CTI 2500P-ACP1

Application Coprocessor

User Manual

Version 2.06

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 at 1-800-537-8398 or email us at *sales@controltechnology.com*.

REVISION HISTORY		
V1.0	7/1/2014	Initial Release
V1.01	7/22/2014	Corrected typos and added details to several topics
V1.02	8/6/2014	Swapped order of some sections within document. Included additional Firmware Update Error Code descriptions.
V1.03	8/28/2014	Included additional CAMP Client Error Codes.
V1.04	9/8/2014	Corrected page formatting errors.
V1.05	9/11/14	Corrected typos and improved descriptions in several sections. Corrected description of operation for DIP switch positions. Corrected description of Cycle Watchdog Timer operation.
V1.06	9/18/14	Corrected Corrective Action comments for CAMP Client Error Codes 172-173.
V1.07	10/9/14	Corrected Data Cache V-Memory and K-Memory data type selections. Added references to ***CTI Workbench*** as Part No. 2500P-WB-USB.
V1.08	6/11/15	Added descriptions for EtherNet/IP interfaces: I/O Scanner, I/O Adapter, and Tag Client
V1.09	6/27/15	Added CIP General Status Codes and UCMM Extended Status Codes for EtherNet/IP interfaces.
V1.10	7/16/15	Corrected typos.
V1.11	7/24/15	Improved tables for CIP General Status and UCMM Extended Status Codes. Added details to Chapter 8 – Application Development.
V1.12	8/3/15	Corrected missing cross-reference and updated screen capture photos.
V1.13	9/11/15	Corrected typos regarding the number of Data Cache variables and Modbus commands.
V1.14	5/18/16	Corrected formatting errors and added descriptions for features added in firmware V3.08: TCP/UDP Management functions (Section 7.1.8) and Ethernet Port Isolation (Sections 2.5, 3.1.2, and 6.1).
V1.15	9/14/16	Corrected typos and invalid cross-reference links. Added 'Bookmarks' and 'Table of Contents Hyperlinks' to PDF document.
V1.16	9/19/16	Corrected capitalization for all instances of "EtherNet/IP".
V1.17	6/5/17	Added decimal values and details to CIP General Status Code and CIP UCMM Extended Status Codes reported by EtherNet/IP fieldbus protocol.
V1.18	1/9/18	Corrected decimal values in the CIP UCMM Extended Status Codes reported by EtherNet/IP fieldbus protocol (Appendix A).
V1.19	7/16/18	Added section references for details of the Host PLC interface methods listed in Introduction (Chapter 1).

REVISION HISTORY		
V2.0	12/11/18	Corrected errors in descriptions for Ethernet/IP Scanner (Section 7.1.5), Ethernet/IP Adapter (Section 7.1.6), and Ethernet/IP Tag Client (Section 7.1.7). Added functionality for ACP1 enhancements made in Firmware Version 4: MQTT Client protocol description (Section 7.1.9) and Error Codes. Ethernet/IP Scanner Explicit Messaging description (Section 7.1.5). Module Configuration: Ethernet Network Parameters (Sections 5.1.1 – 5.1.2). Module Configuration via Web Server (Section 5.2). Added IP alias feature to description of Ethernet Port operation (Section 2.5). Added reporting for Run-time Error 392: 'Invalid Array Index' (Appendix A).
V2.01	12/21/18	Enhanced description of MQTT Client to include operation and configuration details (Section 7.1.9).
V2.02	2/19/19	Removed Data Type Conversion operation from CTI Data Cache configuration description (Section 6.1.2). These functions were removed from ACP1 Firmware V4.09 to maintain compatibility with existing ACP1 projects.
V2.03	1/6/21	Corrected typos.
V2.04	8/13/21	Removed reference to high-speed backplane in Section 6.1
V2.05	4/30/2024	Added a note that online change is not applicable to the ACP1
V2.06	9/19/2024	Added note in Section 8.4.4 that RETAIN variable storage is limited to 11,500 bytes, or 992 bytes when using F_SAVERETAIN and F_LOADRETAIN function blocks.

PREFACE

This ***User Manual*** provides reference information for the CTI 2500P-ACP1 Application Coprocessor module. The information in this manual is directed to individuals who will be installing, configuring, and operating the product, developing an application to execute on the product, and those who will be designing systems that utilize the product. Detailed information regarding module setup and installation is included in this document.

This manual also provides a functional overview of the 2500P-ACP1 product and IEC-61131-3 compliant operation provided by the module firmware. It is not intended to serve as a programming reference for application development. Those details are provided in the ***CTI Workbench*** on-line Help system.

USAGE CONVENTIONS

NOTE
Notes alert the user to special features or procedures.

CAUTION
Cautions alert the user to procedures that could damage equipment.

WARNING
Warnings alert the user to procedures that could damage equipment and endanger the user.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

The 2500P-ACP1 Application Coprocessor module is a general-purpose auxiliary controller that enhances the capabilities of all Siemens Series 505® and CTI 2500 Series® PLC systems. This Advanced Function Module (AFM) includes high-speed processing and multi-protocol communications support to provide existing systems with a significant increase in performance and functionality.

The 2500P-ACP1 generally runs as a PLC coprocessor performing complex logic/math functions, data logging, and communications with external devices. Although the 2500P-ACP1 can operate as a stand-alone controller, the application generally requires data transfer between a host PLC and the module. The ACP1 module includes two options for real-time data transfer:

- Data Cache: Proprietary link offering enhanced data throughput to CTI 2500 Series® controllers via an external Ethernet connection. Provides access to all PLC memory types and up to 4096 PLC memory locations. This interface is described in Section 6.1.
- Direct PLC I/O bus: Allows the ACP1 coprocessor to emulate a standard I/O module so that discrete and/or word image register data is transferred each PLC scan. This operation is described in Section 6.2.

CTI Workbench is used for configuration of module parameters and development of the user application program for the 2500P-ACP1. ***CTI Workbench*** is a full-featured utility with integrated configuration tool, programming editor, debugger, data monitor, and simulator. The application program may be developed in any of five IEC-61131 programming languages. A complete library of functions is provided to perform the following tasks:

- Complex mathematical computations
- Boolean logic
- Data conversion
- String handling
- Timer/Counter operations
- PID control
- Alarm monitoring
- Complete data logging system (including time-stamped data storage and automatic FTP file transfer to remote FTP Server)

The 2500P-ACP1 supports Ethernet communications through two 10/100Mb Ethernet ports and RS-232/RS-422 connections through the serial port.

Embedded client and server protocols can be used for data transfer with other controllers and devices. All message processing for client operation (request triggering, packet send/receive, message validation, data insertion/extraction, and error handling/retries) is done in the module with no PLC logic required.

The 2500P-ACP1 module supports operations for the following communications protocols:

- CAMP Client
- Open Modbus Client and Server
- Network Data Exchange (Binding) Publish/Subscribe
- Modbus RTU Master and Slave (serial)
- General ASCII Send/Receive (serial)
- EtherNet/IP I/O Scanner, I/O Adapter, and Tag Client
- User-defined messages via TCP/IP client and server functions
- User-defined messages via UDP client and server functions

The module provides extensive diagnostic facilities, accessible via a standard web browser, to monitor module status and aid in the detection and correction of errors. The web server provides access to product information, Ethernet operational statistics, and diagnostic history.

IMPORTANT NOTE: The 2500P-ACP1 does **NOT** support online change of the Workbench application during operation. If you need this capability, you should use the 2500P-JACP Janus Application Coprocessor.

1.1 Definition of Terms

Host PLC	PLC which controls the I/O for the system where the ACP1 runs as a Co-Processor. The Host PLC may be a SIMATIC® 505 or CTI 2500 Series® controller. The 2500P-ACP1 module exchanges data with the Host PLC via I/O backplane or high-speed Ethernet (Data Cache) channel supported by CTI 2500 Series® PLCs.
IP	Internet Protocol: A suite of protocols used to relay data packets across networks.
LAN	An acronym for Local Area Network. In this manual it refers to an Ethernet local area network. All devices on a local area network can directly communicate with each other and are members of the same broadcast domain.
Mapping	The association of data embedded in protocol messages with application program variables, enabling data to be transferred between the ACP1 program and external devices.
Network Device	General term for equipment with which the ACP1 can communicate, such as PLCs, workstations, operator interfaces, motor drives, and weighing stations.
PLC	An acronym for a Programmable Logic Controller
Protocol	A system of digital message formats and rules for exchanging messages between systems. Also, the task that implements the protocol.
VLAN	An acronym for Virtual Local Area Network. VLANs provide a means of subdividing a physical Ethernet LAN into logical isolated LANs, each with its own broadcast domain. VLANs are implemented in Ethernet switches.
SD Card	A ***S**ecure **D**ata* Card is a portable non-volatile memory card with a data format and form factor that complies with standards maintained by the SD card association. The 2500P-ACP1 uses an SD card for non-volatile storage of system and user files.

1.2 Getting Started

If you are not familiar with the 2500P-ACP1 Application Coprocessor module, you should read the remainder of this chapter and *CHAPTER 7* which describes the communication interfaces supported by the ACP1 module.

To begin using your ACP1, you will need to install and configure the module, make the necessary Ethernet and/or serial connections, develop program logic (if required), and configure the communication protocols as needed to meet your application requirements.

- **Installation and Configuration:** The 2500P-ACP1 installs in a CTI 2500 Series® base or a Siemens Series 505® base. See *CHAPTER 3* for installation planning tips and instructions for installing the module, and *CHAPTER 5* describes the steps for module configuration.
- **Communications:** The ACP1 includes drivers for communicating with many different devices. The optional Host PLC Interfaces are described in *CHAPTER 6*. Ethernet and serial communication protocols are detailed in *CHAPTER 7*.
- **Application Development:** *CHAPTER 8* provides a comprehensive overview of design, organization, development, and execution of 2500-ACP1 applications using the ***CTI Workbench*** software product. ***CTI Workbench*** (sold separately as Part No. 2500P-WB-USB) is a full-featured Integrated Development Environment (IDE) consisting of program editors, protocol configuration tools, debugger, simulator, and on-line Help system. Contact CTI or visit http://controltechnology.com for more information.
- **ACP1 Module Firmware Update:** *CHAPTER 9* of this manual contains instructions on procedures for updating the 2500P-ACP1 firmware.

CHAPTER 2 FRONT PANEL

2.1 Status Indicator LEDs

At the top of the module front panel are three status LEDs. The function of the LEDs is described in the following table.

LED	State	Indication
STATUS	Off	Module not operational
	Flashing	Module not ready – Operator action required (Module error, Watchdog timeout, Application program not found, or Host interface failure)
	On	Module operation is normal
ACTIVE	Off	Application program stopped
	Flashing	Program loaded but logic is not running (PAUSED state). I/O interface and communication protocols are active.
	On	Application program is executing (RUN state)
USER	Off	Controlled by application logic
	Flashing	
	On	

2.2 LED Multi-Segment Display

The Multi-Segment Display (MSD) is located below the status LEDs. The MSD is used to display status and error codes. During normal operation the MSD displays the TCP/IP address of the product, one octet at a time. When an error is encountered, the MSD will also display an Error Code. See *APPENDIX A: ERROR CODES* for a list of error codes and descriptions.

2.3 Reset Button

The Reset Button allows you to initiate a “soft reset” for the 2500P-ACP1 module. This reset is equivalent to cycling power to the module. When the button is depressed (using a pointed object such as a ball point pen, the module is restarted after an orderly shutdown of the application program. This reset action can be disabled by setting module switch (SW3) to CLOSED position. Section 3.1.2 contains information on location and setting for module switches.

2.4 Ethernet Status Indicators

The Ethernet LEDs indicate the state of the TCP/IP interface and whether the module is transmitting and receiving data via the Ethernet as shown in the following table.

