CTI 2572
ETHERNET TCP/IP ADAPTER
PROGRAMMING REFERENCE MANUAL
Version 2.1

CTI Part # 062-00166

This manual is published by Control Technology Inc., 5734 Middlebrook Pike, Knoxville, TN 37921. This manual contains references to brand and product names which are tradenames, trademarks, and/or registered trademarks of Control Technology Inc. Siemens® and SIMATIC® are registered trademarks of Siemens AG. Other references to brand and product names are tradenames, trademarks, and/or registered trademarks of their respective holders.

DOCUMENT DISCLAIMER STATEMENT

Every effort has been made to ensure the accuracy of this document; however, errors do occasionally occur. CTI provides this document on an "as is" basis and assumes no responsibility for direct or consequential damages resulting from the use of this document. This document is provided without express or implied warranty of any kind, including but not limited to the warranties of merchantability or fitness for a particular purpose. This document and the products it references are subject to change without notice. If you have a comment or discover an error, please call us toll-free at 1-800-537-8398.

REVISION HISTORY

Version 1.0	9/9/94	Original Release
Version 1.1	12/9/94	Revised, added and corrected Error Code documentation Added ladder logic examples Expanded description of the PLC Command Interface Incorporated minor corrections and additions
Version 2.0	5/4/95	Removed PLC logic section (now in 2572 IOG) Added Packed task code description Incorporated minor corrections and additions
Version 2.1	3/10/98	Added Memory Exchange command. Documented additional address classes. Deleted “Raw” NITP and “Embedded Task Code” descriptions. Changed Chapter 3 title from “Troubleshooting” to “Application Development” and included sections on performance and TCP/IP coding.

PREFACE

This Reference Manual is intended for individuals who wish to develop computer system applications which interface to the CTI 2572 Ethernet TCP/IP Adapter.

Chapter 1 provides a summary of the module features and an overview of the software development requirements. Chapter 2 describes the message formats used by the module and provides coding examples.

For specific module hardware information, including module installation and checkout, please refer to the *CTI 2572 Ethernet TCP/IP Adapter Installation and Operation Guide (2572 IOG)*, CTI Part Number 62-146. If you will be using task codes, you will need to reference an applicable Siemens® publication, such as the *SIMATIC® 575 Task Code User Manual (*Order # *PPX:5575-8104-1).*

We assume you are familiar with the installation and operation of SIMATIC® 505 programmable controllers. Please refer to the appropriate SIMATIC® user documentation for specific information on SIMATIC® 505 programmable controllers and I/O modules.

We also assume that you are familiar with TCP/IP concepts and programming conventions.

Note:
On June 1, 1996, Siemens® Energy and Automation Inc. was granted exclusive rights to market the CTI 2572 product as the Siemens® SIMATIC® 505-CP2572. The contents of this manual fully apply to the 505-CP2572.

USAGE CONVENTIONS

NOTE:
Notes alert the user to special features or procedures

CAUTION:
Cautions alert the user to procedures which could damage equipment.

WARNINGS:
Warnings alert the user to procedures which could damage equipment and endanger the user.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1. 2572 OVERVIEW

1.1. Hardware Overview

The 2572 Ethernet TCP/IP Adapter is a single wide I/O module for SIMATIC® 505 controllers. The 2572 provides connectivity to Ethernet local area networks and allows the PLC to communicate with other network nodes using the Transmission Control Protocol/ Internet Protocol (TCP/IP) suite. Using the 2572, other devices on the network can acquire data from the PLC, send data and programs to the PLC, and exercise supervisory control over the PLC operation. In addition, the PLC can use the facilities of the 2572 to send messages to another node on the network.

The 2572 attaches to all Ethernet media specified by IEEE 802.3 including 10Base5 ("thick" coaxial cable), 10Base2 ("thin" coaxial cable), 10BaseT (unshielded twisted pair cabling), FOIRL (fiber optic cable) and 10BaseFL (fiber optic cable). 10BaseT cabling can be attached directly to the 2572 via an RJ-45 connector. Other IEEE 802.3 media may be connected to the AUI (Attachment Unit Interface) port via a user supplied transceiver.

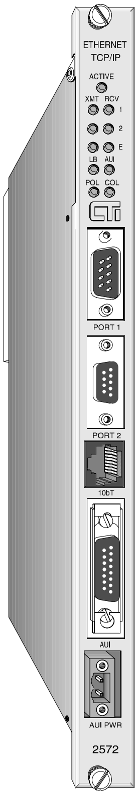


Figure 1. CTI 2572

The 2572 also provides two serial ports that can be used as programming ports for the local PLC or another PLC on the network. In addition, these ports can be used for module configuration operations.

The 2572 module itself requires no customer programming. PLC logic can be used to set module configuration and to control the operation of the module. Optionally, all configuration options can be set by module switches and a serially attached personal computer.

Please refer to the *2572 Installation and Operation Guide* for additional details regarding the hardware.

1.2. Functional Overview

Modern network communication is based on the concept of a client-server relationship. The *client* is a program on one network node that seeks a service, such as reading PLC data, from another network node. The *server* is the responding program which can provide the service.

The 2572 can operate as both a PLC server and a PLC client. As a PLC server, the 2572 *responds* to messages sent by another network node. As a PLC client, the 2572 *initiates* messages on command from the PLC.

PLC Server Function

The 2572 functions as a server to clients who wish to access the PLC. The following figure illustrates the typical message dialog between the client, the 2572, and the PLC.

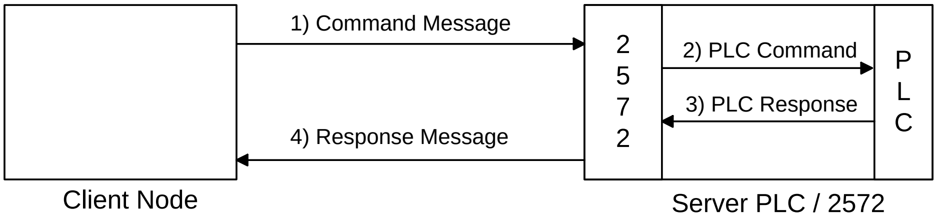


Figure 2. PLC Server Function

1) The client node sends a command message to the 2572 via TCP/IP. For example, the client may request that the 2572 read and return 25 words of V memory.

2) Based on the contents of a command message, the 2572 sends commands and data to the PLC processor via the backplane. For example, the 2572 would issue the applicable command to the PLC to retrieve 25 words of V memory.

3) The PLC processor responds to the command via the backplane. In the example, the PLC would return 25 V memory words.

4) After the PLC responds, the 2572 builds the appropriate message and returns it to the client node. In this example, the 2572 would build a network message containing the 25 words of data and send it to the client that requested it.

Messages between the 2572 and the client node are encapsulated in the TCP/IP protocol. The client creates the 2572 command message and sends it to the server using UDP or TCP. The response from the server is returned using the same delivery protocol. The client node may be a suitably programmed computer or another 2572 on the network (see next section).

The 2572 will support multiple concurrent server sessions. To operate the CTI 2572 as a PLC server, no PLC logic changes are required. However, you may choose to use PLC logic to set the network parameters for the module.

PLC Client Function

The 2572 can also function as a *PLC Client.* As a PLC Client, the 2572 acts as an agent for the PLC; it sends messages to other nodes and processes the responses under control of the PLC logic. Data in the PLC program specifies the recipient and data contents of the message. PLC logic sets a “trigger” bit to cause the 2572 to send the message.

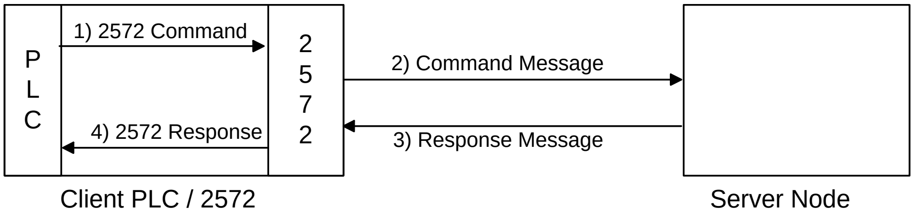


Figure 3. PLC Client Function

1) When the PLC detects a specified event, it sends a command to the local 2572. For example, the command could be to read 5 words from another node on the network.

2) Based on the command, the 2572 sends the applicable command via TCP/IP to the specified network (server) node.

3) The server node processes the command and returns a response via TCP/IP. In the example, the server node would return a message containing the specified words.

4) The 2572 processes the network message and notifies the PLC that the operation is complete. In the example, the 2572 would place the words in a specified PLC memory location and signal completion of the task.

The 2572 can support multiple concurrent client sessions. The server node shown in the illustration could be another 2572 or a computer programmed to emulate a 2572 PLC server.

The 2572 can support multiple server sessions and multiple client sessions concurrently. Therefore, networked PLCs can use the facilities of the 2572 to participate in multi-session peer-to-peer communications.

1.3. TCP/IP Overview

The 2572 uses TCP/IP (Transmission Control Protocol/Internet Protocol) to transport messages between the module and other nodes on the network. TCP/IP provides routing and delivery services for messages between application programs running on different processors (called hosts in TCP/IP terminology). You may select between connectionless (packet based) or connection-oriented (stream based) delivery services.

Connectionless Delivery

Connectionless delivery services allow you to send a message to another node without previously establishing a logical connection to the other node. TCP/IP provides a format known as the User Datagram Protocol (UDP) for sending and receiving connectionless messages. Connectionless delivery is simple to implement and consumes a small amount of system resources. However, delivery of UDP messages is not confirmed by the network protocol. It is up to cooperating application programs to acknowledge receipt. The application message protocols used with the 2572 will acknowledge receipt of a command message.

Connection-Oriented Delivery

With connection-oriented services, you must first establish a logical connection (known as a *virtual circuit*) before network nodes can exchange messages. TCP/IP uses the Transmission Control Protocol (TCP) format to implement connection-oriented services. TCP provides guaranteed delivery and message flow control. TCP is stream oriented, meaning the application program sees a properly sequenced stream of data rather than individual packets. TCP is often used for file transfer applications such as program downloads. You may also choose to use TCP when you want to ensure that the other node is available before you send a message.

Socket Interface

TCP/IP uses a standard structure known as a *socket,* for the application program interface. The de facto socket standard is the *Berkeley Socket,* named for the University of California at Berkeley, who originally distributed TCP/IP. Originally, the Berkeley Sockets were used with only the Unix operating system. Today, software which implements the Berkeley Socket standard is available for MS-DOS, IBM OS/2, and Microsoft Windows. Microsoft, in conjunction with several TCP/IP software providers, has established the Winsock standard to promote interoperability among TCP/IP software using Windows.

Summary

TCP/IP has the largest market share of any network protocol. Because it is based on open standards and has proven to be practical and reliable, the use of TCP/IP is growing dramatically. TCP/IP can be implemented with most PC, network, and minicomputer operating systems. Originally used by defense contractors and government facilities, TCP/IP is making major inroads into other segments of the industrial market. One important application of TCP/IP is communication among factory controller systems and supervisory workstations. For more information on TCP/IP features and services please refer to one of the TCP/IP publications commonly available at bookstores. An excellent reference is *Internetworking with TCP/IP* by Douglas Comer (1991, Prentice Hall).

