3.1 Automatic Operation

The PID function can be called as often as every 10 milliseconds by providing power flow to the Enable input and no power flow to the Manual input. The function block compares the current CPU time with the last PID solution time stored in the reference array. If the interval between the two times is equal to or greater than the Sample Period (word 3 of the reference array) and also equal to or greater than 10 milliseconds, the PID algorithm is solved using this time interval. Both the last solution time and CV output are updated. In Automatic mode, the output CV is placed in the Manual Command parameter (word 14 of the reference array).

***Note:*** *If you call the PID block in Auto mode after a long delay, you may want to use SVC_REQ 16 or SVC_REQ 51 to load the current CPU time into the stored previous solution time (word 24 of the reference array, described on page 14-9). This will update the last PID solution time and avoid a large step change of the integral term.*

## 14.3.2 Manual Operation

The PID function block is placed in Manual mode by providing power flow to both the Enable and Manual input contacts. The output CV is set from the Manual Command parameter. If either the UP or DN inputs have power flow, the Manual Command word is incremented (UP) or decremented (DN) by one CV count every PID solution. For faster manual changes of the output CV, it is also possible to add or subtract any CV count value directly to from the Manual Command word (word 14 of the reference array).

The PID function block uses the CV Upper Clamp and CV Lower Clamp parameters to limit the CV output. If a positive Minimum Slew Time (word 12 of the reference array) is defined, it is used to limit the rate of change of the CV output. If either CV Clamp or the rate of change limit is exceeded, the value of the integral (reset) term is adjusted so that CV is at the limit. The anti-reset- windup feature assures that when the error term tries to drive CV above (or below) the clamps for a long period of time, the CV output will move off the clamp immediately when the error term changes sufficiently.

This operation, with the Manual Command tracking CV in Automatic mode and setting CV in Manual mode, provides a bump-less transfer between Automatic and Manual modes. The CV Upper and Lower Clamps and the Minimum Slew Time always apply to the CV output in Manual mode and the integral term is always updated. This assures that when a user rapidly changes the Manual.

Command value in Manual mode, the CV output cannot change any faster than the slew rate limit set by the Minimum Slew Time, and the CV cannot go above the CV Upper Clamp limit or below the CV Lower Clamp limit.

### 14.3.3 Time Interval for the PID Function

The start time of each PLC sweep is used as the current time when calculating the time interval between solutions of the PID function. The times and time intervals have a resolution of 100 microseconds. When an application uses multiple PID functions, all of them use the same time value.

The PID algorithm is solved when the current time is equal to or greater than the time of the last PID solution plus the Sample Period or 10 milliseconds; whichever is larger. If the Sample Period is set for execution on every sweep (value = 0), the PID function is restricted to a minimum of 10 milliseconds between solutions. If the sweep time is less than 10 milliseconds, the PID function waits until enough sweeps have occurred to accumulate an elapsed time of 10 milliseconds. For example, **if the sweep time is 9 milliseconds, the PID function executes every other sweep, and the time interval between solutions is 18 milliseconds**. If a specific PID function is executed more than once per sweep (by referencing the same reference array location in multiple PID function blocks), the algorithm is solved only on the first call.

The longest possible interval between executions is 65,535 times 10 milliseconds, or 10 minutes, 55.35 seconds.

## 14.4 PID Algorithm Selection (PIDISA or PIDIND) and Gain Calculations

The PID function supports both the Independent Term (PID_IND) and ISA standard (PID_ISA) forms of the PID algorithm. The Independent Term form takes its name from the fact that the coefficients for the proportional, integral and derivative terms act independently. The ISA algorithm is named for the Instrument Society of America (now the International Society for Measurement and Control), which standardized and promoted it.

The two algorithms differ in how words 6 through 8 of the reference array are used and in how the PID output (CV) is calculated.

The Independent term PID (PID_IND) algorithm calculates the output as:

PID Output = Kp * Error + Ki * Error * dt + Kd * Derivative + CV Bias

where Kp is the proportional gain, Ki is the integral rate, Kd is the derivative time, and dt is the time interval since the last solution.

The ISA (PID_ISA) algorithm has different coefficients for the terms:

PID Output = Kc * (Error + Error * dt/Ti + Td * Derivative) + CV Bias

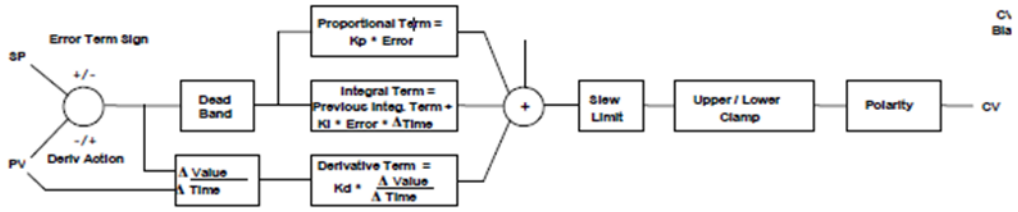
where Kc is the controller gain, Ti is the Integral time and Td is the Derivative time. The advantage of PID_ISA is that adjusting Kc changes the contribution for the integral and derivative terms as well as the proportional term, which can simplify loop tuning.

If you have the PID_ISA Kc, Ti and Td values, use the following equations to convert them to use as:PID_IND parameters:

Kp = Kc, Ki = Kc/Ti, and Kd = Kc * Td

The following diagram shows how the PID_IND algorithm works:

**Figure 214**




The ISA Algorithm (PID_ISA) is similar except that its Kc gain coefficient is applied after the three terms are summed, so that the integral gain is Kc / Ti and the derivative gain is Kc * Td.

Bits 0, 1 and 2 in the Config Word set the Error sign, Output Polarity and Derivative Action, respectively.

## 14.4.1 Error Term

Both PID algorithms calculate the Error term as

Error = (SP – PV),

which can be changed to Reverse Acting mode:

Error = (PV – SP)

by setting the Error Term mode (bit 0) in the Config Word – (word 13 of the reference array) to 1.

Reverse -Acting mode is used if you want the CV output to move in the opposite direction from PV input changes (CV down for PV up) instead of the normal CV up for PV up.

| Bit 0 of Config Word (Word 13) | Error Term Mode | Error Term |
|---|---|---|
| 0 | Normal | SP – PV |
| 1 | Reverse Acting | PV – SP |

## 14.4.2 Derivative Term

The Derivative Term is Kd (word 7 of the reference array) multiplied by the time rate of change of the Error term in the interval since the last PID solution.

Derivative = Kd * ΔError / dt = Kd * (Error – previous Error) / dt,

where

dt = Current PLC time – PLC time at previous PID solution.

Two bits in the Config Word (word 13 of the reference array) affect the calculation of ΔError. There are four cases to consider.

In Normal mode, the change in the error term is:

ΔError = (Error – previous Error) = (SP – PV) – (previous SP – previous PV)

= (previous PV – PV) – (previous SP – SP) = – ΔPV + ΔSP

= ΔSP – ΔPV

where

ΔPV = (PV – previous PV), and ΔSP = (SP – previous SP).

However, in Reverse-Acting mode, the current error term is (PV – SP), and the sign of the change in the error term is reversed:

ΔError = (Error – previous Error) = (PV – SP) – (previous PV – previous SP)

= (PV – previous PV) – (SP – previous SP) =ΔPV – ΔSP.

The change in the error term depends on changes in both SP and PV. If SP is constant,

ΔSP = 0,

and SP has no effect on the derivative term. When SP changes, however, it can cause large transient swings in the derivative term and hence the output. Loop stability may be improved by eliminating the effect of SP changes on the derivative term. To calculate the Derivative based only on the change in PV, set bit 2 of the Config Word to 1. This modifies the equations above by assuming SP is constant (ΔSP = 0).

For bit 2 set in normal mode (bit 0 = 0): ΔError = – ΔPV,

For bit 2 set in Reverse-Acting mode (bit 0 = 1): ΔError = ΔPV.

## 14.4.3 CV Bias Term

The CV Bias term (word 9 in the reference array) is an additive term separate from the PID inputs. It may be useful if you are using only Proportional gain (Kp) and you want the CV to be a non-zero value when the PV equals the SP and the Error is 0. In this case, set the CV Bias to the desired CV when the PV is at the SP. CV Bias can also be used for feed forward control where another PID loop or control algorithm is used to adjust the CV output of this PID loop.

If a non-zero Integral rate is used, the CV Bias will normally be 0 as the integral term acts as an automatic bias or "reset." Just start up in Manual mode and use the Manual Command word (word 14 of the reference array) to set the desired CV, and then switch to Automatic mode. This will immediately calculate the required value for the integral term.

### 14.4.4 CV Amplitude and Rate Limits

The PID block does not send the calculated Output directly to CV. Both PID algorithms can impose amplitude and rate of change limits on the output Control Variable. If the Minimum Slew Time (word 12 of the reference array) is non-zero, the rate of change (slew rate) limit is determined by dividing the maximum CV value (32,000) by the Minimum Slew Time. For example, if the Minimum Slew Time is 100 seconds, the rate limit will be 320 CV counts per second. If the solution interval was 50 milliseconds, the new CV output cannot change more than 320*50/1000 or 16 CV counts from the previous CV output.