LED	State	Indication
NS (Network Status)	Off	TCP/IP is not operational.
	On-Red	TCP/IP is operational. A device with the same IP address as this 2500P-ACP1 module has been detected.
	On-Green	TCP/IP is connected and operational.
XMT (Transmit)	Flashing	Ethernet port is transmitting data
RCV (Receive)	Flashing	Ethernet port is receiving data.

2.5 Ethernet Ports

The 2500P-ACP1 provides two Ethernet ports capable of operating at 10/100Mb, half or full duplex. The speed and duplex mode are automatically negotiated with the device connected to the port. Each port supports auto-crossover capability, allowing the port to be connected to an external Ethernet switch or directly to a device, such as a laptop or 2500 Series® controller. Both ports are functionally equivalent.

The 2500P-ACP1 incorporates an Ethernet switch which is connected to both Ethernet ports and the microprocessor. The switch allows either port to communicate with the microprocessor. The two ports can be connected or isolated from each other (see 'Port Isolation' below). Besides providing this connectivity, the switch also provides hardware protection against network broadcast/multicast storms.

It is also possible to enable 'IP aliasing" by configuring an Alias IP Address and Alias Subnet Mask. This allows two IP addresses to be associated with the ACP1 module so that each Ethernet port can be connected to a separate sub-network. When 'IP aliasing' is enabled, either Ethernet port can be used with either sub-network (i.e. the IP Address and Subnet Mask is not port specific). When using this feature, Port Isolation should always be enabled.

Port Isolation can be enabled by setting module switch (SW4) to CLOSED position. This setting blocks forwarding of all Ethernet packets between the two ports and allows the ACP1 to be used with redundant network topologies without creating network loops. Section 3.1.2 contains information on location and setting for module switches.

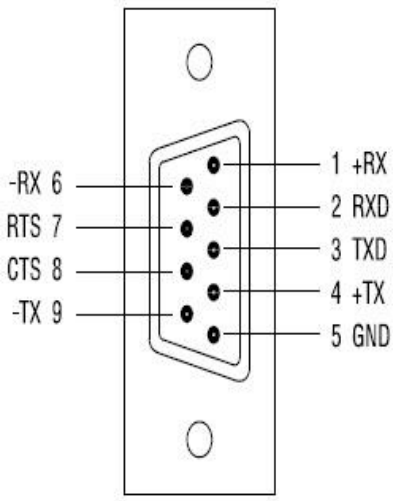
2.6 Ethernet Port LEDs

Each Ethernet port connector contains two embedded LEDs. The **LINK** LED indicates whether the Ethernet port is successfully connected to another Ethernet device, such as a network switch. The **ACTIVITY** (ACT) LED provides visual indication that Ethernet packets are being received or transmitted via the port. See the following table below for more information.

LED	State	Indication
Link	Off	Ethernet link is not available.
	On	Ethernet link is available.
Act (Activity)	Off	No Ethernet frames are being transmitted on the network to which the port is connected.
	Flashing	Ethernet frames are being transmitted on the network to which the port is connected

2.7 Serial Port

The male DB9 connector on the front panel provides the serial port interface. Modbus-RTU (Master or Slave) and General ASCII Send/Receive data protocols are supported for the serial port and managed by the application program. All port parameters, including the selected electrical interface (RS-232 or RS-422), are set by software configuration via ***CTI Workbench***. The cable used with the external device must connect to the pins used by the selected electrical interface.



RS-232 (Subset) Pinout:

Pin	Signal	Description
2	RXD	Receive Data
3	TXD	Transmit Data
5	GND	Signal Ground
7	RTS	Request to Send (optional)
8	CTS	Clear to Send (optional)

RS-422 Pinout:

Pin	Signal	Description
1	+RX	Receive Data (+)
4	+TX	Transmit Data (+)
6	-RX	Receive Data (-)
9	-TX	Transmit Data (-)

NOTE:
A serial port connection to CTI Workbench is not supported by the ACP1 module. This interface must be made using TCP/IP connection.

CHAPTER 3 INSTALLATION

This section discusses the items to consider while planning the 2500P-ACP1 application, and the actual steps for installation of the module.

3.1 Installation Planning

3.1.1 SD Card

The 2500P-ACP1 uses a Secure Date (SD) card for storage of module configuration data, executable program, application program source files, and user data files. Because all configuration and operational files are contained on the SD card, the complete module profile can be transferred simply by swapping SD cards.

A 4GB High-Capacity (SDHC) card is shipped with each unit and should be inserted into the SD card receptacle located on the circuit board as shown below. Although the module will operate without a SD card installed, the functionality is severely diminished.

SD Card Selection

There are three sizes of SD cards: standard, mini, and micro. The SD card receptacle on the module is designed for a standard size card, which is the largest form factor. A passive adapter can be used to accommodate the smaller mini or micro sizes, if necessary.

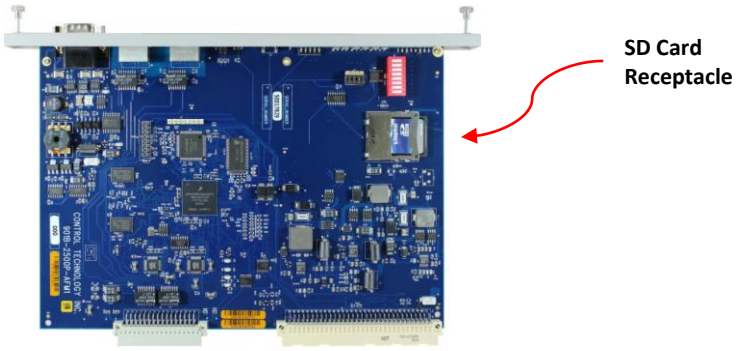
For most applications, the 4GB card is more than adequate. A larger capacity card (i.e. SDXC cards offer storage capacities to 256GB) is needed only when the SD card storage is used to maintain a number of extremely large files such as data log files. Smaller capacity cards can be used, but we recommend as a minimum that an SDSC card with 256MB be installed.

Another decisive factor for SD card selection is access speed. The SD card access speed is indicated by a class rating, which indicates the minimum continuous write speed. For example, a class rating of ④ indicates 4 MB/sec. As an alternative, the access speed may be provided by the “x” rating method as used for CD-ROM speed, where speed is indicated as a multiple of an audio CD (150 Kb/s). In this case, a class rating of ‘300x’ indicates 45MB/sec.

The speed of the SD card has a minimal effect on the time required for the module to start up or store an application program. However, a very fast SD card can considerably improve performance (up to 25-30%) when the program is executing a repetitive “write” operation, such as data logging.

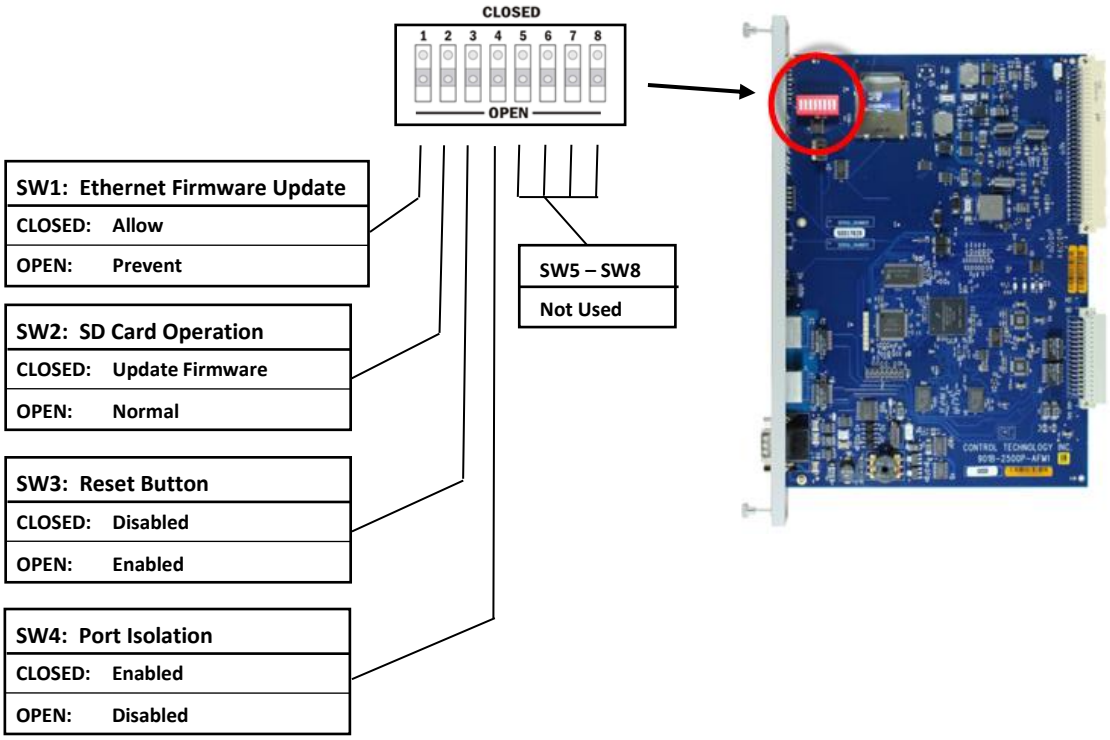
SD Card Installation

The module must be removed from the base to install the SD card. Insert the card in the receptacle face up, with the beveled edge facing the receptacle. Continue insertion until you hear a click, then release. To remove the card, apply insertion pressure until you hear a click, then release pressure.



3.1.2 Module Switches

The location and function of module user switches are shown below:



SW1: Ethernet Firmware Update

The position of this switch determines whether or not the module will permit the firmware update operation to be executed thru an Ethernet network connection to ***CTI Workbench***. When set to OPEN, the module will not accept a request to update firmware from ***CTI Workbench***. In this case, the firmware update file must be read from the embedded SD card using the method described in SW2. When set to the CLOSED position, the firmware update operation can be executed from a remote location thru a network connection to ***CTI Workbench*** or using the embedded SD card. See CHAPTER 9 for details on the firmware update operation.

SW2: SD Card Operation

This switch position determines the operation of the SD card at module startup. When SW2 is OPEN, the module starts up in the normal operating mode. When set to CLOSED position, the module attempts to read a firmware update file from the SD card and update the ACP1 module firmware. See Section 9.3 for details on the firmware update operation using the SD card method.

SW3: Reset Button

This switch controls the operation of the front panel reset switch. When OPEN, the module performs a "soft reset" and restarts the module when the Reset Button is depressed. If an application program resides on the SD card, it is automatically loaded into memory with operational state set by *Module Startup Mode* configuration setting (see Section 5.2). When CLOSED, the Reset Button is disabled, and the module does not reset when it is pressed.

SW4: Port Isolation

The position of this switch enables/disables isolation between the front panel Ethernet ports. When OPEN, isolation is disabled and Ethernet traffic on each is determined by the internal Ethernet switch. Broadcast messages received on either port are forwarded to the other port. When CLOSED, port isolation enabled. This setting blocks forwarding of all Ethernet packets between the two ports. This setting should be used when the ACP1 is used with redundant network topologies to prevent network communication loops.

SW5 – SW8: Reserved

These switches are reserved for future use.

3.1.3 Ethernet Cabling

The 2500P-ACP1 must be connected to the Ethernet network for configuration and program download. For best results you should use cables that are rated Category 5e.

3.1.4 Power Requirements

The CTI 2500P-ACP1 module consumes a maximum of 5 watts of +5 VDC power. To calculate the total power required for a base, you need to add the power requirements for the other modules you will install in the base. Ensure the power supply is capable of meeting power requirements for all installed modules.

3.2 Installing the 2500P-ACP1 module

3.2.1 Unpacking the Module

Open the shipping carton and remove the special anti-static bag that contains the module. Ensure you are properly grounded and have discharged any static buildup before removing the unit from the static bag. Do not discard the anti-static bag; use it for protection against static damage when the module is not inserted into the I/O base.

CAUTION

The components on the CTI 2500P-ACP1 printed circuit card can be damaged by static electricity discharge. To prevent this damage, the module is shipped in a special anti-static bag. Static control precautions should be followed when removing the module from the bag and when handling the printed circuit card during configuration.