1.4. Programming Overview

PLC Logic

If you are using the 2572 as a PLC server only, you are not required to provide any external PLC logic other than those required to set the network address parameters. The standard 2572 PLC Network Server function will reply to commands sent over TCP/IP from a client node.

If you are using the PLC Client function of the 2572, you will need to develop PLC logic to control the operation of the module. For example, if you want the 2572 to send a message when a particular event occurs, you will need to add the PLC logic to trigger the appropriate module command. The PLC logic for client Operations is described in the *2572 Installation and Operation Guide*.

Computer TCP/IP Support

The 2572 uses the TCP/IP protocol to transport data across the network. If you are programming on a UNIX machine, TCP/IP is a component of the operating system. Microsoft Windows and IBM OS/2 also provide TCP/IP drivers. If you are writing a DOS application, you will need to obtain this software from third party sources.

Microsoft has established a standard specification known as *Winsock*, which provides a common API for all Windows applications. The Winsock specification may be obtained from Microsoft via the Microsoft Internet FTP site, Microsoft Download Services, or the applicable Microsoft Software Development kit. Microsoft TCP/IP protocol stacks are available for Windows for Workgroups, Windows 95, and Windows NT. Windows 95 and Windows NT include TCP/IP support with the operating system. For Windows for Workgroups, you must obtain stack from Microsoft. The TCP/IP-32 stack may be downloaded from Microsoft Download Services or from their Internet FTP site.

Due to the increasing popularity of TCP/IP, there are a number of development tools which make it easy to send and receive messages via TCP/IP. In particular, several vendors offer “custom controls” for TCP/IP. Custom Controls are objects which may be used with Microsoft Visual BASIC and Visual C++. By manipulating the properties of the custom control, a developer can cause the object to perform TCP/IP operations including connect, send, receive and disconnect. Since the custom control makes the appropriate socket calls, the programmer is relieved of this task.

Application Logic

If you want to use a computer system as a client node (using the PC to access the PLC), your software must create the 2572 command message and process the response returned by the 2572. If you wish to process *unsolicited* messages from the 2572 (where the 2572 is a client), you will need to provide software which emulates the 2572 PLC server function. The 2572 message protocol is described in Chapter 2 of this manual.

CHAPTER 2. 2572 MESSAGE PROTOCOL

2.1. Overview

The 2572 uses TCP/IP as a means to transport messages over the network. Messages which contain 2572 commands and responses are encapsulated in the TCP/IP protocol. CTI uses a protocol known as CAMP (Common ASCII Message Protocol) to send the commands and to process the responses.

CAMP is used with all CTI communications products, including the CTI 2572. CAMP can be used over serial data links as well as over Ethernet networks. It has been designed to provide a sufficiently robust protocol while keeping the programming requirements simple. CAMP allows you to transfer large blocks of memory as well as to send vendor specific commands such as Siemens® SIMATIC® 505 Task codes.

CAMP is a command/response protocol. When a CAMP command is sent to another node, a response is expected. This dialog allows the application to determine whether a command was successfully completed. By providing delivery confirmation, CAMP provides reliability to connectionless (UDP) delivery.

2.2. CAMP Protocol Description

Message Format

CAMP messages use ASCII character format. The figure below illustrates the message format:

- The message begins with an ASCII left bracket **[** (0x5B),
- The type field identifies the type of data contained in the message data area,
- The error character is used to flag an error in the protocol or message data,
- The Message ID character is used to add an identifier to the message,
- The message data area contains command or response messages in a format indicated by the type field,
- The block check character (BCC) field is a checksum on the message,
- The message ends with an ASCII right bracket **]** (0x5D).

Message Start Delimiter	Message Type	Error Character	Message ID	Data	Block Check Characters	Message End Delimiter

Figure 4. CAMP Message Format

Numeric Data Representation

All numeric data within the CAMP packet is encoded Hexadecimal ASCII (Hex-ASCII). Hex-ASCII represents the hexadecimal equivalent of a byte as two ASCII characters. For example, if the hexadecimal equivalent of a byte is 0A, the message would contain the ASCII character 0 (0x30) followed by the ASCII character A (0x41). Valid Hex-ASCII characters are:

- ASCII 0 - 9 (0x30 - 0x39)
- ASCII A - F (0x41 through 0x46).

Representing numeric data as Hex-ASCII provides several benefits. First, it simplifies communications programming. Since the valid characters which define the data are limited, other characters not used for data can be chosen for control characters. Therefore, when you receive a character representing a message control, you can assume that it is *not* data. In contrast, a binary protocol must rely on certain byte sequences to represent control characters since the data byte can be any hex value. Second, Hex-ASCII provides a standard data representation independent of the way binary data is actually stored in memory. Therefore the programmer does not have to translate between PLC data format and PC data format. Since the message length is generally short and PLC scan times are relatively long, the lack of protocol density does not affect overall performance significantly.

Hex-ASCII Character Order

In Hex-ASCII, the character representing the high nibble of the byte is transferred first followed by the character representing the low nibble of the byte. CAMP uses 16 bit words in which the characters representing the most significant byte are transferred first, followed by the characters representing the least significant byte. Thus, values will appear in “natural” order on a protocol analyzer.

Message Identification

The CAMP protocol provides a single character message ID field which can be used by an application program to tag a message. By assigning a unique message ID to the message, the application can associate a response with a particular command. The allowable ID characters are ASCII 0-9 and A-F.

The 2572 Client function will cycle through the allowable characters when generating a command message and will reject replies which do not have a matching message ID. Programs which implement a server function are expected to behave like a 2572 server and echo the message ID in the response message.

Block Check Characters

The four block check characters (BCC) are the Hex-ASCII equivalent of a 16 bit word. The BCC immediately precedes the message terminating character. The calculations do not include the message delimiters ([or]).

To derive the BCC:

1. Convert any binary data to Hex-ASCII characters,
2. Add the lower 7 bits of each ASCII character to a 16 bit unsigned accumulator, ignoring carries,
3. Convert the 16 bit accumulator to 4 digit Hex-ASCII BCC field.

For example, assume you want to send a message containing the ASCII character F (0x46) followed by the ASCII character 3 (0x33). The result of performing a binary add on the characters is 0x79 (0x46 + 0x33). Thus, the BCC field contains the ASCII characters 0079 (0x30 30 37 39). See more examples later in this chapter.

To check the BCC:

1. Add the lower 7 bits of each character to an unsigned16 bit accumulator, ignoring carries,
2. Convert the BCC to binary and compare to the above or convert the accumulator to HEX ASCII and compare with the BCC.

Command / Response Design

CAMP is a command /response protocol. When you send a command message, the recipient should reply. This provides positive confirmation that the message was received. The TYPE field indicates whether the message is a command or a response.

Software performing a server function should always reply to a command message indicating successful completion or that an error condition was encountered. Software performing a client function should respond to a command message with an error reply, since a client cannot act on a command. Software (server or client) should never reply to a response message.

The command/response protocol should be observed by your application regardless of the underlying TCP/IP protocol chosen. Even when using TCP, the network protocol guarantees only that the application will be notified if TCP cannot deliver the message to the 2572 after several retries. It does not guarantee the PLC is able to process the encapsulated command which the 2572 delivers to the PLC. The CAMP response will provide notification of the PLC command processing.

Message Types

The CAMP protocol supports both device-independent message formats and vendor specific formats. The 2572 provides CAMP support for the memory transfer format (device independent) and SIMATIC® 505 Task Code (vendor specific). The module also supports the CAMP general error message type. These are described in the following sections.

2.3. Memory Transfer Messages

Memory transfer messages are used to send word-oriented data between processors. The 2572 maps these words to PLC memory. A significant benefit of this type is that it allows a large number of words to be sent in a single message packet.

Read Data Command

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE	2	ASCII 04 (Read Data Command)
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	ASCII 0-9, A-F
ADDRESS CLASS	4	ASCII 0000
ADDRESS WORD	4	Hex ASCII representing a 16 bit address
WORD COUNT	2	ASCII 01
DATA WORD 1	4	Hex ASCII - Number of words to be read (1-256)
BCC	4	Hex ASCII representing 16 bits of checksum data
END OF MESSAGE	1	ASCII] (0x5D)

Type Field: ASCII 04 is a command to Read Data from the remote device.

Error Character: Commands always have the error character set to 0.

Message ID: This character is set by the program which initiates the command. The response will echo the message ID. The message ID must be a valid hex-ASCII character (0-9, A-F).

Address Class: This field specifies the particular type of memory to be accessed. A value of 0000 indicates the default memory type of PLC V memory. See Appendix B regarding the use of other Address Class values with the 2572.

Address: This is the beginning address of the memory to be read or written. For Address Class 0000, this is the V memory number. For example, a value of 100 represents V100.

Word Count: This is a count of the number of data words contained in the message.

Data Word 1: This is the number of data words to be read from the remote device. Maximum = 256 words.

Read Data Response

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE	2	ASCII 05 (Read Data Response)
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	Echoes the Command Message ID
ADDRESS CLASS	4	Echoes the Address Class in the command message
ADDRESS	4	Echoes the Address in the command message
WORD COUNT	2	Hex ASCII indicating the number of data words read
DATA WORD 1	4	Hex ASCII representing a 16 bit data word
DATA WORD 2	4	Hex ASCII representing a 16 bit data word
..	..	..
..	..	..
DATA WORD n	4	Hex ASCII representing a 16 data word
BCC	4	Hex ASCII representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (0x5D)

Type Field: ASCII 05 is a response to a Read Data command.

Error Character: Valid responses will contain ASCII 0 in this field. If an error is detected, this field will contain an ASCII F and the data area will contain error information. See page 23 for the error message format.

Message ID: This field echoes the Message ID sent in the command message.

Address Class: This field echoes the Address Class value sent in the command message.

Address: This field echoes the Address value sent in the command message.

Word Count: This field is a count of the number of data words contained in the message.

NOTE:
For 1 - 255 words, the word count will be set to the number of data words following the word count. A word count 00 represents 256 words.

Data Words 1-n These data words are the Hex-ASCII equivalent of the numeric data. Up to 256 words can be included in one CAMP message.

Write Data Command

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE	2	ASCII 06 (Write Data Command)
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	ASCII 0-9 A-F
ADDRESS CLASS	4	ASCII 0000
ADDRESS.	4	Hex ASCII representing a 16 bit memory address
WORD COUNT	2	Hex ASCII indicating the number of data words
DATA WORD 1	4	Hex ASCII representing 16 bits
DATA WORD 2	4	Hex ASCII representing a 16 bit data word
..	..	..
..	..	..
DATA WORD n	4	Hex ASCII representing a 16 bit data word
BCC	4	Hex ASCII representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (0x5D)

Type Field: ASCII 06 is a command to Read Data from the remote device.

Error Character: Commands always have the error character set to 0.

Message ID: This character is set by the program which initiates the command. The response will echo the message ID.