The CV output is then compared to the CV Upper Clamp and CV Lower Clamp values (words 10 and 11 of the reference array). If CV is outside either limit, the CV output is clamped to the appropriate limit value. When the CV output is modified to impose either slew rate or amplitude limits (or both) the stored integral term would normally accumulate a large value over time. This phenomenon is known as reset windup. Reset windup introduces errors in CV after the PID output no longer needs to be limited. For example, windup would prevent the CV output from moving off a clamp value immediately.

There are two optional methods for preventing reset windup. If the Anti-resetwindup Action bit (bit 4) of Config Word (word 13 of the reference array) is zero (the default), the integral term is adjusted at each PID solution to match the error input and limited CV output exactly. When PV changes while CV is clamped, or when CV is both rate and amplitude limited in a particular PID solution, this option assures that a smooth transition will always occur after CV is no longer limited.

If the Anti-reset-windup Action bit of Config Word is set, then the integral term stored on the previous PID solution is simply retained as long as CV is limited. This option was added to assure compatibility with existing PID applications when the default action described above was introduced. This option is not recommended for new applications.

Finally, the PID block checks the Output Polarity (bit 2 of the Config Word ) and changes the sign of the output if the bit is 1.

CV = Clamped PID Output if Output Polarity bit set, or

Clamped PID Output if Output Polarity bit cleared.

If the block is in Automatic mode, the final CV is placed in the Manual Command (word 14 of the reference array). If the block is in Manual mode, the PID equation is skipped because CV is set by the Manual Command, but the slew rate and amplitude limits are still checked. This assures that the Manual Command cannot change the output above the CV Upper Clamp or below the CV Lower Clamp, and the output cannot change faster than allowed by the Minimum Slew Time.

## 14.4.5 Sample Period and PID Function Block Scheduling

The PID function block is a digital implementation of an analog control function, so the dt sample time in the PID Output equation is not the infinitesimally small sample time available with analog controls. The majority of processes being controlled can be approximated as a gain with a first or second order lag, and (possibly) a pure time delay.The PID function block sets a CV output to the process and uses the process feedback PV to determine an Error to adjust the next CV output. A key process parameter is the total time constant, which is how fast the process can change PV when the CV is changed. As discussed in the section, Determining the Process Characteristics, the total time constant, Tp+Tc, for a first order system is the time required for PV to reach 63% of its final value when CV is stepped. The PID function block will not be able to control a process unless its Sample Period is well under half the total time constant. Larger Sample Periods will make it unstable.

The Sample Period should be no bigger than the total time constant divided by 10 (or down to 5 worst case). For example, if PV seems to reach about 2/3 of its final value in 2 seconds, the Sample Period should be less than 0.2 seconds, or 0.4 seconds worst case.

On the other hand, the Sample Period should not be too small, such as less than the total time constant divided by 1000, or the Ki * Error * dt term for the PID integral term will round down to 0. For example, a very slow process that takes 10 hours or 36,000 seconds to reach the 63% level should have a Sample Period of 40 seconds or longer.

Variations of the time interval between PID function solutions can have short term effects on the CV output. For example, if a step change to PV caused by measurement noise occurs between solutions, the value of the derivative term will be inversely proportional to the time interval. The performance of PID loops that are tuned for quick response may be improved when the solution interval is held constant by configuring the PLC CPU for constant sweep mode. Depending on the CPU model and the application, constant sweep times of 10 milliseconds, integer multiples of 10 milliseconds, or exact divisors of 10 milliseconds (1, 2 or 5 milliseconds) will be possible. The Sample Period can then be set for a suitable multiple of 10 milliseconds.

If many PID loops are used, allowing the application to solve all the loops on the same sweep may lead to wide variations in CPU sweep time. If the loops have a common Sample Period that is at least equal to the number of PID loops times the sweep time, a simple solution is to sequence one or more 1 bits through an array of zero bits and to use these bits to enable power flow to individual PID function blocks. The logic should assure that each PID function block is enabled no more often than its Sample Period.

# 14.5 Determining the Process Characteristics

The PID loop gains, Kp, Ki and Kd, are determined by the characteristics of the process being controlled. Two key questions when setting up a PID loop are:

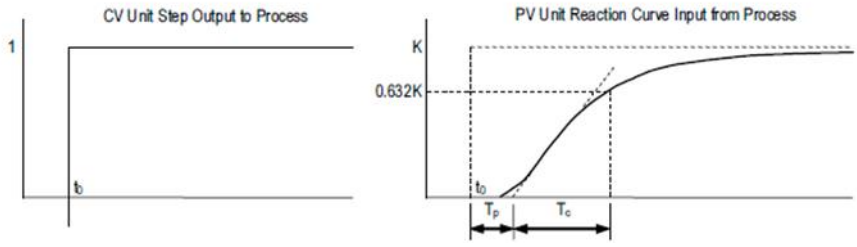
- How big is the change in PV when CV is changed by a fixed amount, or what is the open loop gain of the process?
- How fast does the system respond, or how quickly does PV change after the CV output is stepped?

Many processes can be approximated by a process gain, first or second order lag and a pure time delay. In the frequency domain, the transfer function for a first order lag system with a pure time delay is:

$$\frac{PV(s)}{CV(s)} = G(s) = Ke^{-T_p/(1+T_c s)}$$

Plotting the response to a step input at time t0 in the time domain provides an open-loop unit reaction curve:

**Figure 215**




The following process model parameters can be determined from the PV unit reaction curve

| | |
|---|---|
| K | Process open loop gain = final change in PV/change in CV at time $t_0$ |
| $T_p$ | Process or pipeline time delay or dead time after $t_0$ before the process output PV starts moving |
| $T_c$ | First order Process time constant, time required after $T_p$ for PV to reach 63.2% of the final PV |

One way to measure these parameters is by putting the PID function block in Manual mode, making a small step change in the CV output by changing the Manual Command (word 14 of the reference array), and then plotting the PV response over time. For slow processes this can be done manually, but for faster processes a chart recorder or computer graphic data-logging package will help. The CV step size should be large enough to cause an observable change in PV, but not so large that it disrupts the process being measured. A good step size may be from 2 to 10% of the difference between the CV Upper and CV Lower Clamp values.

# 14.6 Setting Tuning Loop Gains

## 14.6.1 Basic Iterative Tuning Approach

Because PID parameters are dependent on the process being controlled, there are no predetermined values that will work. However, a simple iterative process can be used to find acceptable values for Kp, Ki, and Kd gains.

1. Set all the reference array parameters to 0, then set the CV Upper and CV Lower Clamps to the highest and lowest CV expected. Set the Sample Period to a value within the range Tc/10 to Tc/100, where Tc is the estimated process time constant defined in the previous section, Determining the Process Characteristics.
2. Put the PID function block in Manual mode and set the Manual Command (word 14 in the reference array) to different values to check if CV can be moved to Upper and Lower Clamp. Record the PV value at some CV point and load it into SP.
3. Set a small gain, such as 100 * Maximum CV/Maximum PV, into Kp and turn off Manual mode. Step SP by 2% to 10% of the Maximum PV range and observe PV response. Increase Kp if PV step response is too slow or reduce Kp if PV overshoots and oscillates without reaching a steady value.
4. Once a Kp is found, start increasing Ki to get overshooting that dampens out to a steady value in two to three cycles. This may require reducing Kp. Also try different SP step sizes and CV operating points.
5. After suitable Kp and Ki gains are found, try adding Kd to get quicker responses to input changes providing it doesn't cause oscillations. Kd is often not needed and will not work with noisy PV.
6. Check gains over different SP operating points and add Dead Band and Minimum Slew Time if needed. Some Reverse Acting processes may need setting of Config

Word Error Term or Output Polarity bits.

## 14.6.2 Setting Loop Gains Using the Ziegler and Nichols Tuning Approach

This approach provides good response to system disturbances with gains producing an amplitude ratio of 1/4. The amplitude ratio is the ratio of the second peak over the first peak in the closed loop response.

1. Determine the three process model parameters, K, Tp and Tc for use in estimating initial PID loop gains.
2. Calculate the Reaction rate:

   $$R = K/Tc$$

   For Proportional control only, calculate Kp as:

   $$Kp = 1/(R * Tp) = Tc/(K * Tp)$$

   For Proportional and Integral control, use:

Kp = 0.9/(R * Tp) = 0.9 * Tc/(K * Tp) Ki = 0.3 * Kp/Tp

For Proportional, Integral and Derivative control, use:

Kp = G/(R * Tp) where G is from 1.2 to 2.0

Ki = 0.5 * Kp/Tp

Kd = 0.5 * Kp * Tp

3. Check that the Sample Period is in the range

   (Tp + Tc)/10 to (Tp + Tc)/1000

## 14.6.3 Ideal Tuning Method

The Ideal Tuning procedure provides the best response to SP changes that are delayed only by the Tp process delay or dead time.

1. Determine the three process model parameters, K, Tp and Tc for use in estimating initial PID loop gains.
2. Calculate Kp, Ki, and Kd as follows:

   Kp = 2 * Tc/(3 * K * Tp)

   Ki = Tc

   Kd = Ki/4 if Derivative term is used

3. Once initial gains are determined, convert them to integers.
4. Calculate the Process gain, K, as a change in input PV Counts divided by the resulting output step change in CV Counts. (Not in process PV or CV engineering units.) Specify all times in seconds.
5. Once Kp, Ki and Kd are determined, Kp and Kd are multiplied by 100 while Ki is multiplied by 1000. The resulting values are entered into the corresponding reference array word locations.