3.2.1 Physical Installation

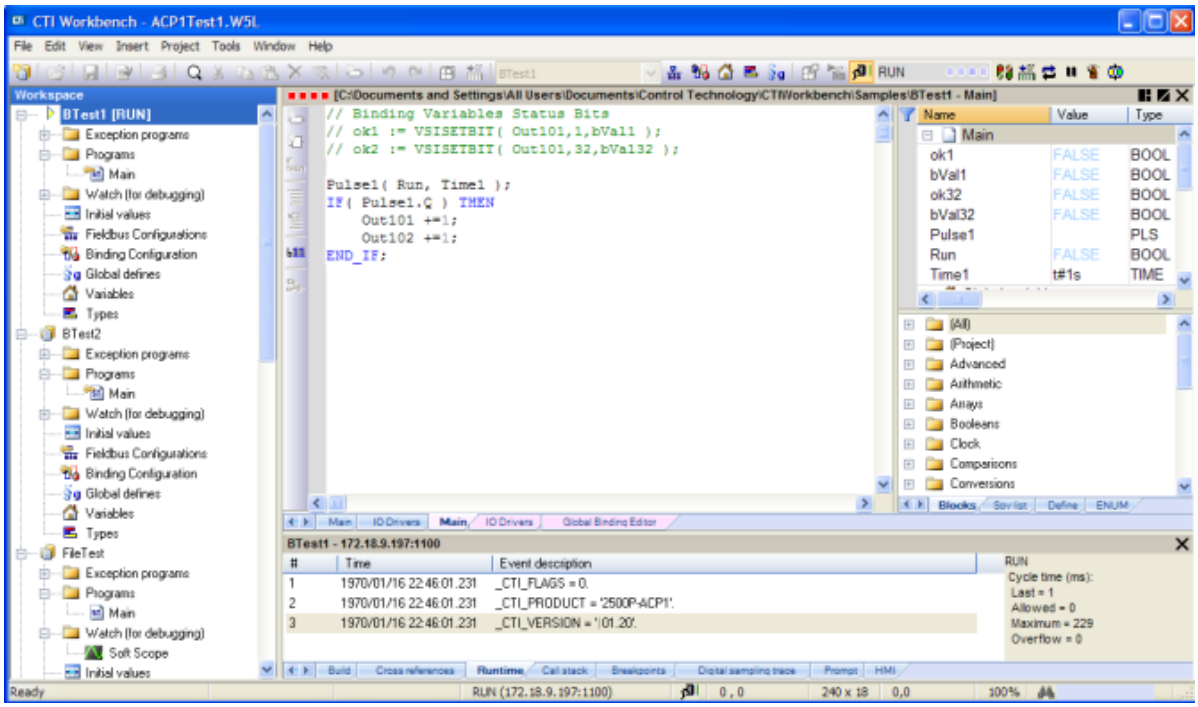
Before installing the module, remove AC power from the rack. Using the guides, align the circuit board with one of the I/O slot connectors in the base. Slide the 2500P-ACP1 module into the rack until the connector seats. Then use the thumbscrews to secure the module in the rack.

CHAPTER 4 USER INTERFACES

This section describes the methods supported in the 2500P-ACP1 for connection to user interface software. This includes connection of ***CTI Workbench*** Integrated Development Environment (IDE) software for program download, debug, and monitoring as well as connection to Web Server for module configuration and operational status.

4.1 CTI Workbench

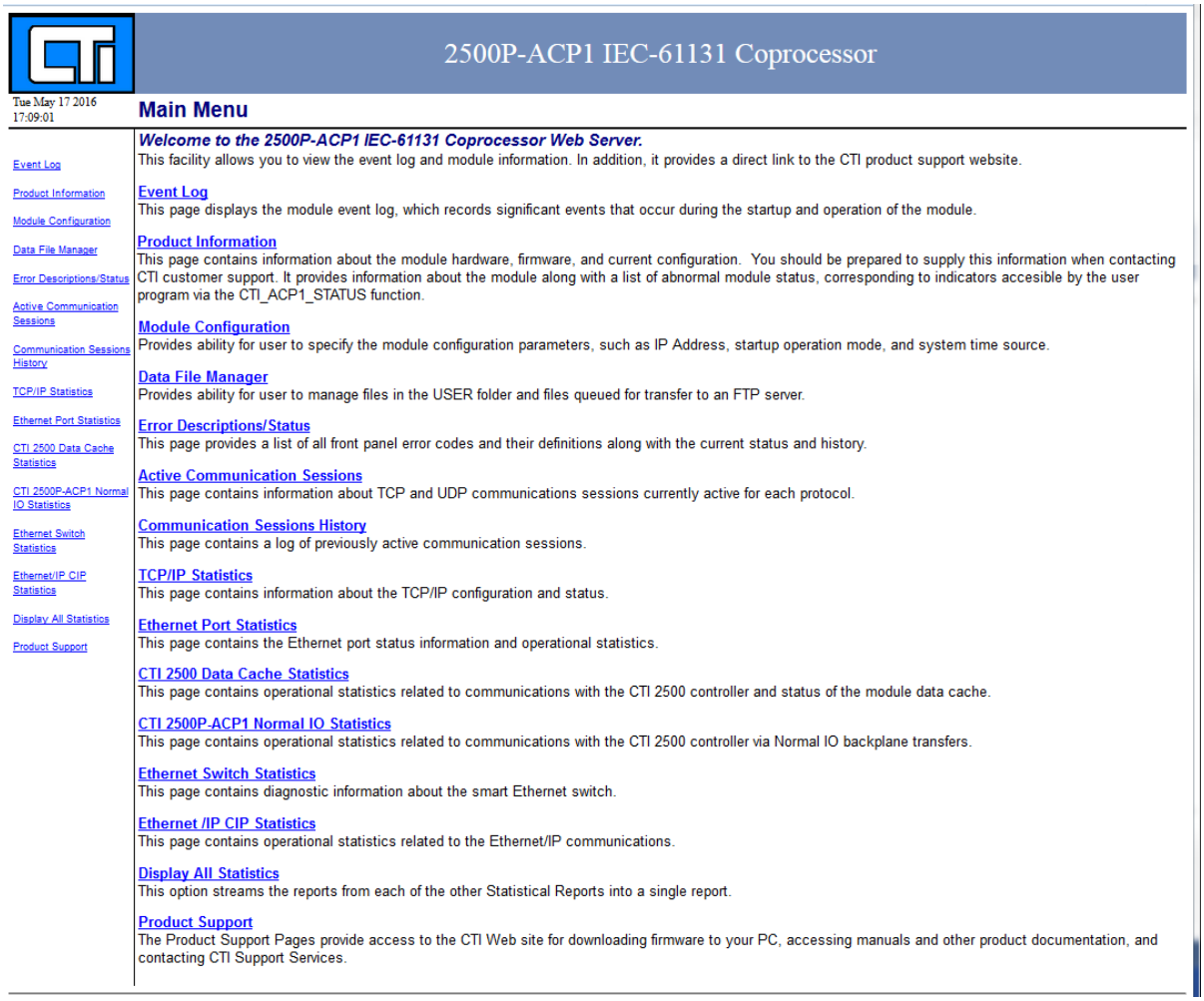
The 2500P-ACP1 includes a Server for Ethernet connection to ***CTI Workbench*** on Port 1100. The computer running ***CTI Workbench*** can connect to either of the front panel Ethernet ports. A maximum of 3 simultaneous ***CTI Workbench*** connections to the ACP1 module is allowed.



4.2 Web Server

The 2500P-ACP1 provides an embedded Web Server that can be accessed from any computer with an Ethernet connection to the module by entering the module IP address into the navigation toolbar of any web browser. The Web Server provides a simple method for module configuration (see Section 5.1.2) and access to product information, module event log, and statistical diagnostic data. A maximum of 3 users can be connected to the Web Server simultaneously.

The home page of the ACP1 Web Server is shown below:



CHAPTER 5 MODULE CONFIGURATION

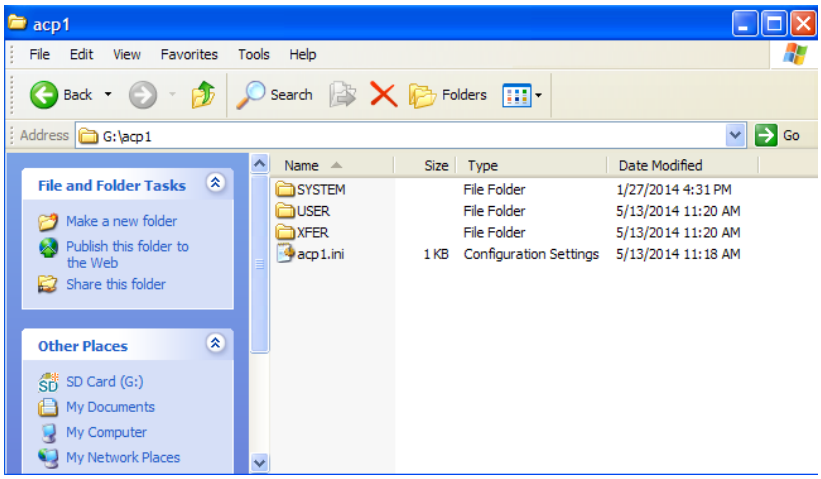
This section describes the steps required to configure the 2500-ACP1 module for operation.

5.1 Ethernet Network Parameters

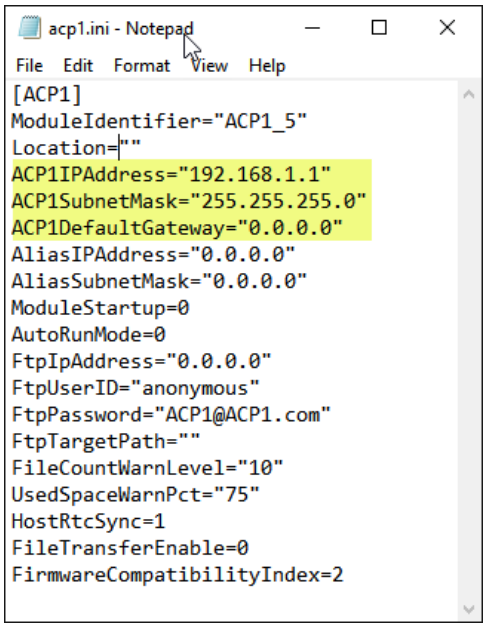
You must assign the Ethernet network settings for your 2500P-ACP1 module. An Ethernet connection is required for ***CTI Workbench*** and/or access to the embedded product web server. Two methods are possible for setting the module network configuration.

5.1.1 Method 1 – Module generates configuration file

a. Install the SD card as described in Section 3.1.1, and insert ACP1 module into the base.
b. Apply AC power so that the ACP1 module executes power-on reset.
c. When Status LED starts blinking, remove ACP1 module from the base.
d. Remove SD card from ACP1 and insert in PC card reader.
e. Open drive letter assigned to SD Card and navigate to \acp1 folder:



f. Open ‘acp1.ini’ file with a text editor such as ‘Notepad’ and modify highlighted fields to match your network settings. All other items can be configured via the Web Server (see Section 5.1.2).



g. Save the modified file and return SD card to ACP1.
h. Reinsert ACP1 module into base as described in Section 3.2.1.
i. New settings are detected on module power-up and saved in non-volatile memory.

5.1.2 Method 2 – Offline configuration

a. Remove SD card from ACP1 and insert in PC card reader.
b. Open a text editor such as ‘Notepad’ and copy/paste the following text. Modify the values next to keywords to match your network setup.
   All other items can be configured via the web server.

```
ACP1IPAddress="192.168.1.1"
ACP1SubnetMask="255.255.255.0"
ACP1DefaultGateway="0.0.0.0"
```

c. Select ‘Save As’ option; then select drive letter assigned to SD Card.
d. Press ‘Create New Folder’ icon and rename ‘New Folder’ as ‘acp1’.
e. Enter file name as ‘\acp1\acp1.ini’ and press [Save].
f. Reinsert ACP1 module into base as described in Section 3.2.1.
g. New settings are detected on module power-up and saved in non-volatile memory.