Address Class: This field specifies the particular type of memory to be accessed. A value of 0000 indicates the default memory type of PLC V memory. See Appendix B regarding the use of other Address Class values with the 2572.

Address: This field is the beginning address of the memory to be read or written. For Address Class 0000, this is the V memory number. For example, a value of 100 represents V100.

Word Count: This field is a count of the number of data words contained in the message.

NOTE:
For 1 - 255 words, the word count will be set to the number of data words following the word count. A word count 00 represents 256 words.

Data Words 1-n These words are the Hex-ASCII equivalent of the numeric data. Up to 256 words can be included in one CAMP message.

Write Data Response

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE Write Data Response	2	ASCII 07
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	Echoes the Command Message ID
ADDRESS CLASS	4	Echoes the Address Class in the command message
ADDRESS	4	Echoes the Address in the command message
WORD COUNT	2	ASCII 01
DATA WORD1	4	Hex ASCII - Number of words written
BCC	4	Hex ASCII representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (0x5D)

Type Field: ASCII 07 is a response to a Write Data command.

Error Character: Normal responses will contain ASCII 0 in this field. If an error is detected, this field will contain an ASCII F and the data area will contain error information. See page 23 for the error message format.

Message ID: This field echoes the Message ID sent in the command message.

Address Class: This field echoes the Address Class value sent in the command message.

Address: This field echoes the Address value sent in the command message.

Word Count: This field is set to 01.

Data Word 1: This word represents the number of words successfully written. If the write was not successful an error response will be returned.

Memory Exchange Command

The memory exchange format facilitates a high speed memory transfer between two devices. Essentially this is a combination of a CAMP memory write and a CAMP memory read. The command contains the data to be written to the remote device. The response contains the data read from the remote device.

Description	**Lengt h**	**Value**
START OF MESSAGE	1	ASCII [(hex 5B)
TYPE	2	ASCII 08 (Memory Exchange Command)
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	ASCII 0-9 A-F
ADDRESS Class	4	ASCII 0000 = Default Class
ADDRESS	4	Hex ASCII representing a 16 bit memory address to which data will be written
WORD COUNT	2	Hex ASCII indicating the number of data words in the message
DATA WORD 1	4	HexASCII Address from which data will be read
DATA WORD 2	4	Hex ASCII Number of words to be read
DATA WORD 3	4	HexASCII representing 1st word of PLC data to be written
..		
DATA WORD n	4	Hex ASCII representing last word of PLC data to be written
BCC	4	Hex ASCII representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (hex 5D)

Type Field: ASCII 08 is a command to Exchange Data from the remote device.

Error Character: Commands always have the error character set to 0.

Message ID: This character is set by the program which initiates the command. The response will echo the message ID.

Address Class: This field specifies the particular type of memory to be accessed. A value of 0000 indicates the default memory type of PLC V memory.

NOTE:
When using the Exchange Data command, the Address Class should always be set to 0000.

Address Write: This field is the beginning address of the memory to be written. For the SIMATIC® Series 505 PLCs, this is the V memory number. For example a value of 100 represents V100.

Word Count: This field is a count of the number of words contained in the data portion of the message, including the two words used to specify the read address and number of words to be read. This is equivalent to the number of PLC data words to be written plus 2. For example, if you were writing 6 words to the PLC, the Word Count would be 8.

NOTE:
For 1 - 255 words, the word count will be set to the number of data words following the word count. A word count 00 represents 256 words.

Data Word 1 Indicates the starting address from which data will be read.

Data Word 2 Indicates the number of data words to be read.

Data Word 3 Contains the first data word to be written. These words are the Hex-ASCII equivalent of the numeric data.

Data Words 4 - n Contain additional data words to be written. A maximum of 254 words can be written using this message format.

Memory Exchange Response

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE	2	ASCII 09 (Memory Exchange Response)
ERROR CHARACTER	1	ASCII 0
MESSAGE ID	1	Echoes the Command Message ID
ADDRESS CLASS	4	Echoes the Address Class in the command message
ADDRESS	4	Echoes the Address in the command message
WORD COUNT	2	Hex ASCII indicating the number of data words read
DATA WORD 1	4	Hex ASCII representing a 16 bit data word
DATA WORD 2	4	Hex ASCII representing a 16 bit data word
..	..	..
..	..	..
DATA WORD n	4	Hex ASCII representing a 16 data word
BCC	4	Hex ASCII representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (0x5D)

Type Field: ASCII 09 is a response to a Memory Exchange command.

Error Character: Valid responses will contain ASCII 0 in this field. If an error is detected, this field will contain an ASCII F and the data area will contain error information. See page 23 for the error message format.

Message ID: This field echoes the Message ID sent in the command message.

Address Class: This field echoes the Address Class value sent in the command message.

Address: This field echoes the Address value sent in the command message.

Word Count: This field is a count of the number of data words contained in the message.

NOTE:
For 1 - 255 words, the word count will be set to the number of data words following the word count. A word count 00 represents 256 words.

Data Words 1-n These data words are the Hex-ASCII equivalent of the numeric data. Up to 256 words can be included in one CAMP message.

Command Error Response

If an error occurs when processing the contents of the data field (which could be caused by an invalid address, invalid number of words specified, or a PLC processing error), then a Command Error response will be returned. The format is as follows:

Description	**Length**	**Value**
START OF MESSAGE	1	ASCII [(0x5B)
TYPE	2	
Read Data Response		ASCII 05
Write Data Response		ASCII 07
Memory Exchange Resp.		ASCII 09
ERROR CHARACTER	1	ASCII F
MESSAGE ID	1	(Echoes the message ID sent in the command message)
ADDRESS CLASS	4	(Echoes the address class sent in the command message)
ADDRESS	4	(Echoes the address sent in the command message)
WORD COUNT	2	ASCII 04
MODULE ID	4	Hex ASCII
PROTOCOL MANAGER NO.	4	Hex ASCII
ERROR CODE	4	Hex ASCII (See Appendix A)
EXTENDED ERROR INFORMATION	4	ASCII
BCC	4	Hex ASCII Characters representing a 16 bit checksum
END OF MESSAGE	1	ASCII] (0x5D)

Type Character: ASCII 05 is a response to a Read Data command.
ASCII 07 is a response to a Write Data command.
ASCII 09 is a response to a Memory Exchange command.

Error Character: Always set to ASCII F for a data error

Message ID: The message ID field will contain the message ID sent in the command message.

Address Class: The address class field will contain the address class sent in the command message.

Address: The address field will contain the address sent in the command message.

Word Count: The word count is set to ASCII 04 to indicate that there are 4 error data words.

Module ID: The Module ID is provided by CTI modules to help in problem diagnosis. This field for user information only; the value in this field is ignored by CTI 2572 error processing logic. Your software can set this to any valid Hex-ASCII value when returning an error message.

Protocol Manager ID: The protocol manager ID is provided by CTI software to help in problem diagnosis. This field for user information only; the value in this field is ignored by CTI 2572 error processing logic. Your software can set this to any valid Hex-ASCII value when returning an error message.

Error Code: The error code describes the specific error detected. Appendix A describes the error codes in detail.

Extended Error Info: This word is used for extended error information by some error responses.

Read Data Command /Response Example

The command ADDRESS WORD contains the starting memory address in the remote device and the first DATA word contains the number of words to be read. The response echoes the ADDRESS WORD contained in the command, sets the WORD COUNT to the actual number of words read, and places the requested words in the CAMP message following the word count.

For example, assume you want to read 3 words beginning at V memory location 5 in the remote PLC. Assuming the message ID is set to 0, the read data command message would be:

	Msg Start	**Type**	**Err**	**ID**	**Class**	**Address**	**Count**	**Data**	**BCC**	**Msg End**
Char	[	04	0	0	0000	0005	01	0003	036D	]
Hex	5B	30 34	30	30	30 30 30 30	30 30 30 35	30 31	30 30 30 33	30 33 36 44	5D

Where the BCC is calculated as:
BCC = (30+34)+30+30+(30+30+30+30)+(30+30+30+35)+(30+31)+(30+30+30+33) = 036D

Assuming that the memory locations contain the decimal values 6, 12, and 21 (0x0006,000C, and 0015), the normal response message would be :

Msg Start	**Type**	**Err**	**ID**	**Class**	**Address**	**Count**	**Word1**	**Word2**	**Word3**	**BCC**	**Msg End**
[	05	0	0	0000	0005	03	0006	000C	0015	050C	]

If the 2572 were unable to read the V memory, it would return the following error response:

Msg Start	Type	Err	ID	Class	Addr	Count	Mod ID	PM No.	Error Code	Resv	BCC	Msg End
[	05	F	0	0000	0005	04	2572	0023	00AC	0000	05FD	]

The error code AC indicates a memory read error (See Appendix A).

Write Data Command / Response Example

The command ADDRESS WORD contains the starting address in the remote device and the WORD COUNT contains the number of words to be written. The word data is included in the data section of the message. The response echoes the MESSAGE ID, THE ADDRESS CLASS, and the ADDRESS.

Assuming the message ID is set to 1, the following example message writes 3 words of decimal value 12, 33, and 4 (0x0C,0x21, and 0x04) starting at V memory location V10.

Msg Start	Type	Err	ID	Class	Addr	Count	Word1	Word2	Word3	BCC	Msg End
[	06	0	1	0000	000A	03	000C	0021	0004	0515	]

The normal response from the 2572 should be:

Msg Start	Type	Err	ID	Class	Addr	Count	Word1	BCC	Msg End
[	07	0	1	0000	000A	01	0003	02B9	]

If the 2572 server were unable to write V memory, it would return an error message of:

Msg Start	Type	Err	ID	Class	Addr	Count	Mod ID	PM No.	Error Code	Rsvd	BCC	Msg End
[	07	F	1	0000	000A	04	2572	0023	00AD	0000	060D	]

The error code 0xAD indicates an error writing V memory (see Appendix A).

2.4. Packed Task Code Message

Siemens® SIMATIC® 505 PLCs support a command protocol known as *Task Codes*. Using task codes, an application can access most of the variable types within the PLC. For example, programming software such as TISOFT uses task codes to program the PLC. Similarly, Operator Interface panels use Task Codes to read and write to the PLC. See APPENDIX C. TASK CODE SUMMARY for additional information.

Typically, task codes are transmitted via serial communications using the Non Intelligent Terminal Protocol (NITP) or the Transparent Binary Protocol (TBP). The 2572 allows task codes in NITP message format to be transmitted via TCP/IP. In addition, the 2572 supports a much denser (and therefore higher performance) variation known as CAMP Packed Task Code.

The CAMP Packed Task Code message format allows *multiple* task code messages to be enclosed in a single CAMP message packet. The task code messages are formatted per the SIMATIC® TI500/505 NITP (Non-Intelligent Terminal Protocol) specifications, except that the beginning and ending delimiters are omitted. The responses to the packed task code messages will be placed in the CAMP response message in the same order as they were received. For example, the response to the second task code message in the CAMP command message will be returned in the second task code message in the CAMP response message.