## 14.6.4 Example

The following PID example has a sample period of 100 ms, a Kp gain of 4.00 and a Ki gain of 1.500. The set point is stored in %R0001, the control variable is output in % AQ0002, and the process variable is returned in %AI0003. CV Upper and CV Lower Clamps must be set, in this case to 20000 and 4000, and an optional small Dead Band of +5 and -5 is included. The 40-word reference array starts in %R0100. Normally, user parameters are set in the reference array, but %M0006 can be set to reinitialize the 14 words starting at %R0102 (word 3) from constants stored in logic (a useful technique).

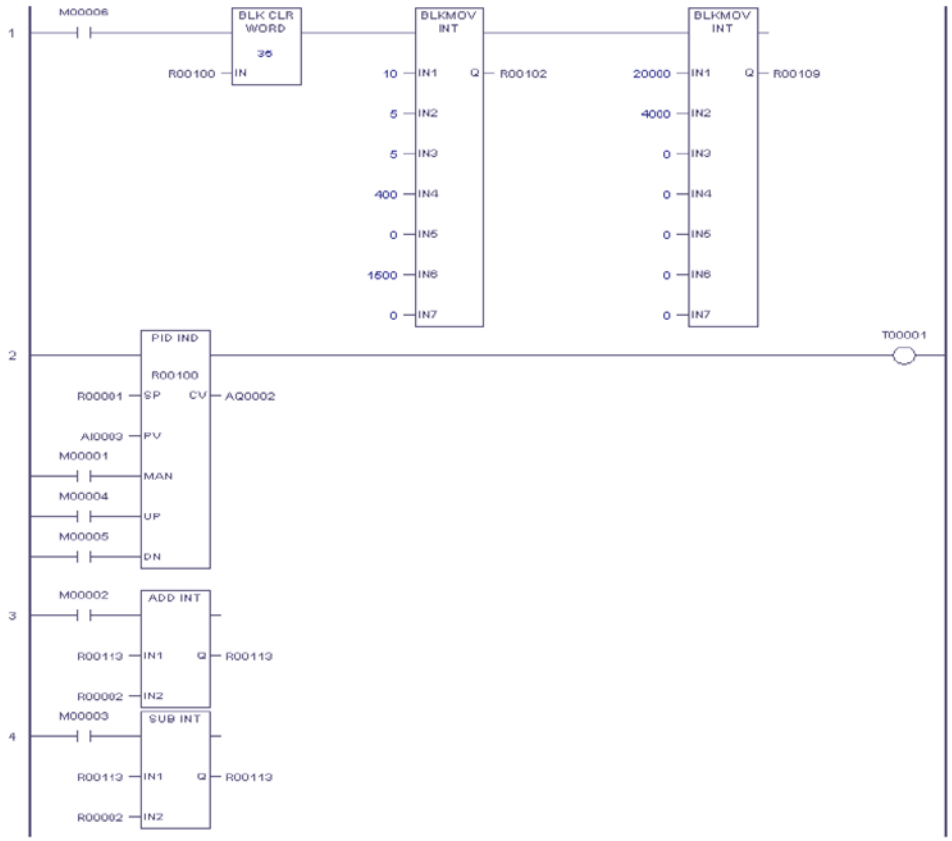
The block can be switched to Manual mode with %M1 so that the Manual Command, %R113, can be adjusted. Bits %M4 or %M5 can be used to increase or decrease %R113 and the PID CV by 1 every 100 ms solution. For faster manual operation, bits %M2 and %M3 can be used to add or subtract the value in %R2 to/from %R113 every CPU sweep. The %T1 output is on when the PID is OK.

## Reference Array Initialization using %M00006

For details on the contents of the reference array, refer to the section, Reference Array for the PID Function earlier in this chapter

| Reference Array Initialization using %M00006 | Reference Array Initialization using %M00006 | Reference Array Initialization using %M00006 | Reference Array Initialization using %M00006 |
|---|---|---|---|
| 3 | Sample Period | %R102 | 10 |
| 4 | + Dead Band | %R103 | 5 |
| 5 | - Dead Band | %R104 | 5 |
| 6 | Kp | %R105 | 400 |
| 7 | Kd | %R106 | 0 |
| 8 | Ki | %R107 | 1500 |
| 9 | CV Bias | %R108 | 0 |
| 10 | CV Upper Clamp | %R109 | 2000 |
| 11 | CV Lower Clamp | %R110 | 400 |
| 12 | Minimum Slew Time | %R111 | 0 |
| 13 | Config Word | %R112 | 0 |
| 14 | Manual Command | %R113 | 0 |
| 15 | Control Word | %R114 | 0 |
| 16 | Internal SP | %R115 | 0 |

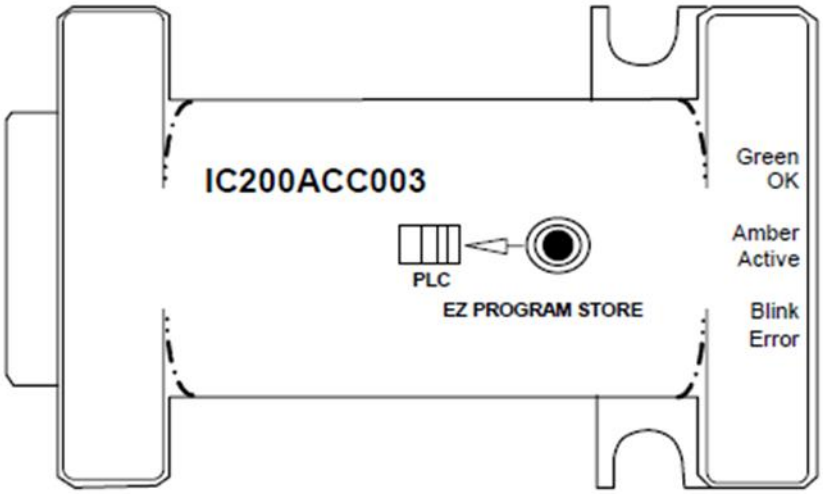
**Figure 216**


1
M00006
BLK CLR WORD
36
R00100
IN
BLKMOV INT
10
5
5
400
0
1500
0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
Q
R00102
BLKMOV INT
20000
4000
0
0
0
0
0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
Q
R00109
2
T00001
PID IND
R00100
R00001
SP
CV
AQ0002
AI0003
PV
M00001
MAN
M00004
UP
M00005
DN
3
M00002
ADD INT
R00113
IN1
Q
R00113
R00002
IN2
4
M00003
SUB INT
R00113
IN1
Q
R00113
R00002
IN2

# Chapter 15: The EZ Program Store Device

This chapter describes the VersaMax EZ Program Store device (IC200ACC003), which can be used to transfer program, configuration, and reference tables data from one PLC to one or more others of the same type.

**Figure 217**



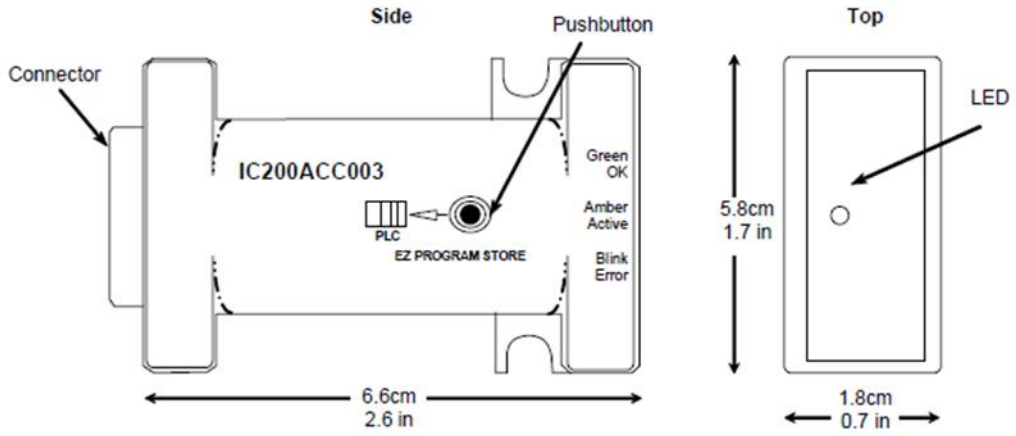

Contents of this chapter:

- Description of the EZ Program Store device
- Details of Using the EZ Program Store device
- Read/Write/Verify Data with a Programmer Present
- Write Data to a PLC CPU without a Programmer Present

The EZ Program Store device (IC200ACC003) can be used to store and update the configuration, application program, and reference tables data of a VersaMax PLC. The update can include Ethernet Global Data and Advanced User Parameters for Ethernet. A programmer and PLC CPU are used to initially write data to the device. In addition to writing data to the device, the programmer can read data already stored on an EZ Program Store device, and compare that data with similar files already present in the programmer.

Once the data is written to the EZ Program Store device, the data can be written to one or more other PLC CPUs of the same type, with no programmer needed.