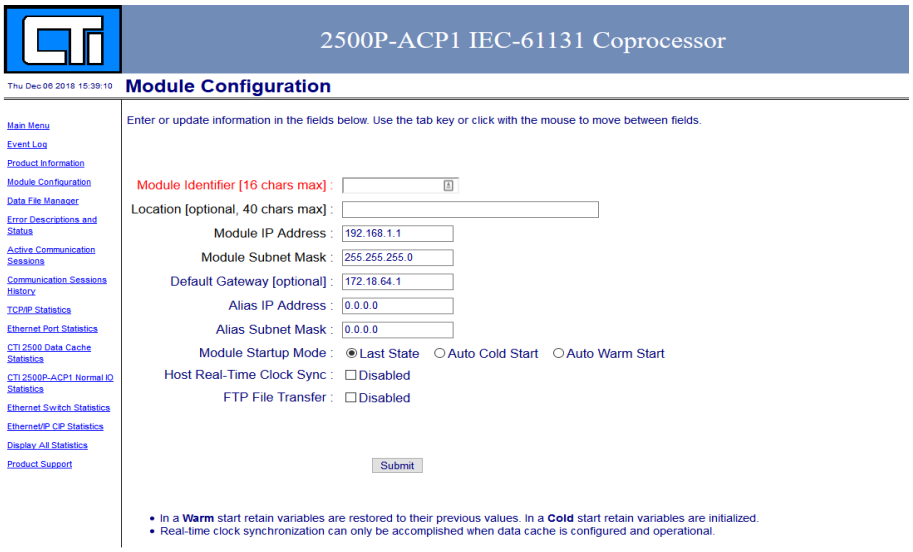
5.2 Module Configuration via Web Server

When the Ethernet Network Parameters have been setup (see Section 5.1), the embedded web server can be used to configure module parameters and operation.

a. Connect to the web server from any computer with an Ethernet connection to the module by entering the module IP address into the navigation toolbar of any web browser.

b. Select *Module Configuration* menu item as highlighted below:

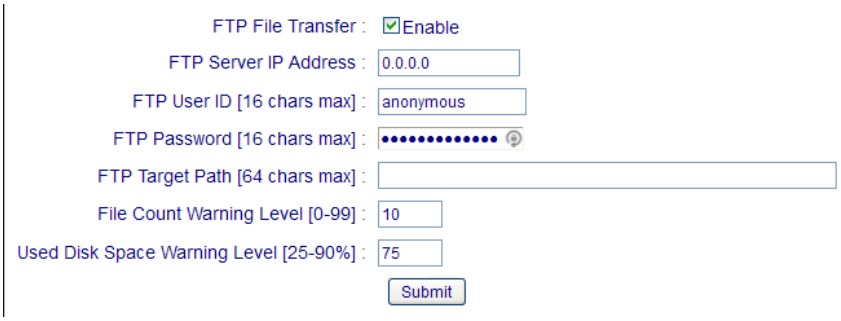
c. The *Module Configuration* screen is displayed.



- *Module Identifier* is a free- form text field with a maximum length of 16 characters. An entry in this field is required, and one or more characters must be entered.

  Note: If *Method 1* described in Section 5.1.1 was used to create the module configuration file, an entry in the Module Identifier field was automatically generated consisting of "ACP1-" + 9-digit module Serial Number. This entry can be edited as desired.

- *Location* is an optional text field with a maximum length of 40 characters. This field can provide additional information to uniquely identify the module if desired.
- *Module IP Address, Module Subnet Mask,* and *Default Gateway* display values currently stored in non-volatile memory (as provided by the acp1.ini configuration file on SD card). The assigned IP address is associated with both Ethernet ports. These values can be edited when required, but the changes do not take effect until the module is reset.
- *Alias IP Address and Alias Subnet Mask* display values currently stored in non-volatile memory (as provided by the acp1.ini configuration file on SD card). Entering these values enable IP aliasing so the ACP1 module is associated with two different IP addresses. Both IP addresses are associated with both Ethernet ports (i.e. the IP addresses are not port specific). These values can be edited when required, but the changes do not take effect until the module is reset.
- *Module Startup Mode* determines the operational state (STOP/RUN/PAUSE/ERROR) set immediately following a power-on reset. This assumes a valid application program is available on the SD card when the module is powered up. Otherwise, the module remains STOPPED.
  - *Last State*: The module operational state is set based on the run-time state when the module reset or lost power.
  - *Auto Cold Start*: The application program is loaded and set to RUN mode with all variables set to their initialization values.
  - *Auto Warm Start*: The application program is loaded and set to RUN mode with RETAIN variables restored to their last values and all other variables set to initialization values.
- *Host Real-Time Clock Sync*: This selection enables the synchronization Real-Time Clock (RTC) in the ACP1 module with the RTC in the Host Controller. Time synchronization is available only when the CTI Data Cache Interface is enabled (see Section 6.1).
- *FTP File Transfer*: This selection enables the embedded FTP Client service and associated tasks to transfer user data files created by the ACP1 application program to a remote FTP Server. This operation allows important system information to be saved to a network server while the ACP1 is executing. However, performing an *FTP File Transfer* while ACP1 is executing will extend the ACP1 cycle time up to 500 msec. The *FTP File Transfer* operation can be controlled programmatically (via ***CTI_XFER_FILE*** function) or manually via the Web Server. Enabling this feature displays additional keywords for configuration of the *FTP File Transfer* operations as shown below:



- *FTP Server IP Address*: IP Address of the device where the FTP Server is running. This will normally be a computer executing the FTP Server software.
- *FTP User ID*: User ID required to login to the specified FTP Server. If the FTP Server is setup to accept anonymous login, the default User ID “anonymous” can be used. A maximum length of 16 characters may be entered in this field.
- *FTP Password*: Password required for login to the specified FTP Server. Since many FTP Server require an email address as the password for anonymous FTP, the default Password “ACP1@ACP1.com” can be used for anonymous FTP.
- *FTP Target Path:* The directory path relative to the root directory set by the FTP Server. Directory folder names are separated by the slash character (/).For example: /acp1_T101/files. Path string is limited to a maximum length of 64 characters.

  If no path is entered, files are written to the root directory established by the FTP Server.
- *File Count Warning Level*: File count that triggers the *FTP Pending File Count Warning* (Error Code = 408) when exceeded. “File count” is the number of files in the SD Card /ACP1/XFER folder waiting for transfer to the FTP Server. This alarm is normally set to a level that indicates a problem with the connection to the FTP Server while the ACP1 file management is still operating normally. Valid range is 0-99 where an entry of ‘0’ generates the alarm whenever one or more files are waiting to transfer to the FTP Server and ‘99’ generates the alarm only when the FTP transfer queue is completely full (100 files).
- *Used Disk Space Warning Level*: Percent of used SD Card disk space that triggers the *SD Card Free Space Warning* (Error Code = 398). This alarm is normally set to a level that indicates a potential problem with the amount of free disk space on the installed SD Card. Valid range is 25-90 entered as the “percentage of used disk space on SD Card”. The alarm is generated when the ratio of used disk space to available space exceeds value entered.

d. Press [Submit] to save changes to the module configuration. If Ethernet network settings have changed, module must be reset (or power cycled) to use the new parameter values.

5.3 2500P-ACP1 Firmware Update

Although CTI installs the latest available firmware version prior to shipping a product, it is possible that a new firmware version has been released since your ACP1 module was shipped. We recommend that you check the firmware versions and download the latest firmware update file for the 2500P-ACP1 from the CTI web site, http://www.controltechnology.com/downloads/ .

The methods for the firmware update process are described in CHAPTER 9.

CHAPTER 6 HOST PLC INTERFACES

Although the 2500P-ACP1 can operate as an independent controller, it normally functions as a coprocessor that enhances a PLC-based control system by providing high-performance data computations and communication functions with data transferred to/from the system controller.

Two different PLC interfaces are included that provide real-time communication paths between the ACP1 module and the Host controller. The PLC interface to be used with the application program is specified via software configuration using ***CTI Workbench***.

6.1 CTI Data Cache Interface

The CTI 2500 Data Cache interface is a proprietary communications method designed to produce high-performance data transfer between the Host controller and 2500P-ACP1 while minimizing the workload on the PLC. This interface utilizes a proprietary, highly efficient protocol on a high-speed Ethernet channel that provides a significant increase in amount of data and PLC memory access not available via I/O backplane.

The CTI 2500 Data Cache interface can be configured to access up to 4096 different PLC memory locations, and all 505 memory types are supported (including Drum, Loop, and Alarm variables). For most applications, all data is transferred each PLC scan. The only exception is the rare case where PLC scan time is less than the ACP1 execution cycle. In this case, the data transfer occurs once per ACP1 execution cycle.

The CTI 2500 Data Cache interface is available when the 2500P-ACP1 is used with a CTI 2500 Series® PLC installed as the system controller. An Ethernet connection between the 2500P-ACP1 and PLC is required, and an external connection must be added when using all existing Siemens Series 505® and CTI 2500 Series® bases.

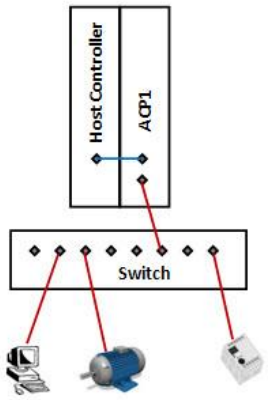
The Data Cache Interface works successfully only when the following conditions exist:

1. The CTI 2500 Series® controller must be running a firmware V7.05 or later to support this interface. The installed firmware version may be determined by accessing the PLC Product Information web page or by reading Status Words 260-261. Since controller enhancements and bug fixes are on-going, we recommend that the controller firmware be upgraded to the latest release. A firmware update file can be downloaded from the CTI web site at http://controltechnology.com/downloads/ .
2. The ACP1 module and the Host PLC must be on the same IP network, as determined by the module IP address and Network Mask. See *APPENDIX B: IP ADDRESS INFORMATION* for additional information regarding IP address selection.
3. The ACP1 module and the Host PLC must be on the same Ethernet LAN or VLAN if used.

6.1.1 CTI Data Cache Connection Options

Direct Connection to the Host PLC

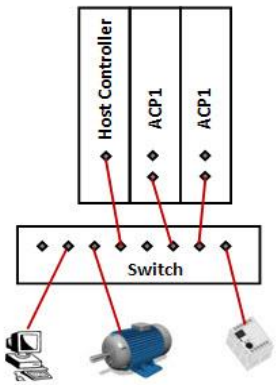
When a single CTI Advanced Function Module (2500P-ACP1 or 2500P-ECC1) is installed in the system and can be placed in the same base as the Host PLC, the preferred Data Cache Interface connection is a short Ethernet cable (Cat 5 or Cat 5e) from one of the Ethernet ports on the 2500P-ACP1 to the Ethernet port on the 2500 Series® CPU. A second Ethernet cable is then connected from the other port to a network switch that provides a path to the other devices.



This direct connection method utilizes the embedded switch in the ACP1 module to “redirect” all Ethernet traffic from the remote devices to the CPU without affecting the performance of the 2500P-ACP1. An added advantage of this method is that the embedded switch provides broadcast storm protection for the 2500 Series® programmable controller. Port Isolation must be disabled when using this method.

Alternately, you may choose to connect both the ACP1 module and the Host Controller to the network Ethernet switch. See the following section.

Connection via a Network Switch



If you are connecting more than one CTI Advanced Function Module to the same Host PLC, you should make all Ethernet connections through an external Ethernet switch as shown in the figure on the left.

Making the Ethernet connections by “daisy chaining” cables between modules could cause an interruption of network communications whenever one of the modules is powered down or reset.