NOTE:
Before placing the NITP message in a CAMP packet, you must strip off the NITP message delimiters (colon and semicolon).

Message Format

The message format is described below. You can cause the CAMP server to bypass NITP error checking for a task code message by setting the NITP checksum to 4 ASCII Question Marks (????). The CTI PLC Server function will always generate the NITP checksums in the response; you may choose to ignore it. You can include up to 14 task codes messages in one CAMP message.

Description	**Length**	**Value**
Start Of Message	1	ASCII [(0x5B)
Type	2	
Device Specific Command		ASCII 02
Device Specific Response		ASCII 03
Error Character	1	ASCII 0
Message ID	1	ASCII 0-9 A-F
Device Class	4	ASCII 0001
Reserved (Unused)	4	ASCII 0000
Reserved (Unused)	2	ASCII 00
Msg 1: NITP Character Count	2	Hex ASCII representing an 8 bit number
Msg 1: Task Code	2	Hex ASCII
Msg 1: Task Code Data	variable	Hex ASCII
Msg 1: NITP Err Check Char	4	Hex ASCII representing 16 data bits Set to ASCII ???? = 0x3F3F3F3F to ignore
Msg 2: NITP Character Count	2	Hex ASCII representing an 8 bit number
Msg 2: Task Code	2	Hex ASCII
Msg 2: Task Code Data	variable	Hex ASCII
Msg 2: NITP Err Check Char	4	Hex ASCII representing 16 data bits Set to ASCII ???? = 0x3F3F3F3F to ignore
Repeat up to 14 task code messages.		
Msg n: NITP Character Count	2	Hex ASCII representing an 8 bit number
Msg n: Task Code	2	Hex ASCII
Msg n: Task Code Data	variable	Hex ASCII
Msg n: NITP Err Check Char	4	Hex ASCII representing 16 data bits Set to ASCII ???? = 0x3F3F3F3F to ignore
BCC	4	Hex ASCII - representing a 16 bit

		checksum
End Of Message	1	ASCII] (0x5D)

Type : ASCII 02 indicates a Device Specific command. ASCII 03 indicates an Device Specific response.

Error Character: Always set to ASCII 0 for a command and a valid response.

Message ID: This character is set by the program which initiates the command. The response will echo the message ID.

Device Class: ASCII 0001 is used to indicate SIMATIC® 505 packed task code format.

Reserved: These fields are reserved for future use. They should be set to 0 in the command.

Character Count: The NITP character count is calculated exactly as specified in SIMATIC® documentation. Although you do not place the NITP starting and ending delimiters in the CAMP packet, they are still added to the NITP character count. This approach allows you to use the same character count and checksum as you would when sending standard NITP messages. Thus, the NITP character count includes the number of characters in the character count, the task code, the task code data, and the NITP checksum plus 2. The valid range for character count is 10-72 (0x0A - 0x48).

Task Code: The task code is a hexadecimal number represented by two ASCII characters. See the SIMATIC®/TI documentation for a complete description of the task codes.

Task Code Data: The task code data is dependent upon the task code, See the SIMATIC®/TI documentation.

NITP Err Check: This is a checksum on the NITP characters. See the SIMATIC®/TI documentation for details. If you do not wish to calculate the checksum, set the checksum characters to ASCII ???? (0x3F3F3F3F).

Message Processing Example

The following example reads one word from V memory location 3 and writes one word to V memory location 4. Note that the message address fields are 0002 and 0003, since NITP uses a 0 offset . The NITP Error Check Characters (ECC) were set to ???? (ASCII question marks) to avoid eliminate calculations and the Message ID was set to 0. The first NITP character count = 2 (count) +2 (task code)+4 (address) +4 (ECC) +2 = 14 (0x0E). The second NITP character count = 2 (count) + 2 (task code) + 4 (address) + 4 (data) + 4 (ECC) + 2 = 18 (0x12).

Msg Start	**Type**	**Err**	**Msg ID**	**Device Code**	**Rsvd**	**Rsvd**	**NITP Count**	**Task Code**	**Mem Addr1**	**NITP ECC**
[	02	0	0	0001	0000	00	0E	01	0002	????

NITP Count	**Task Code**	**Mem Addr1**	**Data**	**NITP ECC**	**BCC**	**Msg End**
12	02	0003	1900	????	0885	]

NOTE:

The memory addresses in NITP are 0 relative. Memory address 0 will retrieve the first V memory location.

Assuming V memory location 2 contains a value of 2000, the response would be:

Msg Start	**Type**	**Err**	**Msg ID**	**Device Code**	**Rsvd**	**Rsvd**	**NITP Count**	**Task Code**	**Mem Value**	**NITP ECC**
[	03	0	0	0001	0000	00	0E	01	07D0	EA2F

NITP Count	**Task Code**	**NITP ECC**	**BCC**	**Msg End**
0A	02	F5FE	072C	]

Command Error Response

Once the CAMP message has passed all CAMP protocol checks, is received without error, then the module will sequentially evaluate each task code message for a valid length and for the checksum data. If an error is detected during this evaluation, message processing will be suspended and the module will return a CAMP Command Error response. The format is as follows:

Description	**Length**	**Value**
Start Of Message	1	ASCII [(0x5B)
Type Device Specific Response	2	ASCII 03
Error Character	1	ASCII F
Message ID	1	(Echoes the message ID sent in the command message)
Device Class	4	(Echoes the device class sent in the command message)
Address (not used)	4	ASCII 0000
Word Count	2	ASCII 04
Module ID	4	ASCII Characters 0-9, A-F
Protocol Manager No.	4	Hex ASCII
Error Code	4	Hex ASCII (See Appendix A)
Task Code Message No.	4	Hex ASCII (0001 - 000E)
BCC	4	Hex ASCII Characters representing a 16 bit checksum
End Of Message	1	ASCII] (0x5D)

Type Character: ASCII 03 is a response to a Device Specific command.

Error Character: Always set to ASCII F for a data error

Message ID: The message ID field will contain the message ID sent in the command message.

Address Class: The device class field will echo the device class sent in the command message.

Address: The address field is not used in this response. It will be set to 0000.

Word Count: The word count is set to ASCII 04 to indicate that there are 4 error data words.

Module ID: The Module ID is provided by CTI modules to help in problem diagnosis.

Protocol Manager ID: The protocol manager ID is provided by CTI software to help in problem diagnosis.

Error Code: The error code describes the specific error detected. Appendix A describes the error codes in detail.

Task Code Message #: This word identifies which task code message contained the error. For example, if the error was located in the third task code message with the CAMP message, this word would contain the value 0003.

Once all task code messages are verified for correct checksum and length, then they are sent to the PLC in a group. Up to eight task codes can be processed in a single scan; the actual number depends upon the PLC setting for number of task codes per scan. Responses from the PLC will be placed in the CAMP response message in the same order as the corresponding command.

If the PLC processor returns a task code error, the error response will be placed in the CAMP response message in place of the normal response.

NOTE:
A type 00 task code response is not considered to be a CAMP error and will not cause the CAMP error character to be set. A task code 00 is an error response from the PLC processor, not from the 2572. If you use Packed Task Codes, you should check each task code response for a task code 00 and process the result accordingly. See Appendix A for a listing of the task code 00 error codes.

NOTE:
For best performance using this message format, you should set the number of task codes per scan on the PLC to the maximum available (typically 8). If you are using TISOFT, this can be set by using Aux Function 19.

2.5. CAMP Protocol Error Message

A CAMP Protocol Error message (Type FF) is returned when a CAMP protocol error is detected. CAMP protocol errors include missing delimiters, bad checksum, and invalid type. The format of the type FF error message is identical to other CAMP error messages. Since the content of the message packet is suspect when a CAMP protocol error is detected, the message ID (and other fields which are usually echoed) are all set to 0.

Description	Length	Value
START OF MESSAGE	1	ASCII [
TYPE CHARACTER	2	ASCII FF
ERROR CHARACTER	1	ASCII F
MESSAGE ID	1	ASCII 0
UNUSED	4	ASCII 0000
UNUSED	4	ASCII 0000
WORD COUNT	2	ASCII 04
MODULE ID	4	4 Hex-ASCII Characters
PROTOCOL MANAGER NO.	4	4 Hex-ASCII Characters
ERROR CODE	4	4 Hex-ASCII Characters (See Appendix A)
RESERVED	4	ASCII 0000
BCC	4	4 Hex-ASCII characters
END OF MESSAGE	1	ASCII]

The following is a CAMP error message where an invalid type code was detected.

Msg Start	Type	Err	Msg ID	Un-used	Un-used	Count	Mod ID	PM No.	Error Code	Rsvd	BCC	Msg End
[	FF	F	0	0000	0000	04	2572	0023	007B	0000	0614	]

2.6. CAMP Message Summary

The following table summarizes the CAMP message formats described in the previous sections. In the table, n represents any valid Hex-ASCII character (0-9, A-F) and cccc represents 4 block check characters.

Message	Msg Start	Type	Err	Msg ID	Class Dev Cl	Addr/	Word Count	Data Area	BCC	Msg End
Read Data Cmd	[	04	0	n	nnnn	nnnn	01	# words to read	cccc	]
Read Data Resp	[	05	0	Echo	Echo	Echo	nn	Data	cccc	]
Write Data Cmd	[	06	0	n	nnnn	nnnn	nn	Data to be written	cccc	]
Write Data Resp	[	07	0	Echo	Echo	Echo	00	none	cccc	]
Mem Exch Cmd	[	08	0	n	0000	nnnn	nn	Data to be written	cccc	]
Mem Exch Resp	[	09	0	Echo	Echo	Echo	nn	Data	cccc	]
Cmd Error Resp	[	**	F	Echo	Echo	Echo	04	Err Info	cccc	]
Packed T/C Cmd	[	02	0	n	0001	0000	nn	NITP Cmd	cccc	]
Packed T/C Resp	[	03	0	n	Echo	Echo	nn	NITP Resp	cccc	]
Protocol Error	[	FF	F	0	0000	0000	04	Err Info	cccc	]

Figure 5. CAMP Message Summary

** The TYPE field will contain the response code which corresponds to the command. For example, the Error response to a Read Data command have a value in the TYPE field of 05.

CHAPTER 3. APPLICATION DEVELOPMENT

This chapter is intended to provide you general guidelines for designing, coding, and debugging your application. You should refer to the *CTI 2572 Installation and Operation Guide* for module information.