**Figure 218**




The EZ Program Store device and PLC must both have no OEM key password or the same OEM key password for an update to occur. The EZ Program Store device does not perform special processing for other types of passwords.

The EZ Program Store device plugs directly into port 2 on a VersaMax PLC. No cables or connectors are required. Power for the device comes from port 2. Because the EZ Program Store device is not used during normal operation, it does not need to be screwed down to the PLC. The device can be hot inserted and hot removed without disrupting the system.

### Features

- 2-Megabit Serial Data Flash for non-volatile storage
- Pushbutton initiates update from the device to a PLC
- Dual color status LED
- Configurable OEM key password protection
- Compatible with all VersaMax CPU models, release 2.10 and later.

## 15.1 Read/Write/Verify Data with a Programmer Present

With a programmer present, the PLC CPU can read, write, or verify a program, configuration and tables in the EZ Program Store device. When reading or verifying data, it is possible to select hardware configuration, logic, and/or reference tables data.

However, when writing data to the EZ Program Store device, all three data types must be written. If the hardware configuration includes Ethernet Global Data and/or a file of Advanced User Parameters for Ethernet communications, they will also be included.

The programmer must be using version 1.5 or later of the VersaPro programming software.

**⚠ WARNING**

Do not use the pushbutton on the EZ Program Store device to invoke an update while: 1. Loading program logic, configuration data, and/or reference tables from the PLC to the programmer. 2. Verifying program logic, configuration data, and/or reference tables in the PLC with the programmer. Doing so may corrupt the data being loaded or verified and produce unexpected results. You should power-cycle the PLC to restore normal operation.

### 15.1.1 Including All the Necessary Information

When the EZ Program Store device updates a PLC, it writes over existing configuration, program files and data in the target PLC. Therefore, it is important to be sure that the information placed on the EZ Program Store device is complete for proper operation of the PLC system. For example, if the EZ Program Store device contains an application program, but instead of a customized hardware configuration it contains the default PLC configuration, the update will overwrite any existing configuration data in a PLC being updated. If that happens, the modules in the PLC system will then use their default configuration, which may cause unexpected operation.

### 15.1.2 Matching OEM Protection

If the PLC(s) that will be updated by the EZ Program Store device are protected by an OEM key password, be sure the same OEM key password is present in the configuration stored to the EZ Program Store device, otherwise no update will be possible. If the PLC (s) being updated had no OEM key password assigned, the EZ Program Store device must also not have an OEM key password. The device does not use other system passwords.

(See chapter 7, CPU Operation, for information about passwords and the OEM key).

### 15.1.3 Adjusting the Configuration Timeouts

Reading and writing large programs, hardware configurations, and reference tables to or from the EZ Program Store device may take 30 seconds or more to complete. To avoid possible disconnect errors or read/write errors, adjust the request timeouts in the configuration to 30 - 63 seconds (30,000 - 63,000mS).

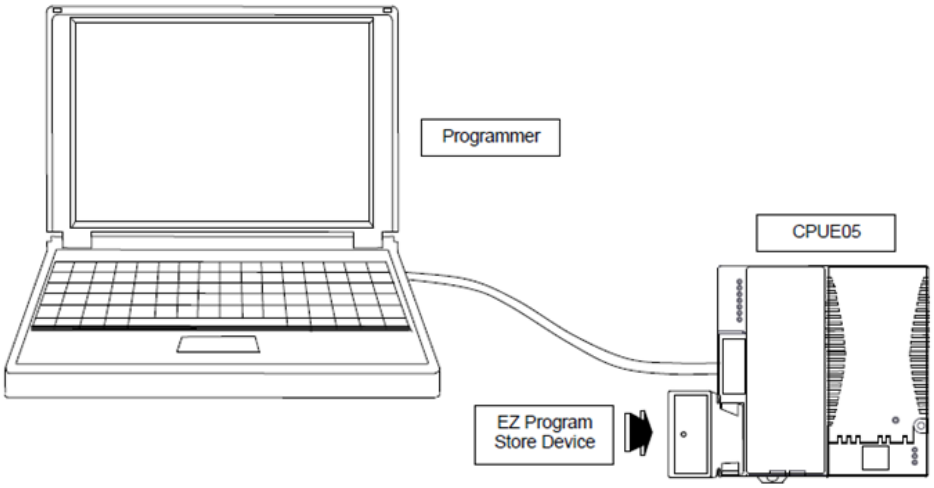
### 15.1.4 Writing Data to RAM or Flash

Folder data is stored from the programmer to the EZ Program Store device in the same way data is stored to Flash memory. Writing to either Flash or to the EZ Program Store device always writes all folder data (regardless of what types are selected). Data stored to the EZ Program Store device is verified in the same manner as data stored in Flash memory is verified. Data can also be read from the device in the same manner as reading from Flash.

The EZ Program Store device can be used to update data in a PLC's RAM memory only, or in both RAM and Flash memory. In the configuration data stored to the EZ Program Store device, be sure to specify which type of memory should be updated. Select RAM only to update only RAM memory in the target PLC. Select RAM & FLASH to update both.

## 15.1.5 Using the EZ Program Store Device with the Programmer

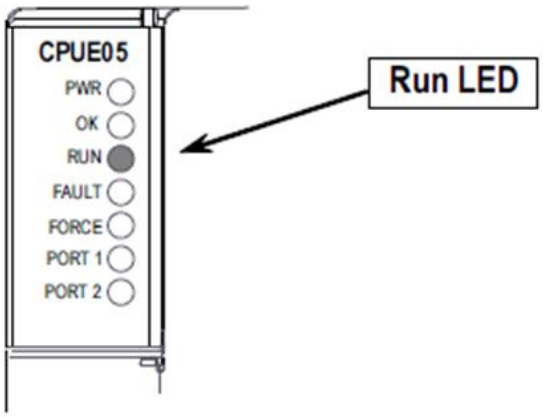
**Figure 219**




To read/write or verify some or all of the data, follow these steps:

1. Plug the EZ Program Store device into port 2 of the VersaMax PLC CPU. The device's LED turns green after about 2 seconds. The delay allows time for proper seating of the device.
2. If the PLC is in Run mode when the EZ Program Store device is connected, the Run LED on the PLC blinks at a 1 Hz rate.

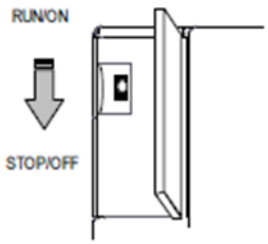
**Figure 220**



This blinking indicates that the Run/Stop switch is enabled, regardless of the configuration of the switch.

3. If the EZ Program Store device's LED is green and the PLC's Run LED is blinking, stop the PLC by moving the Run/Stop switch from the On/Run position to the Stop/Off position.

**Figure 221**




If the switch is already on the Stop/Off position, move it to Run then back to Stop to affirm the change. After the mode is changed to Stop No I/O, the Run LED goes off.

Note that to change the PLC mode from Run to Stop or from Stop to Run mode when an EZ Program Store device is attached, the PLC's Run/Stop switch must be used. If a programmer (computer) is also connected to the PLC at the same time, the programmer cannot be used to change the PLC mode.

4. Start the programming software and change the request timeout values as needed.
5. Connect the programmer to the PLC CPU.
6. Use the programming software to read, write, or verify the data.

When performing an update with the programmer present, the pushbutton on the EZ Program Store device is not used.

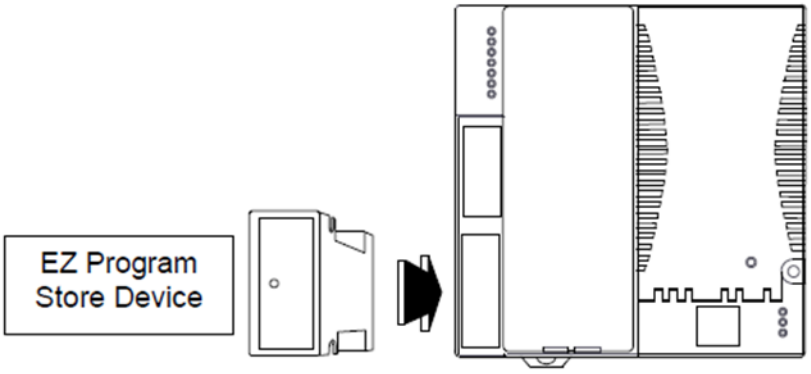
## 15.2 Update a PLC CPU without a Programmer Present

With a program, configuration, tables, Ethernet Global Data, and Advanced User Parameters (if any) already stored in an EZ Program Store device, it can be used to update one or more other PLC CPUs of the same type. All the data stored in the EZ Program Store device will be updated in the PLC CPU.

To update all of the data in a VersaMax PLC CPU, follow these steps:

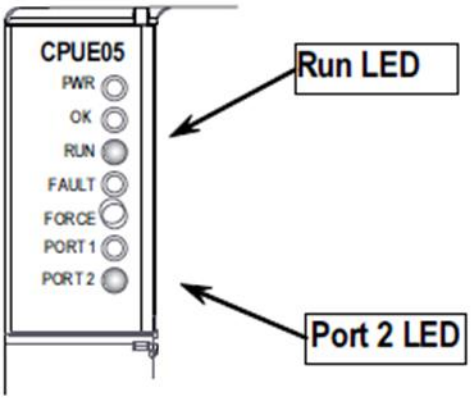
1. Plug the EZ Program Store device into port 2 of the VersaMax PLC CPU.