CAUTION:

When using this method, both Ethernet ports should be connected to the network ONLY if Port Isolation is enabled by setting User Switch 4 (SW4) to CLOSED position. Connecting both ports to the network without enabling Port Isolation could result in a continuous communications loop and disrupt all network communications.

6.1.2 Data Cache Configuration

The Data Cache interface is enabled by including the ***CTI 2500 Data Cache*** fieldbus driver in the application program and configuring at least one element. A maximum of 4096 elements may be configured.

The Data Cache element is configured by specifying three parameters:

- PLC Memory Area (Memory Type and Data Type)
- Cache Direction (Read from PLC, Write to PLC, or Read and Write)
- Corresponding ACP1 Application Program Variable

PLC memory location is designated by *Memory Type* and *Address*. A "block" of PLC memory locations may be created by entering a *Starting Address* and *Offset*. Since it is possible to store different data types (such as Integer or Real) into some PLC memory locations, the configuration must include the data representation or "data type" of the value to be transferred.

PLC Memory Areas

The PLC Memory Areas and associated PLC data types that can be accessed via CTI Data Cache interface are listed below:

Common PLC Memory Area	Data Type(s)
V Memory (V)	INT
V Memory (VL)	DINT
V Memory (V.)	REAL
K Memory (K)	INT
K Memory (KL)	DINT
K Memory (K.)	REAL
Discrete Input (X)	BOOL
Discrete Output (Y)	BOOL
Control Relay (C)	BOOL
WX Memory (WX)	INT
WY Memory (WY)	INT
Status Word (STW)	INT
Timer-Counter Preset (TCP)	INT
Timer-Counter Current (TCC)	INT

Drum Variables	Data Type
Drum Step Preset (DSP)	INT
Drum Step Current (DSC)	INT
Drum Count Preset (DCP)	INT
Drum Count Current (DCC)	INT

Loop Variables	Data Type
Gain (LKC.)	REAL
Reset Time - min (LTI.)	REAL
Rate Time – min (LTD.)	REAL
Sample Rate – sec (LTS.)	REAL
Process Variable (LPV.)	REAL
Process Variable (LPV)	INT
PV High Limit (LPVH.)	REAL
PV Low Limit (LPVL.)	REAL
Set Point (LSP.)	REAL
Set Point (LSP)	INT
SP High Limit (LSPH.)	REAL
SP High Limit (LSPH)	INT
SP Low Limit (LSPL.)	REAL
SP Low Limit (LSPL)	INT
Output - % (LMN.)	REAL
Output (LMN.)	INT
Bias (LMX.)	REAL
Bias (LMX)	INT
Error (LERR.)	REAL
Error (LERR)	INT
High-High Alarm Limit (LHHA.)	REAL
High-High Alarm Limit (LHHA)	INT
High Alarm Limit (LHA.)	REAL
High Alarm Limit (LHA)	INT
Low Alarm Limit (LLA.)	REAL
Low Alarm Limit (LLA)	INT
Low-Low Alarm Limit (LLLA.)	REAL
Low-Low Alarm Limit (LLLA)	INT
Alarm Deadband (LADB.)	REAL
Alarm Deadband (LADB)	INT
Orange Dev Alarm Limit (LODA.)	REAL
Orange Dev Alarm Limit (LODA)	INT
Yellow Dev Alarm Limit (LYDA.)	REAL
Yellow Dev Alarm Limit (LYDA)	INT
Rate of Change Alarm Limit (LRCA.)	REAL
Alarm Acknowledge Flags (LACK)	INT
Derivative Gain Limiting Coeff (LKD.)	REAL
V-Flags (LVF)	INT
Control Flags – MSW (LCFH)	INT
Control Flags – LSW (LCFL)	INT
Ramp/Soak Status Flags (LRSF)	INT
Ramp/Soak Step Number (LRSN)	INT

Alarm Variables	Data Type
Sample Rate – sec (ATS.)	REAL
Process Variable (APV.)	REAL
Process Variable (APV)	INT
PV High Limit (APVH.)	REAL
PV Low Limit (APVL.)	REAL
Set Point (ASP.)	REAL
Set Point (ASP)	INT
SP High Limit (ASPH.)	REAL
SP High Limit (ASPH)	INT
SP Low Limit (ASPL.)	REAL
SP Low Limit (ASPL)	INT
Error (AERR.)	REAL
Error (AERR)	INT
High-High Alarm Limit (AHHA.)	REAL
High-High Alarm Limit (AHHA)	INT
High Alarm Limit (AHA.)	REAL
High Alarm Limit (AHA)	INT
Low Alarm Limit (ALA.)	REAL
Low Alarm Limit (ALA)	INT
Low-Low Alarm Limit (ALLA.)	REAL
Low-Low Alarm Limit (ALLA)	INT
Alarm Deadband (AADB.)	REAL
Alarm Deadband (AADB)	INT
Orange Dev Alarm Limit (AODA.)	REAL
Orange Dev Alarm Limit (AODA)	INT
Yellow Dev Alarm Limit (AYDA.)	REAL
Yellow Dev Alarm Limit (AYDA)	INT
Rate of Change Alarm Limit (ARCA.)	REAL
Alarm Acknowledge Flags (AACK)	INT
V-Flags (AVF)	INT
Control Flags – MSW (ACFH)	INT
Control Flags – LSW (ACFL)	INT

Cache Direction

An ACP1 application program variable is assigned or “mapped” to that PLC memory location. Variables mapped into the Data Cache are designated as one of the following:

Cache Direction	Description of Operation
Read from PLC	PLC memory value is copied to corresponding ACP1 variable each scan.
Write to PLC	PLC memory location is updated whenever the corresponding ACP1 variable value changes.
Read and Write	PLC memory value is copied to the corresponding ACP1 variable each scan. If the variable value is changed by the application program logic, then the new value is copied to the corresponding PLC memory location.

ACP1 Application Program Variables

An ACP1 variable must be “mapped” or associated with each PLC memory location to be included in the Data Cache interface. Each variable has the following configuration properties:

- The *Data Type* of each ACP1 variable mapped to a PLC address must match the D*ata Type* for the PLC Memory Area assigned by the data block. For instance, WX-Memory must map to an ACP1 INTEGER variable, and V-Memory REAL value must map to an ACP1 REAL variable.
- *Offset* specifies the zero-relative element position in the data block where that variable is mapped. In this case, *Offset* is equivalent to an array position index and NOT a “PLC Word” offset.
  For instance, a V-Memory REAL Memory Area is inserted with Starting Address of 21. In this case, the first variable (Offset = 0) is mapped to first PLC memory address positions that can hold the REAL value: V21-V22. The second variable (Offset = 1) is mapped to the next PLC memory positions for a REAL value: V23-V24. Likewise, offset 2 is mapped to V25-26; offset 3 to V27-28, etc.
- This “variable element” offset rule applies for all memory types. The ACP1 module firmware correlates variable data type to PLC memory requirements and adjusts memory offsets as required.
- It is not required that ACP1 variables be mapped into consecutive positions in the data block.
  For example, a Discrete Input (X) Memory Area is inserted with Starting Address of 65. The first variable is mapped to Offset = 0 (corresponding to X65). The next variable is mapped to Offset =3 (corresponding to X68). PLC addresses X66-X67 are not included in the Data Cache operation.
- Two variables cannot be mapped to the same PLC memory address.
- The total number of ACP1 variables that can be mapped to the Data Cache is limited to 4096.

CTI 2500 Data Cache Operation

When the ***CTI 2500 Data Cache*** interface is configured, the ACP1 module establishes the connection to the Host PLC and starts data transfer immediately when an ACP1 application program enters RUN mode. If a communications error is detected (such as mismatched IP Address or cable problem), the ACP1 *Host PLC Data Cache Status* error occurs. The problem must be corrected to clear this error.

The CTI 2500 Data Cache also supports time synchronization with the PLC clock when ***Host Real-Time Clock Sync*** is enabled on the Module Configuration page of the ACP1 web server. *When enabled, the time in the* ACP1 Real-Time Clock is synchronized to the PLC clock when the Data Cache connection is initialized and approximately every two hours. This ensures the RTC date/time in both devices are always within one second of each other.

The status of the CTI 2500 Data Cache operation can be monitored by program logic or via the web server *CTI 2500 Data Cache Statistics* page.

CAUTION:

Although very unlikely, it is possible to create an application program and Data Cache configuration that can result in situations where some ACP1 variable "change of state" values are lost (and not written to the PLC) if all variables simultaneously change value during a single PLC scan.

For applications with rapidly changing variables that are written to the PLC (i.e. 'Write to PLC' and/or 'Read and Write'), it is recommended to use the CTI_ACP1_STATUS function in your application program to monitor the Data Cache operation.

Status Bit 3 is set ON when a Data Cache Transfer Error occurs that may indicate variable data was not written to PLC on change of value in ACP1. If this error is detected, contact CTI Technical Support on how to modify your application program and/or Data Cache configuration to prevent this condition.

6.2 CTI 2500P-ACP1 I/O Interface

The CTI 2500P-ACP1 I/O Interface provides data transfer along the chassis backplane between the 2500P-ACP1 and chassis controller. This CTI 2500 I/O interface works with all existing Siemens Series 505® and CTI 2500 Series® chassis, PLCs, and Remote Base Controllers (RBC).

This interface allows the 2500P-ACP1 to emulate a standard I/O module so that the PLC transfers image register data to/from the module each PLC scan. The ACP1 differs from standard I/O modules in that it supports several different I/O configurations that can be set by software configuration.

This feature is enabled by including the ***CTI 2500P-ACP1 I/O*** fieldbus driver in the application program and configuring the I/O data transfer parameters

The 2500P-ACP1 I/O interface is configured by specifying the following parameters:

- 2500P-ACP1 I/O Definition
- PLC I/O Module Definition and Login Address for slot where 2500P-ACP1 is installed
- ACP1 Application Program Variables

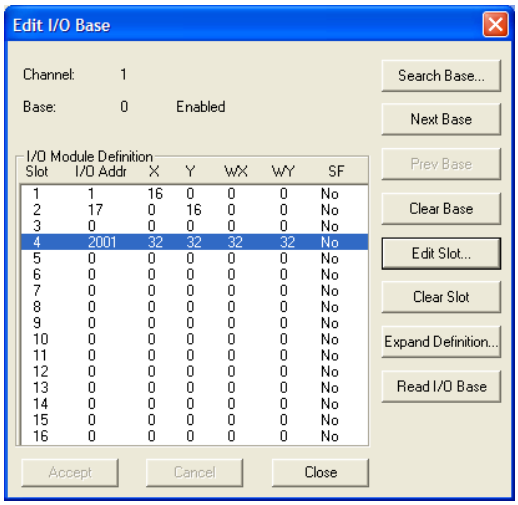
2500P-ACP1 I/O Definition

The ACP1 I/O definition is specified by selecting one the following I/O configurations in the *CTI 2500P-ACP1 I/O Configuration Wizard* provided in ***CTI Workbench***:

- Discrete I/O: 32 inputs / 32 outputs (32X/32Y)
- Word I/O: 32 inputs / 32 outputs (32WX/32WY)
- Mixed I/O: 32 discrete inputs/outputs and 32 word inputs/outputs (32X/32Y/32WX/32WY)

PLC I/O Module Definition and ACP1 Module Login Address

The *PLC I/O Module Definition* must match the I/O Configuration selected in the ACP1 application program. The following example shows the I/O definition required for an ACP1 module installed in Slot 4 configured with a Mixed I/O (32X/32Y/32WX/32WY) interface.