3.1. Factors Affecting Performance

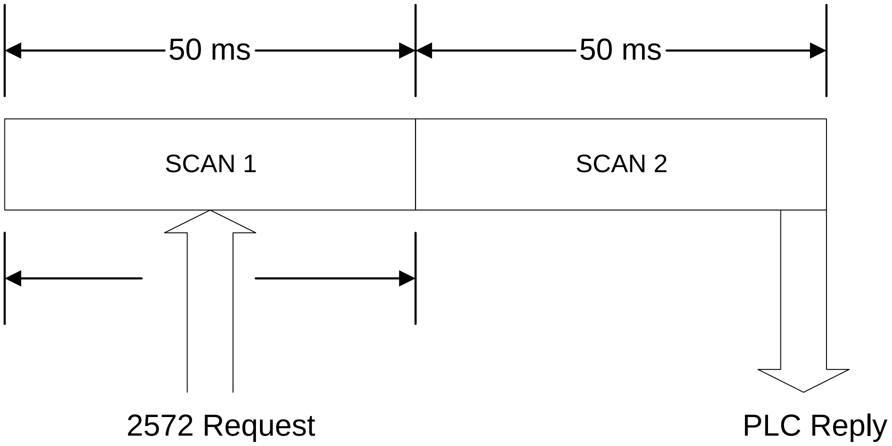
With most Man Machine Interface (MMI) applications, providing rapid screen update is an important consideration. Since communications with the PLC typically involves three different processing systems (host computer, 2572 module, and a SIMATIC® 505 PLC), precisely calculating performance is a complex, if not impossible, task. Even within the host computer, factors such as processor speed, task swapping load, operating system overhead, and efficiency of the polling software can influence performance. Similarly, the speed and configuration of the target PLC can make a huge difference. Although there is no easy formula to calculate performance, understanding the factors which influence performance can help you properly design your application.

PLC Factors

When accessing large amounts of data, the way data is organized in the PLC and the way in which you access the data can have a significant affect on the overall performance of the application. For example, it is more efficient to access data in a large block rather than multiple groups of small blocks. There are also some differences between PLC models.

PLC Interface Timing

The 2572 and the PLC exchange requests and data via a shared RAM interface on the module. Access to the shared RAM is determined by the PLC scan cycle. Requests from the 2572 must be loaded prior to the beginning of a PLC scan. Replies are written to the shared RAM by the PLC at the end of the PLC scan.



Since the timing of the access to the PLC is determined by the PLC scan rate; the longer the scan time, the less often PLC data can be accessed. The minimum time required to access the PLC memory is one scan. However, because a request could arrive just after the PLC scan has started, it could take two scans before a response is returned (see previous figure). On the average, you should estimate the access time as 1.5 times the scan interval. Therefore, if the scan time is 50 ms, then the access time would average 75ms.

PLC Data Access Size

The amount of data that can be requested at one time depends upon the PLC configuration and the method used to request data. The shared RAM interface provides 8 slots; each slot can hold one request to the PLC or reply from the PLC.

When the request is a task code, each task code occupies one slot. Thus, the shared RAM can accommodate a maximum of 8 task code requests at one time. However, the PLC may limit task codes per scan to less than 8. Some PLC's (e.g. SIMATIC® TI525) have a fixed task code per scan of 2. Most others, including the 545, 555, and TI575, default to 2 task codes per scan but allow the value to be set to a value up to eight.

When CAMP Memory Transfer commands are used to access the PLC, the 2572 uses a more efficient block transfer method known as Pseudo DMA (PDMA). Using PDMA, each slot will access up to four times as much data as a typical task code. The number of PDMA slots are unaffected by the task code per scan setting.

Block vs. Random Access

Generally, it is more efficient to access a contiguous block of data in the PLC than to access data randomly. Reading or writing contiguous data significantly increases the amount of data to be transferred in a single scan. For example, using a random read task code to access V memory could access as few as seven words of data per task code. Conversely, a CAMP Memory Read (which uses PDMA) could read as many as 60 words per SFIC slot.

TCP/IP Processing Overhead

Each TCP/IP message requires the resources of the PC and the 2572 to compose and decompose each message, to calculate and evaluate error checking fields, and to generate network acknowledgments (TCP only). The actual overhead will depend upon the speed of the PC, the protocol used (TCP typically requires an additional 10 ms for acknowledgment) and the condition of the network (noisy networks can require multiple retries). Typical TCP/IP overhead (query and response) is 30 - 40 ms per transaction.

Module Input Queue Depth

The 2572 allows multiple clients to communicate with it concurrently. However, access to the PLC is gated by the PLC interface described in previous sections. In situations where the requests arrive faster than the PLC can process them, the 2572 queues the requests. The response time to a request will vary with the queue depth and the processing rate of the PLC interface.

Module Message Processing Time

The module consumes about 1ms per task code unpacking the task code data from the PLC. Other module processing tasks may also consume several milliseconds at times. Some of the module processing time may be overlapped with the PLC and/or host computer operations.

3.2. Performance Enhancement Tips

Performance Tips - PLC

There are several actions you may be able to take to ensure that the PLC is optimized for performance.

- Group data in contiguous locations in PLC memory, which allows you to use block commands to access large groups of data. The PLC logic provides a means to copy data and also allows discrete values to be packed into V memory words. These commands add very little to the PLC scan time. ***This action, coupled with the use of CAMP memory transfer commands, offers the largest potential for performance enhancement.***
- If you are using task codes to access the PLC, ensure that the “task codes per scan” parameter is set to the maximum value.
- Avoid using fixed scan times, if possible. For most PLC models, a fixed scan time could cause task code processing to be deferred until a later scan or could result in unnecessary idle time.
- Install the module in a local base, if possible.

Performance Tips - Application

Supervisory control and monitoring applications usually read a large number of data elements continuously, while writing a few data points occasionally. With this transaction mix, several request messages may be required to obtain all of the data required by the application at each point in time.

A primary means of optimizing total throughput is to minimize the number of messages required to obtain the data by maximizing the amount of data requested in each message. This strategy minimizes the TCP/IP processing overhead and allows 2572 to access more data per scan, potentially reducing the number of scans required to obtain the data.

Data Blocking

One means to accomplish the above is to request large block of data rather than reading random points. If the requested data is already in a single block in the PLC, then this task is rather easy. However, if the data is spread out thorough PLC memory, your application may need to combine the data requests into a large block read. When the data is returned, your application would then pick out the actual data needed and ignore the rest.

The example below indicates how an application the needed V10-V20, V51-V68, and V74-V80 could obtain the data in one CAMP memory read of 81 words.

V10 - V20	V21 - V50	V51-V68	V69 - V73	V74 - V 80
(required)	(unused)	(required)	(unused)	(required)

CAMP Memory Transfer vs. Packed Task Code

For applications where the data can be retrieved by large block reads, using the CAMP memory transfer will result in the most efficient data transfer. Note that, in one CAMP read, you could return up to 256 words which could include 4096 discrete points.

However, since CAMP memory transfer allows only one data type per message, applications which read small blocks many different data types may benefit from using the Packed Task Code format. For example, to read 10 V memory values, 10 control relays, 30 WX values, and 2 loop process variables would require 4 messages (one for the V, one for the C, one for the WX, and one for the loop). Using the packed task code format (which lets you place up to 14 NITP task code requests in a single message), you could obtain the data in a single message.

If you need to write more than a few random values should use the Packed Task Code format. The CAMP memory transfer format allows only one memory range per message.

Multiple Command / Reply Sessions

Another way to improve overall performance is to establish more than one command reply session with the same 2572. Implemented properly, the overlapped processing that occurs between the PC, 2572, and PLC can enhance total throughput.

The best way to accomplish this task is to open a second socket to the module, treating each socket as a separate command / reply stream. Start by sending a request to each socket. When you receive a reply on one of the sockets, process the response and send the next request in the queue to that socket. Continue until all messages have been sent and replies have been processed. This technique has several advantages over other approaches:

- It continues to use the same command/response protocol as a single socket (including message ID sequence)
- Your application can use the socket source to match response messages to the corresponding command message rather than requiring complex logic to accomplish this.
- If your application throughput requirements did require a second 2572 in the same PLC rack, it would be easy to transition to this configuration by simply changing the IP address of the second socket.
- If you use UDP no additional resources are required at the module.

You should exercise care in how you use this technique. In particular:

- Opening more than two sockets to the 2572 will probably not result in significant additional performance improvement.
- Opening a TCP socket to the module will cause the module to allocate resources for the TCP socket. Each 2572 has a limited number of TCP sockets that can be allocated. See the 2572 Installation and Operations Guide.

- If multiple PCs, each using this technique, are accessing the same 2572 concurrently, overall system performance could actually decline, especially if the applications are requesting data faster than actually required.

3.3. Coding Illustrations

Once you understand how to build the command messages and to interpret the reply messages, the only issue remaining is how to send and receive these messages from the PLC.   Since the 2572 acts as a TCP/IP host, you will use standard TCP/IP structures and coding techniques communicate with the module.

You will find the interface to be reasonably simple. Your program communicates with another TCP/IP host via a structure known as a *socket*. When a socket is created, your code receives a “handle” (similar to a file handle) which you can use to reference the connection. You may create a socket for UDP (User Datagram Protocol) or TCP (Transmission Control Protocol). UDP provides an “unreliable” packet based delivery while TCP provides reliable stream oriented delivery. With UDP, you do not have to establish a connection with another node before sending a message; thus, UDP is often termed connectionless. With TCP, you must establish a “virtual circuit” connection with another process before sending data. For more information on TCP/IP features please refer to the *2572 Installation and Operation Guide* and general TCP/IP publications such as *Internetworking with TCP/IP* by Douglas Comer (1991, Prentice Hall).

Client Application

The following example illustrates the general coding technique for acting as a client to the 2572 PLC server. Actual code required will vary depending based upon the TCP/IP software being used. The socket illustrated is for TCP.

```
          /*   Create the Socket             */
          sock=Socket(AF-INET, SOCK_STREAM, ......)
          /*   Connect to the Server              */
          Connect (sock, PLC_Server_IP_Address, ServerPort,
               .....)
          /*   Send the Command                   */
  ------> send(sock, CommandBuffer, CommandLength, ....)
 |L       //   Receive Reply
 |o       receive(sock, ReplyBuffer, ReplyLength,....)
 |o       //   Process the reply information
 |p       ......
  ------  .....
          /*   Terminate Connection               */
          Close(sock)
```

In the example, `CommandBuffer` will contain the message you are sending to the 2572 in the format described in previous sections. Similarly, the `ReplyBuffer` will contain the response from the 2572.

Server Application

The following example illustrates the general coding techniques for opening a TCP server socket.

```
       /*   Create the Socket        */
       sock=socket(AF-INET, SOCK-STREAM, .......)
       /*   Assign a port number to the socket and
       establish a listen queue          */
       bindr=bind(sock, SocketNumber....)
       listen(sock, QueueSize....)
       /*   Wait for a connection         */
       consock=accept(sock,ClientAddress...)
       /*   Process Command               */
       receive(consock, CommandBuffer, CommandLen,....)
  L    /*   Process the command information    */
  o    .....
  o    ......
  p    ......
       send(consock, ReplyBuffer, ReplyLen,...)
       /*   Terminate Connection          */
       close sock
```

In the example above, `CommandBuffer` will contain the message from the PLC Client. `ReplyBuffer` will contain the applicable reply generated by your code.

Error Processing

Due to the nature of network communications, you will experience errors from time to time. Network communications may be lost temporarily, processors may be taken off line, command messages or replies may be lost from time to time. You application must provide recovery for these circumstances. Following are some conditions you should handle.