**Figure 222**



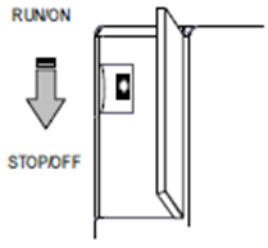

If the PLC is in Run mode when the EZ Program Store device is connected, the Run LED on the PLC blinks at a 1 Hz rate. This blinking indicates that the Run/Stop switch is enabled, regardless of the configuration of the switch.

**Figure 223**




2. If the PLC's Run LED is blinking and the LED on the device is green, stop the PLC by moving the Run/Stop switch from Run/On to Stop/Off position.

**Figure 224**

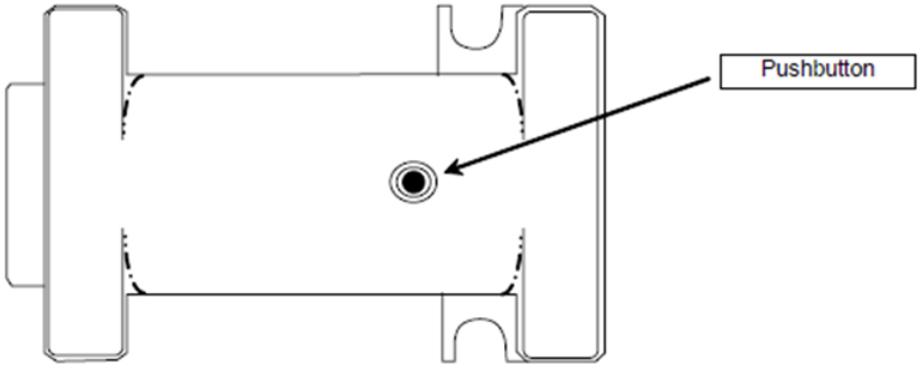



If the switch is already on the Stop/Off position, move it from Run then back to Stop to affirm the change.

After the mode is changed to Stop No I/O, the PLC Run LED goes off.

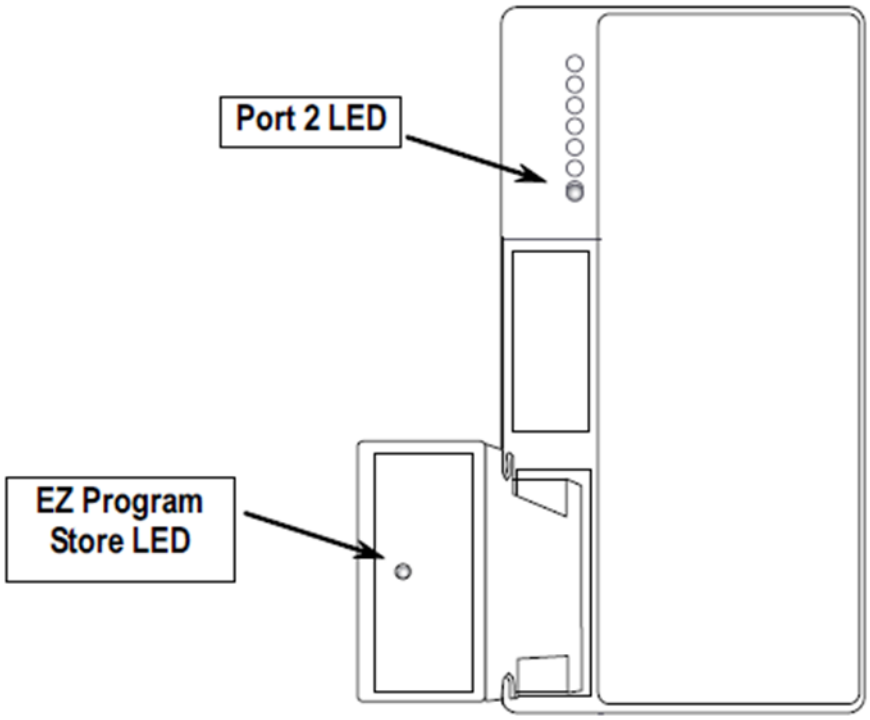
3. To start the update, press the pushbutton on the EZ Program Store device.

**Figure 225**




The LED on the EZ Program Store device turns amber and the Port 2 LED on the PLC blinks.

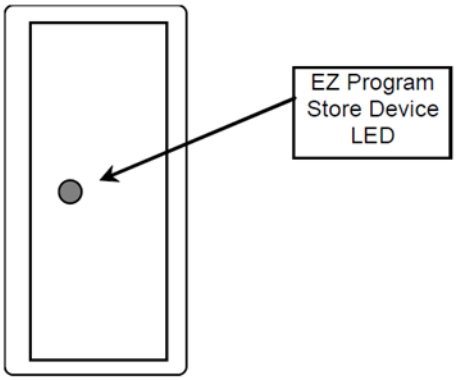
**Figure 226**




4. Wait for the update to complete. Reading and writing large programs, hardware configurations, and reference tables to or from the EZ Program Store device may take 30 seconds or more to complete.

When the device's LED turns solid green and the CPU's Run LED starts blinking, the update has completed successfully.

**Figure 227**




When the PLC is placed into Run mode (by moving the Run/Stop switch from Stop/Off to Run/On position) it uses the new data immediately.

### 15.2.1 Error During Update

If the EZ Program Store device's LED is blinking green/amber and the CPU's Run LED is blinking, an error was detected before the old data was erased. When the PLC is placed into Run mode, it continues using the old data.

If the device's LED is blinking green/amber and the CPU's Run LED is off, an error occurred during the transfer after the data in the PLC was erased. Try the update again by disconnecting and reconnecting the device and pressing the pushbutton. If the second update fails, contact the update provider for service.

Update errors are reported as USD Flash Read faults in the PLC Fault Table. The first two bytes of extra fault data describe the fault.

# Appendix A: Performance Data

This section presents performance data collected on the VersaMax CPUs IC200CPU001, CPU002, CPU005, and CPUE05. The data includes base sweep time, sweep impact of Boolean instructions, function block sweep impact times, function block sizes, and I/O module scan time data.

## A-1 Base Sweep Time

The table below shows the base sweep time with the default program in Run mode, no I/O modules present or configured, and no serial connections to either serial port.

| Model | Time (in milliseconds) |
|---|---|
| CPU001/002 | 1.605 |
| CPU005 | 1.039 |
| CPUE05 | 1.910 |

## A-2 Boolean Instruction Time

This table shows the typical sweep impact time for Boolean instructions

| Model | Typical Time (in microseconds) |
|---|---|
| CPU001/002 | 1.7 |
| CPU005/E05 | 0.8 |

## A-3 Function Block Timing

The following tables show the sweep impact times and size information for all supported function blocks of the CPU.

### A-3.1 Sweep Impact Times

The tables show two sweep impact times are shown for each function. An Increment time is shown for functions that can have variable length inputs (table functions):

**Enabled** Sweep impact time (in microseconds) when a function block has been enabled; power flow to the function block.

**Disabled** Sweep impact time (in microseconds) when a function block has been disabled; no power-flow to function block and/or power-flow to reset of function block.

**Increment** Incremental time (in microseconds/input unit) to add to the base function time for each addition to the length of an input parameter. Only applies to table functions that can have varying input lengths (i.e. Search, Array Moves, etc.).

All timings represent typical execution time. Timings may vary with input and error conditions. Each timing includes the time to execute one contact, and normal overhead including a connection with a programmer.

***Note:*** *Timings listed in previous versions of this manual did not include this overhead.*

- For table functions, increment is in units of length specified.
- For bit operation functions, microseconds/bit.
- For data move functions, microseconds/number of bits or words.
- For functions that have an increment value, multiply the increment by (Length –1) and add that value to the base time to get total instruction time.

## A-3.2 Sizes of Timers, Counters, Math Functions, Trig Functions, Log Functions

The size of a function is the number of bytes consumed in user logic space for each instance of the function in a ladder diagram application program.