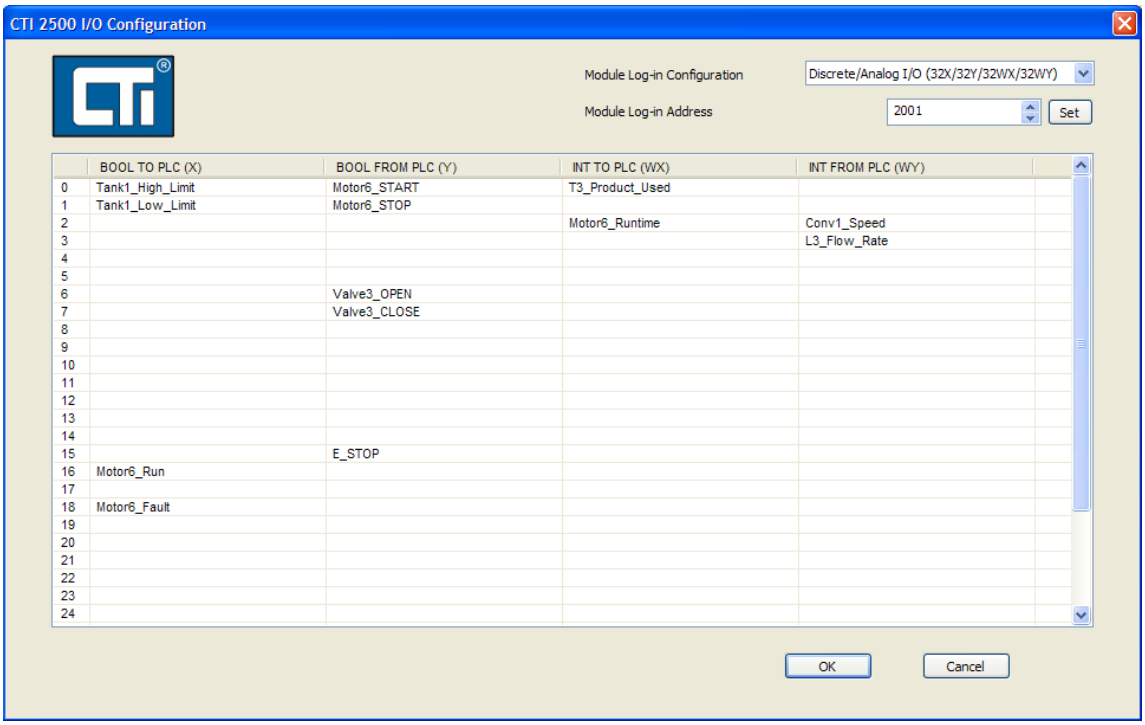
The Mixed I/O interface requires special care when assigning an *I/O Address* because the Series 505® model allows one "login" address for each module slot. Therefore, the *I/O Address* assigns the image register positions for both the Discrete I/O and Word I/O values.

In the example, a value of "2001" is designated as the I/O Address. This equates to the following I/O mapping for ACP1 data within the PLC:

- 32 discrete inputs mapped to X2001-X2032
- 32 discrete outputs mapped to Y2033-Y2064
- 32 word inputs mapped to WX2065-WX2096
- 32 word outputs mapped to WY2097-WY2124

ACP1 Application Program Variables

ACP1 variables must be "mapped" or associated with each PLC memory location to be included in the ACP1 I/O interface. Variables are specified within the same *CTI 2500P-ACP1 I/O Configuration Wizard*. The following screen shot shows an example of how the ACP1 variables are mapped to PLC I/O image registers.



Rules for mapping ACP1 variables into 2500P-ACP1 I/O interface:

- The *Data Type* of each assigned variable must match the data type for the corresponding I/O image register. For instance, Variables mapped to the Discrete (X/Y) image register must be BOOL data type, and variables mapped to the Word (WX/WY) image register must be signed integer (INT) data type.
- *Offset* position listed in the left-hand column specifies the zero-relative position in the image register data buffer for each I/O data element.

  The example shows the *Module Log-in Configuration* of 32X/32Y/32WX/32WY and *Module Log-In Address* = 2001 (which matches the ***PLC I/O Module Definition*** shown above). This results in the following ACP1 variable assignments:

  "Tank1_High_Limit" is assigned to PLC Discrete Input Offset 0 (X2001) – must be BOOL
  "Tank1_Low_Limit" is assigned to PLC Discrete Input Offset 1 (X2002) – must be BOOL
  "Motor6_Run" is assigned to PLC Discrete Input Offset 16 (X2017) – must be BOOL
  "Motor6_Fault" is assigned to PLC Discrete Input Offset 18 (X2019) – must be BOOL
  "Motor6_START" is assigned to PLC Discrete Output Offset 0 (Y2033) – must be BOOL
  "Motor6_STOP" is assigned to PLC Discrete Output Offset 1 (Y2034) – must be BOOL
  "Valve3_OPEN" is assigned to PLC Discrete Output Offset 6 (Y2039) – must be BOOL
  "Valve3_CLOSE" is assigned to PLC Discrete Output Offset 7 (Y2040) – must be BOOL
  "E_STOP" is assigned to Discrete PLC Output Offset 15 (X2048) – must be BOOL
  "T3_Product_Used" is assigned to PLC Word Input Offset 0 (WX2065) – must be INT
  "Motor6_Runtime" is assigned to PLC Word Input Offset 2 (WX2067) – must be INT
  "Conv1_Speed" is assigned to PLC Word Output Offset 2 (WY2099) – must be INT
  "L3_Flow_Rate" is assigned to PLC Word Input Offset 3 (X2100) – must be INT

- It is not required that ACP1 variables be mapped into all Discrete and/or Word image registers positions. Only those positions containing data relevant to ACP1 operation need to be assigned.
- The maximum number of ACP1 variables that can be mapped to the 2500P-ACP1 I/O interface is provided by the Mixed I/O model which supports PLC Output of 32 BOOL/32 INT and PLC Input of 32 BOOL/32 INT values.

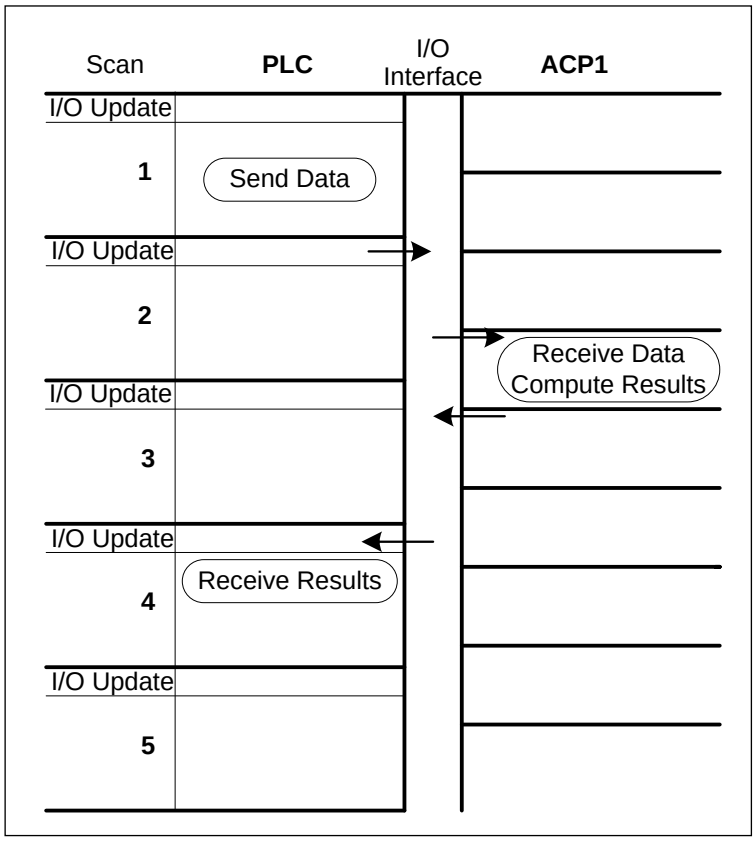
2500P-ACP1 I/O Operation

When the ***PLC Module I/O Definition*** and ***ACP1 Login Address*** have been configured in the Host PLC, it immediately attempts to transfer I/O data with the ACP1. The Host PLC will detect an I/O Mismatch error with the assigned ACP1 module slot until an application program containing a *2500P-ACP1 I/O Configuration* that matches the *PLC Module I/O Definition* is downloaded to the ACP1 module to enable the I/O data transfer. The *I/O Mismatch* error sets the appropriate bit in STW11-STW26 based on the configured module slot and writes all configured (X/WX) inputs for that slot to zero.

The I/O data transfer starts immediately when an ACP1 application program enters RUN mode. If the configurations in the PLC and ACP1 do not match, the ACP1 *Host PLC I/O Data Exchange Timeout* error will be generated. The I/O configuration mismatch must be corrected to clear this error.

The status of the 2500P-ACP1 I/O operation can be monitored by program logic or via the web server ***CTI 2500 ACP1 I/O Statistics*** page.

Because the ACP1 application program runs asynchronous to the PLC scan cycle, methods are implemented to ensure data consistency is maintained for the entire data set passed to/from the PLC. The ACP1 firmware “clocks” the data through intermediate storage buffers within the backplane interface sub-system to complete I/O data transfer between PLC and ACP1 as shown below:



CHAPTER 7 DEVICE COMMUNICATIONS

The 2500P-ACP1 module supports several protocols for communicating with remote devices. This section describes each protocol and provides an overview of its functionality. Connections to ***CTI Workbench*** and embedded Web Server are always active when the module is powered on. All other "Fieldbus Protocols" are enabled by inserting the appropriate protocol template into an application program, configuring its operation, and downloading the project to the ACP1 module. Multiple protocols can be executed concurrently within the restrictions described below Data transferred to/from the remote devices within the protocol message packets are controlled by the logic in the application program.

7.1 Ethernet Communications

The 2500P-ACP1 limits the number of Ethernet connections allowed to external devices in order to provide reliable and efficient communications. The following table describes how these connections are counted:

Connection Type	Maximum Concurrent Connections	ACP1 Connections Included in Count
Client	40 fieldbus connections	CAMP Client (TCP or UDP) Open Modbus Client (TCP or UDP) Variable Binding Subscriber (TCP) EtherNet/IP Scanner/Client (TCP)
Server	30 fieldbus connections	Open Modbus Server (TCP) Variable Binding Publisher (TCP) EtherNet/IP Adapter/Server (TCP)
Server for CTI Workbench	Supports 3 concurrent users	Connects on Port 1100 only
Web Server	Supports 3 concurrent users	Connects on Port 80 only
CTI 2500 Data Cache	1	Connection to CTI 2500 Series® CPU as Host Controller Provides access to all PLC memory types
Custom Client/Server applications using TCP/UDP functions	10	These client/server sockets are in addition to the 40 Client and 30 Server fieldbus connections

7.1.1 CAMP Client

The CAMP Client protocol provides a common means to communicate with CTI 2500 Series® controllers and Siemens Series 505® controllers via a TCP/IP network. When communicating with a Siemens 505® controller, the network connection must through a CTI 2572 or 2572-A/B Ethernet TCP/IP module or another 2500P-ACP1 that uses the CTI 2500-ACP1 I/O backplane interface to the Siemens controller.

Communications to another CTI 2500 Series® CPU can be accomplished via CTI 2572 or 2572-A Ethernet TCP/IP module, the 2500P-ECC1 Ethernet Communications Coprocessor, another 2500P-ACP1 module, or directly to the Ethernet port on the 2500 Series® controller. Each CAMP Client connection to one of these products counts against the limit of 40 client connections supported by the 2500P-ACP1.

The CAMP Client enables you to read or write a block of up to 256 V-memory locations in the specified controller. Messages can be automatically initiated on a user specified time interval, change in value of a designated variable (i.e. PLC memory location), or triggered by logic in the ACP1 module. Message data is mapped directly to/from variables in the application program.

UDP Multicast can be used to produce data that can be consumed by multiple local area network hosts that are configured to receive data on the multicast address.

NOTE:
The CTI 2500 Series® CPU Ethernet port does not support UDP or UDP Multicast.

The ACP1 Web Server provides tools for monitoring and debug. The status of connections to the CAMP Client can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the CAMP Client are listed in *APPENDIX A* under *CAMP Client Error Codes*.