- *Lost Network Connection (TCP).* This condition could occur if the target 2572 was taken off-line or a network path was interrupted (for example, broken cable). If a TCP connection is lost, you must detect the condition, log / report the error, and try to re-establish the connection. After a certain number of retries, you may wish to inform the operator and allow manual restart.
- *Reply Timeout (UDP).* Under certain conditions a reply may not be received by the application within a reasonable time. With connectionless services it is permissible for the 2572 or the host processor (typically a PC) to drop packets occasionally. To deal with this case, you will need to handle timeout conditions. In processing timeouts, make sure that you wait long enough for the reply to be sent, processed by the 2572/PLC and the answer to be returned. Most of the time, replies on a LAN are received within 100-200 milliseconds; however, under heavy loading coupled with slow PLC scan times, the 2572 input queue can be filled, resulting in a response time of several seconds. Typical timeout values for a local area network should be

approximately 4 seconds. This value should be increased if you are communicating over a wide area network (WAN).

- *Message ID Matching.* Your application should check that the message ID in the reply matches the ID in the request. It is possible for replies to arrive out of sequence or after you have declared a timeout. If you receive a reply with an ID that does not match an outstanding request, you should ignore the reply and log an error in your application.
- *2572 Queue Full Messages.* When the maximum number of messages are contained in a 2572 queue, the module will reject the request and return an error message to the requester. You could use this as a signal to reduce your polling rate, assuming other client applications communicating with the same module also cooperate.
- *Other CAMP Error Messages.* See Appendix A for errors that can be returned. Many of these errors result from errors in coding the application. You can use these to debug your code. Others may result from operator entry errors, PLC configuration errors, etc. Obviously you should report these errors to the operator.
- *Event Logging.* It is very useful to log certain events that may occur during processing. In particular you may wish to log errors. Many developers allow logging thresholds to be set, so that the logger can be used to analyze and debug the application as well as to track errors.
-

3.4. Development and Debugging Tools

To develop the application you will need:

- Siemens® SIMATIC® 505 PLC, rack, and power supply,
- CTI 2572 Ethernet TCP/IP Adapter,
- Siemens® SIMATIC® TISOFT PLC programming software or equivalent,
- Ethernet Network (typically a hub, PC adapter, and cabling),
- Appropriate application programming software (editor, compiler, etc.),
- TCP/IP software and reference documentation.

In addition, it is highly recommended that you acquire a network analysis software package. One such package is *Etherprobe* by General Software (Redmond WA). Etherprobe allows you to capture packets sent to and from your host(s) and the 2572 module(s). Once captured, you can analyze the packets for contents to verify the CAMP commands that are actually being sent and received on the network. You can also use the packet timestamps to evaluate PC processing and 2572/PLC response times. Many developers have found this analysis software to be an invaluable debugging tool.

3.5. TCP/IP Problems

If you cannot communicate with a 2572 on the network, ensure that the PLC rack is powered, that the network parameters (including the IP address) are set, that the module is properly logged into the PLC, and that the network cabling is properly connected. See the *2572 Installation and Operation Guide* for additional information .

After you have verified that the physical network is functioning, you can check out the TCP/IP protocol by sending a PING to the module. PING sends an ICMP echo request to a specified network address to determine if the address is reachable. If the 2572 is reachable at the specified address, it will reply. If the 2572 module does reply, you know that the IP address you specified is correct and that the TCP/IP firmware in the module is working. If the module does not reply to the PING, check that the 2572 address is set properly and that you are using the correct address. See your TCP/IP application documentation for information on sending a PING.

Make sure that you have specified the correct TCP/UDP port number for the PLC server on both the PLC and the client code. If you are using TCP, ensure that the maximum number of connections is not exceeded.

3.6. CAMP Message Processing Problems

The CAMP protocol provides comprehensive error reporting which should help considerably in troubleshooting problems. By design, a server using the CAMP protocol will reply to any message it detects except Type F (error) message. See Chapter 3 for details on CAMP. If you are sending messages to a CAMP server via TCP/IP and are not getting a reply, then you may not have established a socket associated with it. See TCP/IP Problems.

If the CAMP server replies with a Type F message, this means the CAMP message packet is invalid. Be sure you have included both message delimiters, used a valid type, and have properly computed the BCC. See Section *2.2. CAMP Protocol Description*, page 9.

Other errors indicate a problem in processing the message request. See Appendix A for error codes and possible solutions.

APPENDIX A. 2572 ERROR CODES

This Appendix provides a description of the error codes that may be returned by the 2572 in a response message and offers potential solutions to the error conditions.

CAMP Error Codes

This section describes the error codes that may be returned by the CAMP protocol. The error code is located in word 3 of the error message. See Command Error Response on page 23.

HEX	DESCRIPTION	SOLUTION
006E	NITP PROTOCOL ERROR This error may occur when sending Packed Task Code messages. An NITP protocol error is defined as: odd number of characters received, character other than 0-9 or A-F received, checksum did not match computed checksum	Although this error could result from a network error, the most likely problem is an error in building the message. Check your coding. For the Packed Task Code Format, Error Word 4 will contain the message number of the Task Code message where the error was detected. The message number is in Hex ASCII format (e.g. message number 14 is shown as 000E).
0073	BAD OR MISSING DELIMITER The remote device detected a missing delimiter in the message	This error could result from a transient error in the data link. Application logic should retry the command. If the problem persists, you are most likely creating the CAMP message incorrectly. Make coding changes as required.
0074	BAD BLOCK CHECK CHARACTER The remote device found the block check character to be bad.	See error code 0073.
0075	INVALID TYPE The remote device detected an invalid TYPE character in the message.	See error code 0073.
0076	INVALID DATA CHARACTER The remote device detected an invalid data character in the message sent from the remote device. Only characters 0-9 and A-F are allowed.	See error code 0073.

HEX	**DESCRIPTION**	**SOLUTION**
0077	ODD NUMBER OF CHARACTERS The remote device detected an odd number of characters in the message. All CAMP messages have an even number of characters.	See error code 0073.
0078	INVALID DEVICE CODE The remote device detected an invalid device code in the message	See error code 0073.
0080	INVALID ERROR CHARACTER The message contained an unknown error character.	Review your error processing logic. Make sure you are returning a valid error code.
0081	NO WORDS TO WRITE The message type indicated data was to be written but the message did not contain any words to write.	You must include at least one word of data to be written. Change your application as required.
0082	INVALID WORD COUNT The word count field does not match the actual number of data words.	Ensure that the word count matches the actual number of words you are sending.
0083	MEMORY ADDRESS = 0 The memory address contained in the read or write message was 0. This is not a valid memory address.	Your software is generating an invalid memory address. CAMP memory addresses are 1 relative. Review the code that creates the message and make corrections.
0084	WRITE UNSUCCESSFUL The remote device was unable to write the number of words you requested	The error may occur if the number of words specified causes the memory boundary to be exceeded while reading memory. If so, correct either the number of words requested or the starting memory address. This may also indicate a remote device error. Retry, if the problem persists, check the remote device.
0085	INVALID COMMAND CODE The message contained an command code not supported by the remote device.	Correct the request and retry.
008F	INVALID NUMBER OF WORDS You specified a value of 0 for the number of words to write.	Correct the request and retry.
0090	UNSUPPORTED ADDRESS CLASS OR DEVICE CLASS You specified an address class or device that is not supported by the remote device.	Examine the value contained in the address/device class field for errors. Correct as necessary. Check the version of the module firmware. Versions earlier than 5.0 do not support an address class value other than 0000.

HEX	DESCRIPTION	SOLUTION
0091	REQUEST TOO LARGE You tried to read more than 256 words.	Correct the request and retry.
0093	CAMP MAXIMUM RESPONSE SIZE EXCEEDED You have asked for more data via a Packed Task Code request than can be returned in a CAMP response message.	Correct the request and retry.
0094	MAXIMUM NUMBER OF TASK CODES PER MESSAGE EXCEEDED You included more than 14 Task Codes in the Packed Task Code request message.	Correct the request and retry.
0095	INVALID TASK CODE CHARACTER COUNT The NITP character count is outside the range of valid values or the count exceeds the number of characters remaining in the message.	Correct the request and retry.
00AC	MEMORY READ ERROR The remote device could not execute the command to read data.	This error may occur because you are specifying a starting memory address that is not valid for the remote device. If so, correct the memory address. The error may occur if the number of words specified causes the memory boundary to be exceeded while reading memory. If so, correct either the number of words requested or the starting memory address. This error may be encountered if you specify an odd number word count when using address classes which access real numbers. Set the word count to an even number, if in error. Some address classes require data to be accessed in multiple words. See Appendix B. This error may be encountered if you do not specify the correct multiple in the word count This error may occur due to a problem with the remote device. Retry the command message. If the problem persists, check the remote device.

HEX	**DESCRIPTION**	**SOLUTION**
00AD	MEMORY WRITE ERROR The remote device could not execute the command to write memory.	This error may occur because you are specifying a starting memory address that is not valid for the remote device. If so, correct the memory address. The error may occur if the number of words specified causes the memory boundary to be exceeded while writing memory. If so, correct either the number of words requested or the starting memory address. This error may occur due to a problem with the remote device. Retry the command message. If the problem persists, check the remote device.
00C7	MESSAGE QUEUE FULL The message was rejected by the module because an input queue contained the maximum number of entries. The module is receiving requests faster than they can be processed by the PLC. The high byte of data word 4 in the error message will identify the queue. 01 = Packed Task Code Queue 02 = PLC Processing Queue The low byte of data word 4 will indicate the maximum entries for the queue.	Re-Send the request. If the problem persists, you must correct the system problem that is causing the module overload. Check the following: PLC Configuration: Make sure the task codes per scan are set to 8. Check the scan time setting; a large fixed scan time will degrade communications performance on most PLC models. For best communications performance in most situations, set the scan to variable. Application: You may be inadvertently sending new requests before waiting an reasonable time for a response. If sustained, over time this action can fill up the queue. Correct as necessary. Other: If multiple PCs are continuously polling a single 2572, you may need to slow the poll rate. Alternately, you may need to install an additional 2572 in the PLC rack and split transactions among them.

NITP Task Code 00 Error Codes

This group of error codes is returned by the remote PLC when an error is encountered processing a task code. The error code is returned in a message which contains a task code 00, followed by the specific error code. The error codes are summarized below for your reference. Many of these errors apply to programming task codes or special function routines. Common errors which you may encounter are shown in italics.