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| Timers | On-Delay Timer | 119 | 90 | 90 | 69 | – | 15 |
| | Timer | 110 | 80 | 81 | 60 | – | 15 |
| | Off-Delay Timer | 110 | 80 | 81 | 60 | – | 15 |
| Counters | Up Counter | 90 | 90 | 70 | 70 | – | 13 |
| | Down Counter | 93 | 90 | 70 | 70 | – | 13 |
| Math | Addition (INT) | 62 | 12 | 50 | 10 | – | 13 |
| | Addition (DINT) | 60 | 12 | 50 | 10 | – | 19 |
| | Addition (REAL) | 139 | 12 | 99 | 10 | – | 17 |
| | Subtraction (INT) | 62 | 12 | 50 | 10 | – | 13 |
| | Subtraction (DINT) | 60 | 12 | 50 | 10 | – | 19 |
| | Subtraction (REAL) | 139 | 12 | 100 | 10 | – | 17 |
| | Multiplication (INT) | 70 | 12 | 50 | 10 | – | 13 |
| | Multiplication (DINT) | 99 | 12 | 50 | 10 | – | 19 |
| | Multiplication (REAL) | 155 | 12 | 108 | 10 | – | 17 |
| | Division (INT) | 80 | 12 | 60 | 10 | – | 13 |
| | Division (DINT) | 70 | 12 | 51 | 10 | – | 19 |
| | Division (REAL) | 244 | 12 | 160 | 10 | – | 17 |

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| | Modulo Division (INT) | 84 | 12 | 60 | 10 | – | 13 |
| | Modulo Division (DINT) | 80 | 12 | 60 | 10 | – | 19 |
| | Square Root (INT) | 85 | 12 | 60 | 10 | – | 10 |
| | Square Root (DINT) | 126 | 12 | 70 | 10 | – | 13 |
| | Square Root (REAL) | 514 | 12 | 340 | 10 | – | 11 |
| | Scale (INT) | 112 | 12 | 78 | 10 | – | 22 |
| | Scale (WORD) | 110 | 12 | 73 | 10 | – | 22 |
| Trigonometric | SIN (REAL) | 1432 | 12 | 945 | 10 | – | 11 |
| | COS (REAL) | 1437 | 12 | 945 | 10 | – | 11 |
| | TAN (REAL) | 2135 | 20 | 1400 | 20 | – | 11 |
| | ASIN (REAL) | 1838 | 12 | 1200 | 10 | – | 11 |
| | ACOS (REAL) | 1793 | 12 | 1200 | 10 | – | 11 |
| | ATAN (REAL) | 820 | 12 | 542 | 10 | – | 11 |
| Logarithmic | LOG (REAL) | 878 | 12 | 577 | 10 | – | 11 |
| | LN (REAL) | 821 | 12 | 542 | 10 | – | 11 |

## A-3.3 Sizes of Exponential Functions, Radian Conversion, Relational Functions

The size of a function is the number of bytes consumed in user logic space for each instance of the function in a ladder diagram application program.

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enable | Disable | Enable | Disable | | |
| Exponential | Power of e | 592 | 12 | 393 | 10 | – | 11 |
| | Power of X | 365 | 12 | 249 | 10 | – | 17 |
| Radian | Convert RAD to DEG | 328 | 12 | 214 | 10 | – | 11 |
| Conversion | Convert DEG to RAD | 106 | 12 | 70 | 10 | – | 11 |
| Relational | Equal (INT) | 43 | 12 | 30 | 10 | – | 10 |
| | Equal (DINT) | 50 | 12 | 37 | 10 | – | 16 |
| | Equal (REAL) | 60 | 12 | 41 | 10 | – | 14 |
| | Not Equal (INT) | 40 | 12 | 30 | 10 | – | 10 |
| | Not Equal (DINT) | 45 | 12 | 30 | 10 | – | 16 |

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enable | Disable | Enable | Disable | | |
| | Not Equal (REAL) | 60 | 12 | 40 | 10 | – | 14 |
| | Greater Than (INT) | 40 | 12 | 30 | 10 | – | 10 |
| | Greater Than (DINT) | 45 | 12 | 30 | 10 | – | 16 |
| | Greater Than (REAL) | 60 | 12 | 40 | 10 | – | 14 |
| | Greater Than/Equal (INT) | 40 | 12 | 30 | 10 | – | 10 |
| | Greater Than/Equal (DINT) | 46 | 12 | 30 | 10 | – | 10 |
| | Greater Than/Equal (REAL) | 60 | 12 | 40 | 10 | – | 14 |
| | Less Than (INT) | 40 | 12 | 30 | 10 | – | 10 |
| | Less Than (DINT) | 46 | 12 | 30 | 10 | – | 16 |
| | Less Than (REAL) | 60 | 12 | 40 | 10 | – | 14 |
| | Less Than/Equal (INT) | 40 | 12 | 30 | 10 | – | 10 |
| | Less Than/Equal (DINT) | 46 | 12 | 30 | 10 | – | 16 |
| | Less Than/Equal (REAL) | 60 | 12 | 40 | 10 | – | 14 |
| | Range (INT) | 50 | 12 | 33 | 10 | – | 13 |
| | Range (DINT) | 55 | 12 | 40 | 10 | – | 22 |
| | Range (WORD) | 50 | 12 | 33 | 10 | – | 13 |

## A-3.4 Sizes of Bit Operations, Data Move Functions

The size of a function is the number of bytes consumed in user logic space for each instance of the function in a ladder diagram application program.

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| Bit Operation | Logical AND | 60 | 12 | 50 | 10 | – | 13 |
| | Logical OR | 60 | 12 | 50 | 10 | – | 13 |
| | Logical Exclusive OR | 60 | 12 | 50 | 10 | – | 13 |
| | Logical Invert, NOT | 50 | 12 | 40 | 10 | – | 10 |
| | Shift Bit Left | 134 | 12 | 80 | 10 | 14.78 | 16 |
| | Shift Bit Right | 129 | 12 | 80 | 10 | 16.31 | 16 |

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| | Rotate Bit Left | 110 | 12 | 70 | 10 | 18.45 | 16 |
| | Rotate Bit Right | 111 | 12 | 70 | 10 | 18.41 | 16 |
| | Bit Position | 76 | 12 | 57 | 10 | – | 13 |
| | Bit Clear | 70 | 12 | 56 | 10 | – | 13 |
| | Bit Test | 60 | 12 | 44 | 10 | – | 13 |
| | Bit Set | 70 | 12 | 56 | 10 | – | 13 |
| | Mask Compare (WORD) | 158 | 12 | 110 | 10 | – | 25 |
| | Mask Compare (DWORD) | 150 | 12 | 100 | 10 | – | 25 |
| | Bit Sequencer | 150 | 109 | 101 | 77 | 0.24 | 16 |
| Data Move | Move (INT) | 45 | 12 | 32 | 10 | 2.83 | 10 |
| | Move (BIT) | 80 | 12 | 60 | 10 | 10.76 | 13 |
| | Move (WORD) | 46 | 12 | 32 | 10 | 2.82 | 10 |
| | Move (REAL) | 60 | 12 | 47 | 10 | 2.75 | 13 |
| | Block Move (INT) | 60 | 12 | 50 | 10 | – | 28 |
| | Block Move (WORD) | 60 | 12 | 50 | 10 | – | 28 |
| | Block Move (REAL) | 113 | 12 | 94 | 10 | – | 13 |
| | Block Clear | 100 | 12 | 83 | 10 | 4.63 | 11 |
| | Shift Register (BIT) | 130 | 12 | 94 | 10 | 0.45 | 16 |
| | Shift Register (WORD) | 120 | 12 | 100 | 10 | 2.76 | 16 |
| | COMM_REQ * | 175 | 175 | 120 | 120 | – | 13 |

* Commreq sent to HSC module.

## A-3.5 Sizes of Table Functions

The size of a function is the number of bytes consumed in user logic space for each instance of the function in a ladder diagram application program.

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| Table | **Array Move** | | | | | | |
| | INT | 110 | 12 | 90 | 10 | 5.50 | 22 |
| | DINT | 100 | 12 | 80 | 10 | 2.76 | 22 |
| | BIT | 129 | 12 | 92 | 10 | 1.08 | 22 |
| | BYTE | 109 | 12 | 80 | 10 | 4.75 | 22 |
| | WORD | 110 | 12 | 90 | 10 | 5.50 | 22 |
| | **Search Equal** | | | | | | |
| | INT | 90 | 12 | 70 | 10 | 6.59 | 19 |
| | DINT | 90 | 12 | 60 | 10 | 7.14 | 22 |
| | BYTE | 81 | 12 | 60 | 10 | 2.58 | 19 |
| | WORD | 90 | 12 | 70 | 10 | 6.59 | 19 |
| | **Search Not Equal** | | | | | | |
| | INT | 100 | 12 | 78 | 10 | 6.66 | 19 |
| | DINT | 110 | 12 | 81 | 10 | 7.14 | 22 |
| | BYTE | 74 | 12 | 57 | 10 | 2.56 | 19 |
| | WORD | 100 | 12 | 78 | 10 | 6.66 | 19 |
| | **Search Greater Than** | | | | | | |
| | INT | 100 | 12 | 80 | 10 | 6.69 | 19 |
| | DINT | 94 | 12 | 70 | 10 | 7.12 | 22 |
| | BYTE | 90 | 12 | 69 | 10 | 2.58 | 19 |
| | WORD | 100 | 12 | 76 | 10 | 6.69 | 19 |
| | **Search Greater Than/Equal** | | | | | | |
| | INT | 90 | 12 | 70 | 10 | 6.79 | 19 |
| | DINT | 90 | 12 | 60 | 10 | 7.15 | 22 |
| | BYTE | 81 | 12 | 60 | 10 | 2.56 | 19 |
| | WORD | 90 | 12 | 70 | 10 | 6.79 | 19 |
| | **Search Less Than** | | | | | | |
| | INT | 80 | 12 | 60 | 10 | 6.59 | 19 |

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| | DINT | 110 | 12 | 80 | 10 | 7.13 | 22 |
| | BYTE | 73 | 12 | 56 | 10 | 2.58 | 19 |
| | WORD | 80 | 12 | 60 | 10 | 6.66 | 19 |
| | **Search Less Than/Equal** | | | | | | |
| | INT | 80 | 12 | 60 | 10 | 6.66 | 19 |
| | DINT | 90 | 12 | 60 | 10 | 7.13 | 22 |
| | BYTE | 72 | 12 | 54 | 10 | 2.59 | 19 |
| | WORD | 80 | 12 | 60 | 10 | 6.66 | 19 |

## A-3.6 Sizes of Conversion and Control Functions

The size of a function is the number of bytes consumed in user logic space for each instance of the function in a ladder diagram application program.