7.1.2 Open Modbus Client

The Open Modbus Client provides a means to communicate with products implementing a Modbus TCP server that complies with specifications published by the Modbus Organization (http://www.modbus.org/). The Open Modbus TCP protocol is an adaptation of the popular Modbus RTU protocol network version of the Modbus RTU that allows it to communicate using TCP/IP.

Since the Modbus protocol is widely supported by a wide variety of PLCs, automation controllers, monitoring equipment, remote terminal units, and other industrial devices, it offers a common method for the exchanging data between devices from different manufacturers. In addition, the Modbus protocol is often used to communicate with process control I/O devices, such as motor starters and variable frequency drives.

The Open Modbus Client allows you to establish TCP connections to server devices that comply with the Open Modbus standard. Alternately, you can use UDP to communicate with Modbus devices that support this IP protocol. If the server device is a Modbus Ethernet to Serial Gateway, the Open Modbus Client can communicate with serial Modbus slave devices attached to the gateway. Each Modbus Client connection to one of these products counts against the limit of 40 client connections supported by the 2500P-ACP1.

The Open Modbus Client can initiate requests using the Modbus Function Codes listed in the table below.

Function Code	Description
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

The ACP1 web server provides tools for monitoring and debug. The status of connections to the Open Modbus Client can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the Open Modbus Client are listed in the *Protocol Error Codes* in *APPENDIX A*.

7.1.3 Open Modbus Server

The Open Modbus Server provides a common means for a wide variety of automation equipment that support the Modbus protocol to communicate with the ACP1 module. You can configure one or more blocks of Modbus data for the Modbus data types shown in the table below.

Modbus Data Type	Description
Coils	Read Coils
Holding Registers	Read Holding Registers
Discrete Inputs	Read Discrete Inputs
Input Registers	Read Input Registers

By mapping a block of Modbus addresses to a block of variables in the application program running in the 2500P-ACP1, external devices acting as Modbus TCP Clients can access specific data as needed. This data can be values from the Host PLC memory, data from another remote device connected to the ACP1, or internal variable data created by the ACP1 application program.

The Open Modbus Server is capable of servicing the Modbus Function Codes indicated in the table below.

Function Code	Description
01	Read Coils
02	Read Discrete Inputs
03	Read Holding Registers
04	Read Input Registers
05	Write Single Coil
06	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

Each connection by a remote Modbus TCP Client to the Open Modbus Server counts against the limit of 30 server connections supported by the 2500P-ACP1. The Open Modbus Server does not support UDP connections.

The ACP1 Web Server provides tools for monitoring and debug. The status of connections to the Open Modbus Server can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the Open Modbus Server are listed in the *Protocol Error Codes* in *APPENDIX A*.

For more information regarding Open Modbus go to the Modbus Organization website, http://www.modbus.org/

7.1.4 Network Data Exchange Publish and Subscribe

The Network Data Exchange protocol (or Variable Binding) provides an efficient means to exchange real-time data between devices connected to an Ethernet network. Variable Binding uses an event-based Publish-Subscribe model to provide high performance and low network traffic by transmitting variable data only when the data value is changed. Each compatible device can publish data and/or subscribe to any data transmitted by others. Multiple devices can subscribe to the same published variable. Status of the TCP connection between each Publisher and Subscriber is provided. Variable Binding network data exchange is currently supported by 2500P-ACP1, 2500P-ECC1, and 2500 Series® HMI products.

When performing the function of a Publisher, the 2500P-ACP1 continually monitors all items specified as published variables and transmits the value of a particular data item to subscribers only when the variable value changes. When acting as a Subscriber, the 2500P-ACP1 receives the published data to which it has subscribed and updates the value and date/time stamp of each corresponding variable. The variables can then be used in application program logic and/or written to any device attached to the ACP1 via serial or Ethernet connection. Additional variables can be added to provide Error Status to program logic.

When Variable Binding is configured, each device that subscribes to data published by the ACP1 module is counted against the limit of 40 client connections. Likewise, when the ACP1 module subscribes to data published by another device, the connection to that device counts against the limit of 30 server connections. The ACP1 Web Server provides tools for monitoring and debug. The status of Network Data Exchange connections can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by a Network Data Exchange subscriber are listed in *APPENDIX A: ERROR CODES*.

7.1.5 EtherNet/IP Scanner

The EtherNet/IP (EIP) Scanner provides a client driver for exchanging data with CIP nodes via CIP implicit messages, also called I/O messages. These messages are used for high-speed transfer of I/O assemblies between CIP devices in applications where the same set of data is transmitted repetitively.

I/O assemblies are referenced in terms of the CIP network controller or I/O Scanner. I/O data transmitted by the I/O Scanner to the Adapter is called O->T (Originator to Target) or Output assembly and I/O data received from the I/O Adapter is called T->O (Target to Originator) or Input assembly. One Output assembly and one Input assembly must be specified for each I/O Adapter (or Server).

The EIP I/O Scanner is configured by specifying each external CIP node (IP Address), the data to be transmitted (T->O assembly object), the data to be received from that node (O->T assembly object. The parameters for each assembly object includes the Assembly Instance number, Requested Packet Interval (RPI) or maximum time between packets, Data size (in bytes), and Connection type (point-to-point or multicast). Data is transferred with the application program by mapping variables to positions in the I/O assemblies.

An optional Configuration Instance can be configured when required by the I/O Adapter. Up to 400 bytes of configuration data can be specified using the built-in grid editor.

The EIP I/O Scanner interface can be used to control drives, motor starters, and other process control equipment provided with an EIP I/O Adapter interface. Each I/O Adapter provides one or more pre-defined Input-Output (I/O) assemblies, each consisting of a collection of device parameters and/or settings. The setup configuration of the EIP I/O Adapter includes assignment of IP Address and selection of the I/O assemblies to be used. Each connection to a remote EIP I/O Adapter counts against the limit of 40 client connections supported by the 2500P-ACP1.

In firmware V4.0 and later, the ACP1 allows the EIP I/O Scanner to communicate via Ethernet/IP Explicit Messages to read and/or write data objects from an EIP Server/Adapter “on demand” rather than cyclically scheduled I/O messages. Explicit messaging is more flexible than I/O messages in terms of what data or services are accessed in the Adapter since all I/O data must be pre-defined. The function blocks used for EIP Explicit Messaging are detailed in the CTI Workbench online help.

The ACP1 web server provides tools for monitoring and debug. The status of connections to the Ethernet/IP Scanner can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the Open Modbus Server are listed in the *Protocol Error Codes* in *APPENDIX A*.

For more information regarding EtherNet/IP and Common Industrial Protocol (CIP), go to the ODVA website, http://www.odva.org/

7.1.6 EtherNet/IP Adapter

The EtherNet/IP (EIP) Adapter provides a server interface for exchanging data with EIP clients via CIP implicit I/O messages and/or CIP Generic (explicit) messages. When configured as an I/O Adapter, a remote EIP I/O Scanner can establish an I/O connection to the 2500P-ACP1 and transfer data via I/O messages.

I/O message data is determined by the I/O assembly configurations. The I/O assemblies are referenced in terms of the CIP network controller or I/O Scanner. I/O data transmitted by the I/O Scanner to the Adapter is called O->T (Originator to Target) assembly and I/O data received from the I/O Adapter is called the T->O (Target to Originator) assembly.

The I/O assemblies can be configured as desired with data specified by mapping application program variables to each assembly. Each connection from a remote EIP I/O Scanner counts against the limit of 30 server connections supported by the 2500P-ACP1.

The EIP Adapter also can be configured to allow the 2500P-ACP1 to respond to general purpose CIP Generic messages issued by an external EIP Generic Message client (such as an RSLogix™ PLC). CIP Generic messages are used to read/write application specific data that is not included in the Input-Output assemblies. The 2500P-ACP1 allows the user to configure dynamic vendor-specific assembly objects to provide application specific data such as Device Profile (Product ID and Program Version) or Diagnostic data.

Vendor specific assembly objects are configured by assigning a Class ID, Instance, and Size (in bytes) of the assembly object. Application program variable(s) are then mapped to positions in the assembly object. Two message services are supported by the EIP Generic Message Adapter:

- Get_Attribute_Single – CIP Generic Message: Service Code 0E (hex)
  Used by a remote Generic Message client to read data from the ACP1 assembly object
  Supported Instance Attributes:
  - Data Length: Attribute ID = 1 (returns Size of assembly object in bytes)
  - Data: Attribute ID = 3 (returns data from all variables mapped to assembly object)
- Set_Attribute_Single – CIP Generic Message: Service Code 10 (hex)
  Used by a remote Generic Message client to write data to the ACP1 assembly object
  Supported Instance Attribute:
  - Data: Attribute ID = 3 (writes data to assembly object and associated variables)

The ACP1 Web Server provides tools for monitoring and debug. The status of connections to the Ethernet/IP Adapter can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the Open Modbus Server are listed in the *Protocol Error Codes* in *APPENDIX A*.

For more information regarding EtherNet/IP and Common Industrial Protocol (CIP), go to the ODVA website, http://www.odva.org/

7.1.7 EtherNet/IP Tag Client

The EtherNet/IP (EIP) Tag Client provides a method to read/write specified tags in Rockwell Automation RSLogix™ IEC-61131-compliant controllers via CIP explicit messages that directly access by “tag name”. This interface is particularly useful for data transfer between the 2500P-ACP1 application and RSLogix™ controller when a slower ‘update interval’ (seconds) can be used.

The EIP Tag Client driver issues ‘CIP Data Table Read’ and ‘CIP Data Table Write’ requests to access the controller data. These requests utilize the following CIP services:

- CIP Read Tag Service - Service Code 4C (hex)
- CIP Write Tag Service - Service Code 4D (hex)

The RSLogix™ controller tags are a collection of values of the same data type (i.e. atomic data type). Tags may be defined as a group of elements in user-defined data type (UDT) or array, allowing a block of data up to 480 bytes to be transferred in one message. Controller tags of the following data types are supported:

Data Type	Size of Data
BOOL	1 byte
SINT	1 byte
INT	2 bytes
DINT	4 bytes
REAL	4 bytes
DWORD	4 byte bit-array

The EIP Tag Client is configured by specifying the controller IP Address, service to be performed (‘Read Tag’ or ‘Write Tag’), RSLogix™ controller Tag name, the Number of Elements to be accessed, and Offset from the first data element in the Tag. Data is transferred to/from application program variables associated with the request. The request message can be triggered by setting Request Interval (period between requests) or by assigning a “trigger variable”.

Each RSLogix™ controller configured to communicate with the EIP Tag Client counts against the limit of 40 client connections supported by the 2500P-ACP1.

The ACP1 Web Server provides tools for monitoring and debug. The status of connections to the Open Modbus Server can be monitored via the ***Active Communication Sessions*** page. Error codes that may be returned by the Ethernet/IP Tag Client are listed in the *Protocol Error Codes* in *APPENDIX A*.

For more information regarding EtherNet/IP and Common Industrial Protocol (CIP), go to the ODVA website, http://www.odva.org/

7.1.8 TCP/UDP Management Functions

CTI Workbench provides a simplified interface to a full set of functions for management of TCP and/or UDP sockets for building client/server applications for communications over Ethernet networks. These functions allow the user to use simple strings and/or arrays to send/receive TCP (or UDP) messages with external devices using custom protocols that would normally require development of device drivers.

The ACP1 allows up to 10 TCP and/or UDP sockets to be in service simultaneously. These sockets are in addition to the 40 client and 30 server fieldbus connections that can be configured.

The following functions are supported:

Function	Description
TCPLISTEN	Create a listening server socket
TCPACCEPT	Accept client connection
TCPCONNECT	Create a TCP client socket and connects it to a server
TCPISCONNECTED	Test if a client socket is connected
TCPISVALID	Test if a socket ID is valid
TCPSEND	Send characters from STRING data
TCPRECEIVE	Receive characters and save to STRING
TCPSENDARRAY	Send characters from Array data
TCPRECEIVEARRAY	Receive characters and save to Array
TCPCLOSE	Release TCP socket
UDPCREATE	Create a UDP socket
UDPADDRMAKE	Build an address buffer for UDP functions
UDPISVALID	Test is a socket ID is valid
UDPSENDTO	Send UDP telegram from STRING data
UDPRCVFROM	Receive UDP telegram and save to STRING

7.1.9 MQTT Client

In firmware V4.0 and later, the MQTT Client operation is supported in the ACP1.