SIMATIC® 505 Task Code 00 Error Listing

HEX	DESCRIPTION
01	Reset Current Transaction
02	Address out of Range (Other than Ladder Logic)
03	Requested Data not Found
04	Illegal Task Code Request (e.g. Task Code not Supported)
05	Request Exceeds Program Memory Size (Ladder Logic)
06	Diagnostic Fail upon Power Up
07	Fatal Error Detected
08	Keylock Protect Error
09	*Incorrect amount of Data sent with Request*
0A	Illegal Request in Current Operational Mode
0B	Network was not Deleted
0C	Attempted Write Operation Did Not Verify
0D	Illegal Number of ASCII Characters Received
0E	Illegal Write to Program Memory (Non Volatile)
0F	Data not Inserted
10	Data not Written
11	*Invalid Data sent with the Command*
12	Invalid Operation with NIM (Obsolete)
13	The store and forward buffer is busy
14	No response from the Special Function Module
15	Illegal Instruction found in program memory on a Program to Run transition
16	*Attempted Write to a Protected Variable (e.g. TCC, TCP)*
17	No response from PLC (e.g. Single Scan not performed)
18	Requested memory size exceeds total available memory
19	Requested Memory size is not a multiple of block allocation size
1A	Requested memory size is less than minimum defined value.
1B	Requested memory size is larger than maximum defined value
1C	*PLC Busy - Cannot complete the requested operation*
1D	Comm error in HOLD mode - Transition to Run not allowed
1E	*Port Lockout is Active (ref task Code 48)*
21	I/O Configuration Error - too many points
22	I/O Configuration Conflict
3F	Bus Error Detected

40	Operating System Error Detected
41	Invalid Control Block Type
42	Control Block Number out of range
43	Control Block does not exist
44	Control Block already exists
46	Offset out of range.
47	Arithmetic error detected while writing loop or analog alarm parameters
48	Invalid SF Program type
49	Instruction number or RAMP/SOAK Step number out of range
4A	*Attempt to access an integer only variable as a real.*
4B	*Attempt to access a real -only value as an integer.*
4C	*Task Code buffer overflow -- too much data requested*
4D	Control Block size error. (Maximum = 32767 bytes)
4E	*Attempt to write a read only variable.*
4F	Invalid variable type for this operation
50	Task code request buffer too large (PLC internal error)
51	Invalid SF Statement size specified
52	Invalid return value
53	Attempt to execute a Cyclic statement in a non-cyclic SF program
54	Control Block is disabled
55	Control Block is not disabled
56	Attempt to perform a FTSR-OUT Statement on an empty FIFO
57	Attempt to perform a FTSR-IN Statement on a full FIFO
58	Stack overflow while evaluating a MAATH, IF-THEN, or IMATH expression
59	Maximum SF Subroutine nesting level exceeded (Max depth = 4)
5A	Arithmetic Overflow
5B	Invalid Operator in an IF, MATH, or IMATH expression
5C	S Memory Overflow
5D	Attempt to divide by 0 (IMATH Statement)
5E	FIFO is incompatible with FTSR-IN or FTSR-OUT statement
5F	FIFO is invalid
60	Invalid Data Type code (Usually MATH, IMATH, or IF-THEN statement)
61	RAMP/SOAK step type mismatch. Error is logged by Task Code 6 (Write RAMP/SOAK step) if the fields are being updated and the step type indicated but the task code does not match the step type of the step being modified.

APPENDIX B. ADDRESS CLASS INFORMATION

General Information

The CAMP Memory Transfer commands contain a field designated as ADDRESS CLASS. Version 5.0 and later of the 2572 firmware allows you to use the ADDRESS CLASS field to access data types other than V memory. The high byte of the ADDRESS CLASS corresponds to the Siemens® SIMATIC® 505 data types as shown in the tables below. The tables indicate the format of the returned information and whether the variable can be read and written (r/w) or read only.
Carefully read the notes that apply to certain variable types.

Accessing Real Numbers and 32 Bit Variables

Real numbers and 32 bit fields require two words data words in a CAMP message. When accessing real numbers or 32 bit fields, you must specify a WORD COUNT which represents the actual number of data words which will be contained in the CAMP message. For example, to retrieve a single Loop Process Variable (Type 25), the WORD COUNT would be set to 2 because a single real number will return two data words.

Common Data Types

Type	**Variables**	**Name**	**Fmt**	**Access**
00	CAMP DEFAULT	V	Integer	r/w
01	V Memory	V	Integer	r/w (1)
02	K Memory	K	Integer	r/w (1)
06	Discrete Input Packed	X	16 bits	r/w (2)
07	Discrete Output Packed	Y	16 bits	r/w (2)
08	Control Register Packed	C	16 bits	r/w (2)
09	Word Input	WX	Integer	r/w
0A	Word Output	WY	Integer	r/w
0E	Timer/Counter Preset	TCP	Integer	r/w
0F	Timer/Counter Current	TCC	Integer	r/w
10	Drum Step Preset	DSP	Integer	r/w
11	Drum Step Current	DSC	Integer	r/w
12	Drum Count Preset	DCP	Integer	r/w (3)
1A	System Status Word	STW	16bit	read (4)
1B	Drum Current Count	DCC	32bit	read

(1) Type 01 and 02 may be used to read memory addresses beyond 65535. The lower byte of the Address Class is combined with the Address field to yield a 24 bit address. For example, to access Type 01 (V Memory) location 73245 (0x 011E1D):

The Address Class field would contain - 01**01**
The Address field would contain - **1E1D**

(2) These data types are accessed as a group of 16 bits. When using CAMP Memory Transfer to write discrete values, you must write them 16 bits at a time. If you wish to write bits in smaller increments, you must use the applicable task code instead.

The first value accessed will be stored in the least significant bit of the word, with each successive bit stored in the next higher bit position. For example, if you read two words of type 08 (control relay) starting at 10, the result returned would be as follows:

WORD 1

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Element	C25	C24	C23	C22	C21	C20	C19	C18	C17	C16	C15	C14	C13	C12	C11	C10

WORD 2

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Element	C41	C40	C39	C38	C37	C36	C35	C34	C33	C32	C31	C30	C29	C28	C27	C26

(3) Type 12 can read and write only event drums, not standard drums. ***This variable must be accessed in groups of 16 words, one word for each drum step.***

(4) The System Status Word can only be read using this data type. However, the applicable task code may be used to write data to applicable Status Words.

Loop Data Types

TT	Loop Variables	Name	Fmt	Access
20	Loop Gain	LKC	Real	r/w
21	Loop Reset Time (min)	LTI	Real	r/w
22	Loop Rate Time (min)	LTD	Real	r/w
23	Loop High Alarm Limit	LHA	Real	r/w (1)
24	Loop Low Alarm Limit	LLA	Real	r/w (1)
25	Loop Process Variable	LPV	Real	r/w
26	Loop PV High Limit	LPVH	Real	r/w (2)
27	Loop PV Low Limit	LPVL	Real	r/w (2)
28	Loop Orange Dev Alarm Limit	LODA	Real	r/w (1)
29	Loop Yellow Dev Alarm Limit	LYDA	Real	r/w (1)
2A	Loop Sample Rate (sec)	LTS	Real	r/w
2B	Loop Setpoint	LSP	Real	r/w
2C	Loop Output (%)	LMN	Real	r/w
2D	Loop Status (V) Flags	LVF	16 bits	read (3)
2E	Loop Control (C) Flags	LCF	32bit	r/w
2F	Loop Ramp/Soak Status Flags	LRSF	16 bits	read (3)
30	Loop Error	LERR	Real	read
31	Loop Bias	LMX	Real	r/w
32	Loop High-High Alarm Limit	LHHA	Real	r/w (1)
33	Loop Low-Low Alarm Limit	LLLA	Real	r/w (1)
34	Loop Rate of Change Alarm Limit	LRCA	Real	r/w
35	Loop Setpoint High Limit	LSPH	Real	r/w (1)
36	Loop Setpoint Low Limit	LSPL	Real	r/w (1)
37	Loop Alarm Deadband	LADB	Real	r/w
38	Loop Raw High Alarm Limit	LHAR	Integer	r/w (1)
39	Loop Raw Low Alarm Limit	LLAR	Integer	r/w (1)
3A	Loop Raw Process Variable	LPVR	Integer	r/w (4)
3B	Loop Raw Orange Dev Alarm Limit	LODAR	Integer	r/w (1)
3C	Loop Raw Yellow Dev Alarm Limit	LYDAR	Integer	r/w (1)
3D	Loop Raw Output	LMNR	Integer	r/w
3E	Loop Raw Setpoint	LSPR	Integer	r/w
3F	Loop Raw Error	LERRR	Integer	read
40	Loop Raw High-High Alarm Limit	LHHAR	Integer	r/w (1)
41	Loop Raw Low-Low Alarm Limit	LLLAR	Integer	r/w (1)
42	Loop Raw Alarm Deadband	LADBR	Integer	r/w
48	Loop Raw Bias	LMXR	Integer	r/w
49	Loop Raw Setpoint Low Limit	LSPLR	Integer	r/w (1)
4A	Loop Raw Setpoint High Limit	LSPHR	Integer	r/w (1)
4B	Loop C Flags - MSW	LCFH	Integer	r/w
4C	Loop C Flags - LSW	LCFL	Integer	r/w
4D	Loop Derivative Gain Limiting Coef.	LKD	Real	r/w
4E	Loop Ramp/Soak Step Number	LRSN	Integer	r/w
4F	Loop Alarm Ack Flags	LACK	Integer	read (3)

(1) When writing these values, you must ensure the limits do not overlap. No checking is performed. For example, you can set the Loop Low Alarm limit (LLA) higher than the Loop High Alarm limit (LHA). However, if you attempt to set the values beyond the process variable limits (LPVL and LPVH), the value will clamp to the nearest endpoint of the range. Similarly, if a process variable limit is changed so that this variable is outside the limit, the variable will be clamped to the nearest endpoint of the range.

(2) Setting the LPVH value lower than the current LPVL will cause the LPVL to clamp to the new LPVH value. Similarly setting the LPVL value higher than current LPVH value will cause the LPVH to clamp to the new LPVL value.

(3) These flags cannot be written using this data type. However, the enable and acknowledge bits may be written using task codes.

Alarm Data Types

TT	**Analog Alarm Variables**	**Name**	**Fmt**	**Access**
50	Alarm High Alarm Limit	AHA	Real	r/w (1)
51	Alarm Low Alarm Limit	ALA	Real	r/w (1)
52	Alarm Process Variable	APV	Real	r/w
53	Alarm PV High Limit	APVH	Real	r/w (2)
54	Alarm PV Low Limit	APVL	Real	r/w (2)
55	Alarm Orange Dev Alarm limit	AODA	Real	r/w
56	Alarm Yellow Dev Alarm Limit	AYDA	Real	r/w
57	Alarm Sample Rate (sec)	ATS	Real	r/w
58	Alarm Setpoint	ASP	Real	r/w
59	Alarm V Flags	AVF	Integer	read (3)
5A	Alarm C Flags	ACF	Real	r/w
5B	Alarm Error	AERR	Real	read
5C	Alarm High-High Alarm Limit	AHHA	Real	r/w (1)
5D	Alarm Low-Low Alarm Limit	ALLA	Real	r/w (1)
5E	Alarm Rate of Change Alarm Limit	ARCA	Real	r/w
5F	Alarm Setpoint High Limit	ASPH	Real	r/w (1)
60	Alarm Setpoint Low Limit	ASPL	Real	r/w (1)
61	Alarm Alarm Deadband	AADB	Real	r/w
62	Alarm Raw High Alarm Limit	AHAR	Integer	r/w (1)
63	Alarm Raw Low Alarm Limit	ALAR	Integer	r/w (1)
64	Alarm Raw Process Variable	APVR	Integer	r/w
65	Alarm Raw Orange Deviation	AODAR	Integer	r/w
66	Alarm Raw Yellow Alarm Limit	AYDAR	Integer	r/w
67	Alarm Raw Setpoint	ASPR	Integer	r/w
68	Alarm Raw Alarm Deadband	ADBR	Integer	r/w
69	Alarm Raw Error	AERRR	Integer	read
6A	Alarm	AHHAR	Integer	r/w
6B	Alarm Raw Low-Low Alarm Limit	ALLAR	Integer	r/w (1)
6F	Alarm Raw Setpoint Low Limit	ASPLR	Integer	r/w (1)

70	Alarm Raw Setpoint High Limit	ASPHR	Integer	r/w (1)

TT	Analog Alarm Variables (Continued)	Name	Fmt	Access
71	Alarm MSW Alarm C Flag	ACFH	Integer	r/w
72	Alarm LSW Alarm C Flags	ACFL	Integer	r/w
73	Analog Alarm - Alarm Ack Flag	AACK	Integer	read (3)

(1) When writing these values, you must ensure the limits do not overlap. No checking is performed. For example, you can set the Analog Low Alarm limit (ALA) higher than the Analog High Alarm limit (AHA). However, if you attempt to set the values beyond the process variable limits (APVL and APVH), the value will clamp to the nearest endpoint of the range. Similarly, if a process variable limit is changed so that this variable is outside the limit, the variable will be clamped to the nearest endpoint of the range.