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| Conversion | Convert INT to REAL | 60 | 12 | 40 | 10 | – | 10 |
| | Convert REAL to INT | 683 | 12 | 455 | 10 | – | 13 |
| | Convert DINT to REAL | 60 | 12 | 40 | 10 | – | 13 |
| | Convert REAL to DINT | 673 | 12 | 451 | 10 | – | 13 |
| | Convert WORD to REAL | 60 | 12 | 40 | 10 | – | 10 |
| | Convert REAL to WORD | 642 | 12 | 429 | 10 | – | 13 |
| | Convert BCD to INT | 57 | 12 | 40 | 10 | – | 10 |
| | Convert INT to BCD | 167 | 12 | 120 | 10 | – | 10 |
| | Convert BCD to REAL | 70 | 12 | 50 | 10 | | 10 |
| | Truncate to INT | 188 | 12 | 130 | 10 | – | 13 |
| | Truncate to DINT | 179 | 12 | 128 | 10 | – | 13 |
| Control | Call a Subroutine | 60 | 12 | 40 | 10 | – | 7 |
| | Do I/O * | 130 | 12 | 130 | 10 | – | 13 |

| Group | Function | CPU001/002 | | CPU005/E05 | | Increment | Size |
|---|---|---|---|---|---|---|---|
| | | Enabled | Disabled | Enabled | Disabled | | |
| | PID – ISA Algorithm | 231 | 85 | 150 | 57 | – | 16 |
| | PID – IND Algorithm | 231 | 85 | 150 | 57 | – | 16 |
| | Service Request | | | | | | |
| | #6 | 77 | 12 | 60 | 10 | – | 10 |
| | #7 (Read) | 221 | 12 | 173 | 10 | – | 10 |
| | #7 (Set) | 2610 | 12 | 2211 | 10 | – | 10 |
| | #14 ** | 169 | 12 | 139 | 10 | – | 10 |
| | #15 | 100 | 12 | 72 | 10 | – | 10 |
| | #16 | 110 | 12 | 80 | 10 | – | 10 |
| | #18 | 346 | 12 | 251 | 10 | – | 10 |
| | #23 | 377 | 12 | 361 | 10 | – | 10 |
| | #26//30 *** | 912 | 12 | 912 | 10 | – | 10 |
| | #29 | 72 | 12 | 60 | 10 | – | 10 |
| | Nested MCR/ENDMCR Combined | 31 | 33 | 31 | 33 | – | 4 |
| | Drum Sequencer | 267 | 222 | 184 | 152 | – | 34 |

* DO I/O timing is the time to output values to discrete output module.

** Service Request #14 (Clear Fault Table) timing was done when fault table contained no faults.

*** Service Request #26/30 (Interrogate I/O) timing was done when I/O configuration was empty and both an MDL740 (16pt out) and MDL640 (16pt in) were physically present.

# A-4 I/O Module Scan Times

The tables that follow show typical scan times for modules in a VersaMax PLC. Each module was configured with its default settings and user power was applied when applicable.

Four tables are included:

- Modules Located in Main Rack
- Modules Located in Local Single Rack
- Modules Located in Multiple Remote Rack
- Modules Located in Isolated Rack

## A-4.1 Reference to Discrete Module Types in the Scan Time Tables

In the scan time tables, discrete modules are grouped by type

| **Module Type** | **Module Catalog Number, IC200:** | | | | | |
|---|---|---|---|---|---|---|
| Discrete Input Type 1 | MDL140<br>MDL640<br>MDD846 | MDL141<br>MDL643<br>MDD847 | MDL143<br>MDD842<br>MDD848 | MDL144<br>MDD843<br>MDD849 | MDL631<br>MDD844<br>MDD850 | MDL635<br>MDD845<br>MDL930 |
| Discrete Input Type 2 | MDL240<br>MDL644 | MDL241<br>MDL650 | MDL243<br>MDD840 | MDL244 | MDL632 | MDL636 |
| Discrete Output Type 1 | MDL329<br>MDD843<br>MDD849 | MDL331<br>MDD844<br>MDD850 | MDL740<br>MDD845 | MDL741<br>MDD846 | MDL743<br>MDD847 | MDD842<br>MDD848 |
| Discrete Output Type 2 | MDL330 | MDL742 | MD744 | MDL750 | MDL840 | MDL940 |
| Discrete Output w/ ESCP Per Point Fault Reporting | MDL730 | | | | | |

For additional information on VersaMax I/O Modules, please refer to VersaMax Modules, Power Supplies, and Carrier User's Manual GFK-1504.

## A-4.2 Modules Located in Main PLC Rack

| Module Type | CPU005/CPUE05 Main Rack Input | CPU005/CPUE05 Main Rack Output | CPU001/CPU002 Main Rack Input | CPU001/CPU002 Main Rack Output |
|---|---|---|---|---|
| Discrete Input Type 1 * | 95 | --- | 158 | --- |
| Discrete Input Type 2 * | 117 | --- | 189 | --- |
| Discrete Output Type 1 * | --- | 84 | --- | 132 |
| Discrete Output Type 2 * | --- | 101 | --- | 152 |
| Discrete Output w/ ESCP Per Point Fault Reporting | --- | 116 | --- | 190 |
| Intelligent Discrete Input 20 Points | 349 | --- | 389 | --- |
| Intelligent Discrete Output 12 Points | --- | 294 | --- | 369 |
| Analog Input 4 Channels | 160 | --- | 190 | --- |
| Analog Input 8 Channels | 239 | --- | 312 | --- |
| Analog Input 15 Channels | 377 | --- | 526 | --- |
| Analog Output 2 Channels | --- | 109 | --- | 161 |
| Analog Output 4 Channels | --- | 145 | --- | 202 |
| Analog Output 8 Channels | --- | 217 | --- | 285 |
| Analog Output 12 Channels | --- | 289 | --- | 367 |
| Intelligent Analog Input 4 Channels | 237 | --- | 281 | --- |
| Intelligent Analog Input 7 Channels | 261 | --- | 305 | --- |
| Intelligent Analog Input 8 Channels | 272 | --- | 313 | --- |
| Intelligent Analog Output 4 Channels | --- | 212 | --- | 264 |
| PLC Network Comm Profibus-DP Slave | ** | ** | ** | ** |
| DeviceNet Network Master/Slave | ** | ** | ** | ** |

* Mixed modules have both and input and output scan time values.

** Network Communications Modules (NCM) Scan Impact Times vary depending upon the network configuration.

## A-4.3 Modules Located in Single-ended Expansion Rack

The table below shows timing for modules located in a single-ended expansion rack with a non-isolated Expansion Receiver module ((C200ERM002). This type of system does NOT have an Expansion Transmitter module (IC200ETM001) in the main rack.

| | CPU005/CPUE05 | | CPU001/CPU002 | |
|---|---|---|---|---|
| | Local Single Rack | | Local Single Rack | |
| **Module Type** | **Input** | **Output** | **Input** | **Output** |
| Discrete Input Type 1 * | 127 | --- | 191 | --- |
| Discrete Input Type 2 * | 179 | --- | 262 | --- |
| Discrete Output Type 1 * | --- | 116 | --- | 167 |
| Discrete Output Type 2 * | --- | 167 | --- | 222 |
| Discrete Output w/ ESCP Per Point Fault Reporting | --- | 176 | --- | 260 |
| Intelligent Discrete Input 20 Points | 643 | --- | 763 | --- |
| Intelligent Discrete Output 12 Points | --- | 714 | --- | 756 |
| Analog Input 4 Channels | 317 | --- | 389 | --- |
| Analog Input 8 Channels | 527 | --- | 631 | --- |
| Analog Input 15 Channels | 896 | --- | 1054 | --- |
| Analog Output 2 Channels | --- | 204 | --- | 266 |
| Analog Output 4 Channels | --- | 296 | --- | 374 |
| Analog Output 8 Channels | --- | 480 | --- | 592 |
| Analog Output 12 Channels | --- | 664 | --- | 809 |
| Intelligent Analog Input 4 Channels | 438 | --- | 533 | --- |
| Intelligent Analog Input 7 Channels | 479 | --- | 580 | --- |
| Intelligent Analog Input 8 Channels | 493 | --- | 596 | --- |
| Intelligent Analog Output 4 Channels | --- | 484 | --- | 613 |
| PLC Network Comm Profibus-DP Slave | ** | ** | ** | ** |
| DeviceNet Network Master/Slave | ** | ** | ** | ** |

* Mixed modules have both and input and output scan time values.