(2) Setting the APVH value lower than the current APVL will cause the APVL to bet set to the new APVH value. Similarly setting the APVL value higher than current APVH value will cause the APVH to be set to the new APVL value.

(3) These flags cannot be written using this data type. However, the enable and acknowledge bits may be written using task codes.

APPENDIX C. TASK CODE OVERVIEW

Devices which communicate with Siemens® SIMATIC® 505 PLCs use task codes to read and write data in the PLC. This Appendix provides a brief summary of task codes and the NITP protocol. The task codes typically used for supervisory control and monitoring applications are described in detail in several Siemens® publications.

NOTE:
Before implementing an application using the task codes, you should obtain a publication such as the SIMATIC® TI575 Task Code User Manual, *Order* # PPX:575-8104-1.

The 2572 uses a variation of the Siemens® SIMATIC® Non-Intelligent Terminal Protocol (NITP) for task code access. NITP is an ASCII protocol with the following format.

Start Delimiter :	Character Count	Message Body	Error Check Code	End Delimiter ;

Both command and response messages start with a colon (ASCII 3A) and end with a semicolon (ASCII 3B). All data is represented as the ASCII equivalent of the hexadecimal representation. For example, the decimal number 58 is expressed as the characters 3A (hex 33 41). The character count is a simple count of all characters in the message, including the beginning and end delimiters. The maximum message length of a single message is 72 characters

NOTE: The CAMP implementation omits the beginning and ending delimiters, but keeps the same character count.

The message body consists of the task code and related parameters and data. Commonly used task codes include:

01	Read Word Memory Random
02	Write Word Memory Random
50	Read User Word Area Block
51	Write User Word Area Block
5A	Write Block

7E	Read Random
7F	Read Block

Word Codes

The types of data that can be accessed using these task codes closely match the data element types listed in Appendix B. However, rather than using the data element type and address, the type and address of the data to be accessed are bit encoded into a structure known as a *word code*. Word codes may consist of 1 or two words, depending upon the data type and address being accessed.

Word codes are classified into three categories:

- Category 1 word codes access variables common to all PLC models (such as V, WX, DCC).
- Category 2 word codes access loop and alarm variables plus a few special types.
- Category 3 word codes allow configuration of the 545, 555, and TI575 timelime.

If you are communicating with a newer PLC (545, 555, or TI575), you may use any word code with the above task codes. For example task code 01 or 7E can be used to randomly read V-memory or loop/alarm variables. Similarly, Task code 50 or 7F can be used to block read V-memory or loop/alarm variables.

If you are communicating with a 565 PLC, you will find that you cannot use Task Code 01 to read the Category 2 variables (you must use 7E). Also, you cannot use Task Code 02 to write Category 2 variables. There is no equivalent loop random write, only a block write (5A). If you want to access a TI525 or TI535 PLC, these do not support loops and also use a different addressing scheme (absolute addressing).

As a practical matter, most installations using the 2572 will be a 545, 555, or TI575. You may, however, find some TI565 PLCs still used in “hot backup” applications. If you don’t care about supporting the TI565 PLCs, it is probably easiest to use task code 01 and 02 for all random access.

There are a few relevant task codes that use the data element types rather than word codes. See appendix B for the data element types. They are:

59	Write Discrete I/O Status by Data Element Type
6B	Read Discrete I/O Status by Data Element Type
9D	Read Random Block via Data Element Type and Offset
9E	Write Random Block via Data Element Type and Offset

Task code 59 and 6B are supported by all 545, 555, TI565, and TI575 PLCs. This is a very useful task code because it reads and writes discrete data in a packed format.

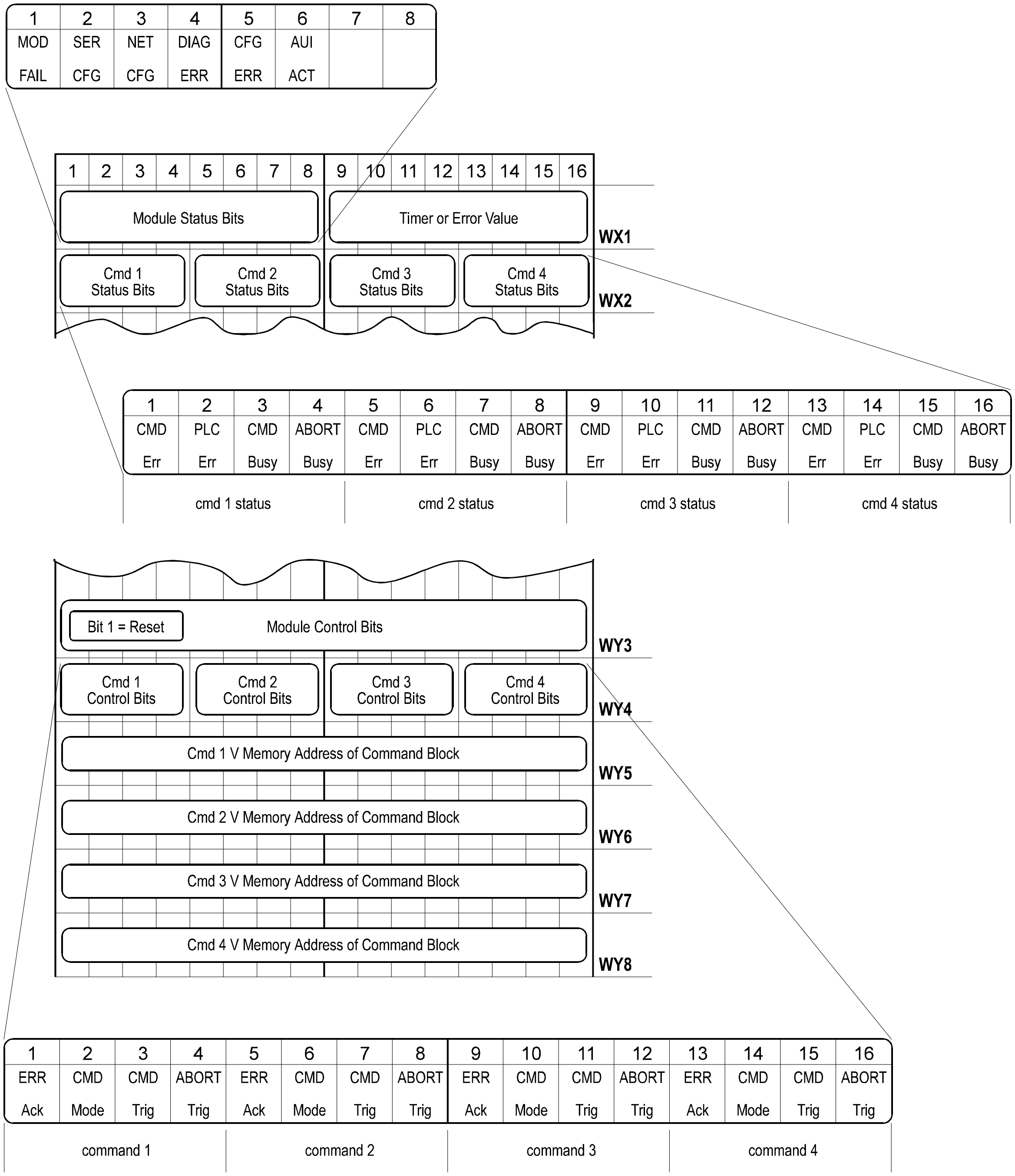
Task Codes 9D and 9E are useful because they can read any data type but avoid the complexity of calculating word codes. They also let you request several blocks of different data types in the same task code. However, Task code 9D and 9E are supported by only later models/ firmware revisions of the SIMATIC® PLCs including: 545-1102 Release 3.0, 555 Release 3.0, and TI 575 Release 3.0. The 545-1101, a commonly installed PLC, does not support task code 9D or 9E. Thus, use task codes 9D and 9E only if you don’t need to support the 545-1101 or older PLC models.

APPENDIX D. REFERENCE MATERIAL

TABLE 1. **Hex to ASCII Conversion**

00	NUL	20	space	40	@	60	`
01	SOH	21	!	41	A	61	a
02	STX	22	"	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	'	47	G	67	g
08	BS	28	(	48	H	68	h
09	HT	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	

WX / WY Quick Reference



1	2	3	4	5	6	7	8
MOD FAIL	SER CFG	NET CFG	DIAG ERR	CFG ERR	AUI ACT		

	1–8	9–16
WX1	Module Status Bits	Timer or Error Value
WX2	Cmd 1 Status Bits, Cmd 2 Status Bits	Cmd 3 Status Bits, Cmd 4 Status Bits

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CMD Err	PLC Err	CMD Busy	ABORT Busy	CMD Err	PLC Err	CMD Busy	ABORT Busy	CMD Err	PLC Err	CMD Busy	ABORT Busy	CMD Err	PLC Err	CMD Busy	ABORT Busy
cmd 1 status				cmd 2 status				cmd 3 status				cmd 4 status			

	1–16
WY3	Bit 1 = Reset; Module Control Bits
WY4	Cmd 1 Control Bits, Cmd 2 Control Bits, Cmd 3 Control Bits, Cmd 4 Control Bits
WY5	Cmd 1 V Memory Address of Command Block
WY6	Cmd 2 V Memory Address of Command Block
WY7	Cmd 3 V Memory Address of Command Block
WY8	Cmd 4 V Memory Address of Command Block

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ERR Ack	CMD Mode	CMD Trig	ABORT Trig	ERR Ack	CMD Mode	CMD Trig	ABORT Trig	ERR Ack	CMD Mode	CMD Trig	ABORT Trig	ERR Ack	CMD Mode	CMD Trig	ABORT Trig
command 1				command 2				command 3				command 4			