** Network Communications Modules (NCM) Scan Impact Times vary depending upon the network configuration.

## A-4.4 Modules Located in Multiple Remote Expansion Rack

The table below shows timing for modules located in the expansion racks of a multiple-rack expansion system that uses only Isolated Expansion Receiver Modules (IC200ERM001). In this type of system, there is an Expansion Transmitter module (IC200ETM001) in the CPU rack.

| | CPU005/CPUE05 | | CPU001/CPU002 | |
|---|---|---|---|---|
| | Multiple Remote Rack | | Multiple Remote Rack | |
| **Module Type** | **Input** | **Output** | **Input** | **Output** |
| Discrete Input Type 1 * | 130 | --- | 193 | --- |
| Discrete Input Type 2 * | 181 | --- | 258 | --- |
| Discrete Output Type 1 * | --- | 118 | --- | 167 |
| Discrete Output Type 2 * | --- | 165 | --- | 223 |
| Discrete Output w/ ESCP Per Point Fault Reporting | --- | 177 | --- | 261 |
| Intelligent Discrete Input 20 Points | 651 | --- | 766 | --- |
| Intelligent Discrete Output 12 Points | --- | 728 | --- | 757 |
| Analog Input 4 Channels | 324 | --- | 393 | --- |
| Analog Input 8 Channels | 541 | --- | 646 | --- |
| Analog Input 15 Channels | 920 | --- | 1087 | --- |
| Analog Output 2 Channels | --- | 206 | --- | 267 |
| Analog Output 4 Channels | --- | 300 | --- | 377 |
| Analog Output 8 Channels | --- | 489 | --- | 596 |
| Analog Output 12 Channels | --- | 678 | --- | 815 |
| Intelligent Analog Input 4 Channels | 442 | --- | 535 | --- |
| Intelligent Analog Input 7 Channels | 484 | --- | 582 | --- |
| Intelligent Analog Input 8 Channels | 497 | --- | 598 | --- |
| Intelligent Analog Output 4 Channels | --- | 490 | --- | 615 |
| PLC Network Comm Profibus-DP Slave | ** | ** | ** | ** |
| DeviceNet Network Master/Slave | ** | ** | ** | ** |

* Mixed modules have both and input and output scan time values.

** Network Communications Modules (NCM) Scan Impact Times vary depending upon the network configuration.

## A-4.5 Modules Located in Single-ended Isolated Expansion Rack

The table below shows timing for modules located in an expansion rack in a single-ended expansion system that has an Isolated Expansion Receiver Module (IC200ERM001) in the expansion rack and an Expansion Transmitter module (IC200ETM001) in the CPU rack.

| Module Type | CPU005/CPUE05 Isolated Rack | | CPU001/CPU002 Isolated Rack | |
|---|---|---|---|---|
| | Input | Output | Input | Output |
| Discrete Input Type 1 * | 466 | --- | 524 | --- |
| Discrete Input Type 2 * | 869 | --- | 913 | --- |
| Discrete Output Type 1 * | --- | 452 | --- | 496 |
| Discrete Output Type 2 * | --- | 837 | --- | 875 |
| Discrete Output w/ ESCP Per Point Fault Reporting | --- | 850 | --- | 914 |
| Intelligent Discrete Input 20 Points | 4050 | --- | 4086 | --- |
| Intelligent Discrete Output 12 Points | --- | 5135 | --- | 5135 |
| Analog Input 4 Channels | 2054 | --- | 2093 | --- |
| Analog Input 8 Channels | 3660 | --- | 3660 | --- |
| Analog Input 15 Channels | 6471 | --- | 6471 | --- |
| Analog Output 2 Channels | --- | 1221 | --- | 1251 |
| Analog Output 4 Channels | --- | 1991 | --- | 2021 |
| Analog Output 8 Channels | --- | 3531 | --- | 3560 |
| Analog Output 12 Channels | --- | 5071 | --- | 5099 |
| Intelligent Analog Input 4 Channels | 3155 | --- | 3196 | --- |
| Intelligent Analog Input 7 Channels | 3401 | --- | 3444 | --- |
| Intelligent Analog Input 8 Channels | 3483 | --- | 3526 | --- |
| Intelligent Analog Output 4 Channels | --- | 2751 | --- | 2811 |
| PLC Network Comm Profibus-DP Slave | ** | ** | ** | ** |
| DeviceNet Network Master/Slave | ** | ** | ** | ** |

* Mixed modules have both and input and output scan time values.

** Network Communications Modules (NCM) Scan Impact Times vary depending upon the network configuration.

# A-5 Ethernet Global Data Sweep Impact

Depending on the relationship between the CPU sweep time and Ethernet Global Data (EGD) exchange's period, the exchange data may be transferred every sweep or periodically after some number of sweeps. Therefore, the sweep impact will vary based on the number of exchanges that are scheduled to be transferred during the sweep. However, at some point during the operation of the PLC, all of the exchanges will be scheduled to transfer data during the same sweep. Therefore, all exchanges must be considered when computing the worst case sweep impact.

The Ethernet Global Data (EGD) sweep impact has two parts, Consumption Scan and Production Scan:

EGD Sweep Impact = Consumption Scan Time + Production Scan Time

Where the Consumption and Production Scans consist of two parts, exchange overhead and byte transfer time:

Scan Time = Exchange Overhead + Byte Transfer Time

## A-5.1 Exchange Overhead

Exchange overhead includes the setup time for each exchange that will be transferred during the sweep. This overhead varies depending on whether the exchange is consumed or produced and if the time-stamp for the exchange originates from the PLC itself or from a remote Network Time Protocol (NTP) server. When computing the sweep impact, include overhead time for each exchange.

*NTP time synchronization feature is supported IC200CPUE05-HK and previous versions only.

| | Consumed Exchange | Produced Exchange |
|---|---|---|
| Exchange Overhead* | 80 | 110 (304**) |

* Times are in microseconds.

** Represents overhead if the exchange is time-stamped with the PLC clock instead of a remote NTP server.

## A-5.2 Byte Transfer Time

This is the time required to transfer data between the PLC CPU module and the Ethernet module. The byte transfer time is slightly greater if the PLC memory being written to could contain overrides due to additional overhead. The times shown in the following table represent the time to transfer one data byte.

| | Consumed Exchange | Produced Exchange |
|---|---|---|
| Byte Transfer Time* | 1 (3.6**) | 1 |

* Times are in microseconds.

** Represents transfer time if memory type supports overrides.

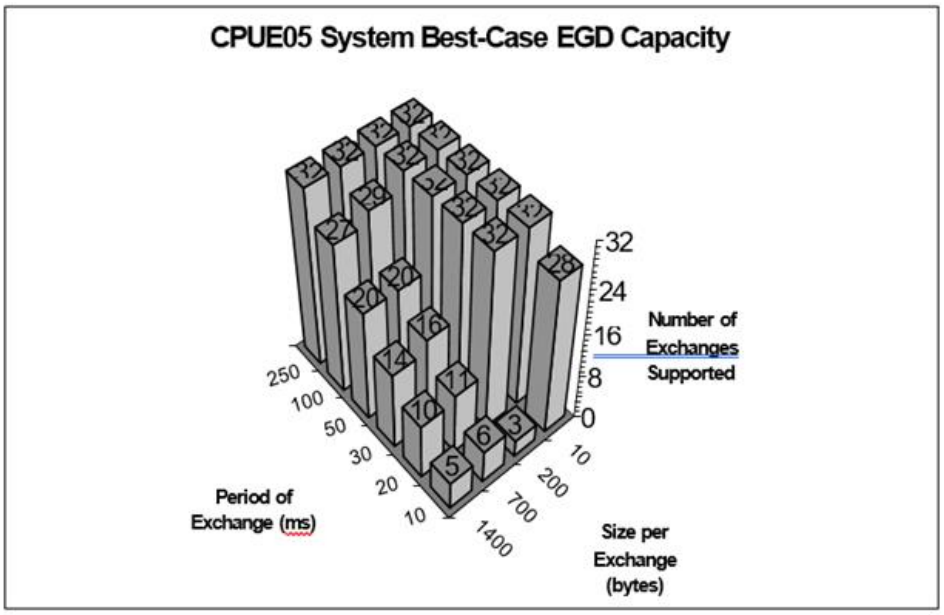
# A-6 Support for Large Ethernet Global Data Configurations

The VersaMax CPUE05 Ethernet Global Data (EGD) feature supports a configuration of up to 32 exchanges, at periods as short as 10 ms, with data sizes as large as 1400 bytes. However, the CPUE05 cannot support a configuration in which every aspect of EGD is maximized. The chart below indicates the maximum number of EGD exchanges that the CPUE05 can realistically support of a certain size and data refresh period under "Best- Case" conditions. These numbers will scale downwards based on the size of the user program, the presence of other Ethernet traffic, etc.

The term "Best-Case" indicates the following setup parameters apply:

- No user logic is present, so the logic sweep time is nearly 0
- There are no modules present in the system.
- No other Ethernet traffic present on the network.
- Assumed data refresh timeout is 2 × refresh period + 10ms

**Figure 228**



## Technical Support & Contact Information

Home link: http://www.Emerson.com/Industrial-Automation-Controls

Knowledge Base: https://www.emerson.com/Industrial-Automation-Controls/support

**Note:** If the product is purchased through an Authorized Channel Partner, please contact the seller directly for any support.

Emerson reserves the right to modify or improve the designs or specifications of the products mentioned in this manual at any time without notice. Emerson does not assume responsibility for the selection, use or maintenance of any product. Responsibility for proper selection, use and maintenance of any Emerson product remains solely with the purchaser.



EMERSON