MQTT is an ISO standard publish/subscribe messaging protocol designed for connections to remote locations where a “small code footprint” is required and/or network bandwidth is limited.

MQTT is lightweight, open, simple, and designed to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication via satellite link, dial-up connections, home automation products, and Internet of Things (IoT) context.

The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. Its features include:

- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications
- A messaging transport that is agnostic to the content of the payload
- Three qualities of service (QoS) for message delivery
- A small transport overhead and protocol exchanges minimized to reduce network traffic
- A mechanism to notify interested parties when an abnormal disconnection occurs

The current version of MQTT is 3.1.1. As compared to V3.1, this version adds the following:

Session Present Flag

If a client connects with a persistent session, an additional flag was introduced in the connection acknowledgement message to indicate that the broker/server already has prior session information for the client (such as subscriptions, or queued messages). This is an important new feature which allows even more efficient communication. Now clients get feedback if the broker/server already has their subscriptions and does not need to re-subscribe to topics unless the flag is set to false.

Additional error code on failed subscriptions

Prior to MQTT 3.1.1, it was impossible for clients to find out if a subscription wasn’t approved by the MQTT broker/server. V3.1.1 adds a new error (0x80) in the MQTT SUBACK (subscription acknowledgement) message, so clients can react on rejected subscriptions.

Client IDs are now allowed to be very large

MQTT 3.1 had a limit of 23 bytes per Client ID which was very inconvenient and led to many problems, for example when using UUIDs for client identifiers. With the removal of this artificial restriction, client IDs can now use up to 65535 bytes.

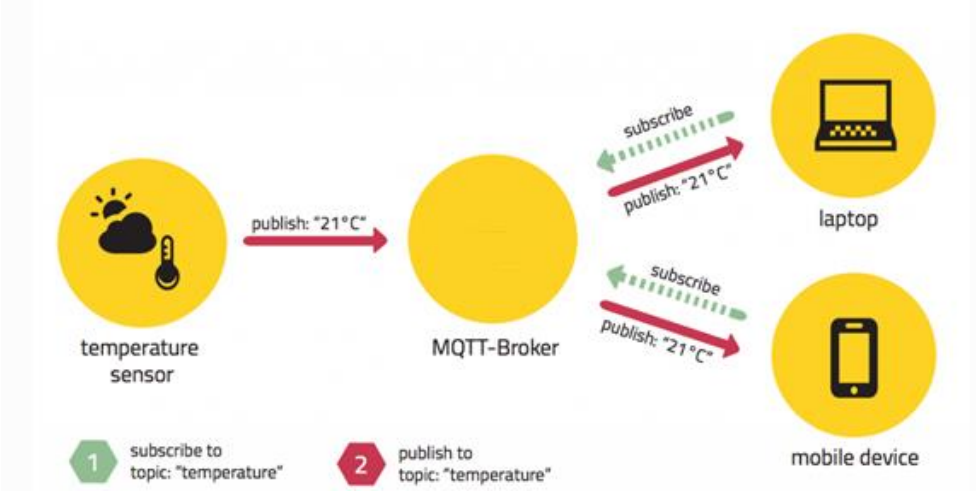
Anonymous MQTT clients

If your use case requires temporary or anonymous MQTT clients, it’s now possible to set the MQTT client identifier to zero byte length. The MQTT broker/server will assign a random client identifier to the client temporarily. *Note: This should be used only in exceptional cases.*

Publish/Subscribe Architecture

The publish/subscribe architecture is an alternative to a traditional client-server model, where a client communicates directly with an endpoint. In the MQTT model, publishers and subscribers do not communicate directly with each other. Instead, all MQTT client devices send publish and subscribe requests to an MQTT broker (also called an MQTT server), a device known to all clients. The broker/server is responsible for delivering published messages to all subscribers. For example, a sensor could send publish request, identified by a topic name, to the broker/server. The broker/server, in turn, would provide the data to clients who have subscribed to the topic.

In this context, a MQTT client can be a publisher, a subscriber, or both a publisher and a subscriber. The MQTT broker/server is primarily responsible for receiving all messages, determining who is interested in the message topic, and then sending the message to all clients who subscribed to the topic. Where security is implemented, it is also responsible for authenticating and authorizing the client.



This architecture decouples the publisher from the subscriber, which provides some important benefits:

- The publisher and the subscriber do not need to know each other's IP address and port,
- The publisher and subscriber do not need to run at the same time,
- Loading on the publisher and subscriber clients are reduced, since the broker/server assumes the work of managing published data and distributing it to all subscribers. This architecture provides greater scalability, since embedded devices are not required to handle a large numbers of TCP/IP connections and communications sessions.

The payload of a message contains the actual data in byte format. The MQTT specification does NOT specify how the payload is structured. It may could text, binary, or any other data format. It is up to the application developer to define the payload format and ensure that publishers and subscribers agree on the context of the transmitted data.

Connections

The MQTT connection is always between one client and the broker/server so that ***no client is connected to another client directly.*** The connection is initiated through a client sending a "connect" message to the broker/server. The broker/server responds with a connection acknowledgement reply which includes a status code. Once the connection is established, the broker/server keeps the connection open until the client sends a disconnect command or the connection is interrupted.

Normally, the broker/server does not maintain session information when a connection is lost. This is referred to as a "clean connection". However, the client can request a ***persistent*** session (by not requesting a "clean connection"), in which case the broker/server will maintain the session information including:

- All subscriptions
- All messages in a QoS 1 or 2 flow that are not confirmed by the client
- All new QoS 1 or 2 messages that the client missed while it was offline
- All received QoS 2 messages that re not yet confirmed to the client.

Topics

A Topic is a UTF-8 string that allows a publisher to identify the message. It is used by the broker/server as a filter to determine which clients, if any, have subscribed to the topic and to deliver the message to them.

A topic can consist of just one level or multiple topic levels. When multiple levels are used, each topic level is separated by a forward slash (topic level separator). Following is an example of a multilevel topic: ***BuildingA/Area3/Temperature***. Each topic must have at least one character.

NOTE: A topic is case sensitive; ***SensorA*** is a different topic than ***sensorA.***

When ***subscribing*** to a topic, the subscribed topic name may include wildcards:

- A plus sign "+" is used for a single level wild card. For example, ***BuildingA/+/Temperature*** would subscribe to the temperature in all ***BuildingA*** areas.
- A pound sign"#" is used for a multilevel wild card. It must be preceded by a slash"/" and be at the send of the topic. For example ***BuildingA/#*** would subscribe to all published topics in ***BuildingA***.

NOTE: Published messages cannot contain wildcard characters in the topic name.

Quality of Service (QoS)

The QoS level is an agreement between the sender and receiver regarding guarantees of delivering the message.

When discussing QoS, it is important to know that there are always two different parts of delivering a message: publishing client to broker/server and broker/server to subscribing client. The QoS level for publishing client to broker/server is set by the publishing client. When the broker/server transfers the message to a subscribing client, it uses the QoS that was specified by the subscribing client.

The operation of the QoS levels are described in the following table:

Level	Label	Description
0	At most once	This level guarantees a best effort delivery. Receipt of the message is not acknowledged by the receiver. The sender does not attempt to re-deliver a message. Often called “fire and forget”. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
1	At least once	At this level, the receiver must acknowledge the message. If the message isn’t received in a reasonable amount of time, the sender will re-send a message, setting a flag indicating it is a duplicate. If the receiver has already processed the first message, it will send an acknowledgement but ignore the message.
2	Exactly once	This level guarantees that each message will be received only once by the recipient. It is the slowest QoS level, since it requires two handshakes between the sender and receiver. HANDSHAKE1: The sender transmits the message to the receiver. When the message is received and processed, the receiver transmits a PUBREC (publish received) acknowledgement to the sender. HANDSHAKE 2: After receiving the PUBREC acknowledgement, the sender discards all stored state information and responds with a PUBREL (publish release) message. When the PUBREL is received, the receiver transmits a PUBCOMP (publish complete) message to the sender.

Quality of Service Best Practices

Use QoS 0 when …

- You have a complete or almost stable connection between sender and receiver. A classic use case is when connecting a test client or a front end application to a MQTT broker/server over a wired connection.
- You don’t care if one or more messages are lost once a while. That is sometimes the case if the data is not that important or will be retransmitted at short intervals where the having the latest data is important.
- You don’t need any message queuing; Messages are only queued for disconnected clients only if they have QoS 1 or 2 and a have established a persistent session.

Use QoS 1 when …

- You need to get every message and your use case can handle duplicates. The most often used QoS is level 1, because it guarantees the message is received at least once. Your subscribing application must tolerate duplicates and process them accordingly.
- You can’t bear the overhead of QoS 2. QoS 1 is much faster in delivering messages.

Use QoS 2 when …

- It is critical to your application to receive all messages exactly once. This is often the case if the application in subscribing clients cannot tolerate duplicate delivery of a message. You should be aware of the overhead associated with the QoS 2 flow.

Operation

The operation of the MQTT Client including connection to broker, publish to broker, subscribe/unsubscribe to broker, receipt and storage of messages received from broker, and status/statistics for broker connections are managed via a set of function blocks located in the MQTT folder. Because MQTT operation is controlled by logic, these functions operate only when the ACP1 is in RUN mode.

Configuration

The MQTT Client configuration requires entry of following properties:

- Connection ID: Used by MQTT function blocks to identify a specific broker connection.
- Server Address/Port: Enter IP Address assigned to broker. Always use TCP Port 1883.
- Client ID: Most MQTT server/brokers require each client connection to have a unique identifier string.
- Buffer Size: Maximum number of characters to be sent (or received) from broker. Messages up to 255 characters may use Strings or Text Buffers for data storage. Messages larger than 255 characters must use Text Buffer function blocks.
- MQTT Version: Use 3.1.1 unless broker requires use of V3.1.

Optional properties include:

- User Name/Password Login: Necessary only when broker requires a password for client connections.
- Clean Session: Select this option to tell the broker to not maintain session information when connection is lost. See details in the ***Connections*** section above**.**
- Will Message: Specifies ***Topic, Message Contents,*** and ***Quality of Service*** for ***Last Will and Testament*** message which is transmitted by broker to all Subscribers when connection to Publisher is lost. These properties are valid only for client connections that publish messages to broker.
- IP Address and selection of the I/O assemblies to be used. Each connection to a remote EIP I/O Adapter counts against the limit of 40 client connections supported by the 2500P-ACP1.

Each connection entered in the MQTT Client fieldbus configuration counts against the limit of 40 client connections supported by the 2500P-ACP1.

7.2 Serial Port Communications

The 2500P-ACP1 provides communications protocols for connection of RS-232/RS-422 compatible devices to the serial port. Only one serial port protocol may be active at any time.

7.2.1 Serial Port Protocols

Function	Description
Modbus RTU Master	Allows ACP1 to function as Modbus RTU Master
Modbus RTU Slave	Allows ACP1 to function as Modbus RTU Slave
General ASCII	Sends and receives ASCII characters. Can be used to communicate to devices using proprietary serial protocols. Data is transmitted and received based on the application logic.

7.2.2 Serial Port Configuration

All serial port parameters, including the electrical interface (RS232 or RS422), are set by software configuration. Parameters are specified by an ASCII string containing descriptors and associated values. This string is used to setup the serial port as required by serial protocol drivers.

The following table lists serial port setup parameters:

Parameter	Descriptor	Valid Values	Default Value
Port	PT	ALWAYS = 1	1
Baud Rate	BD	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	9600
Data Bits	DB	7, 8	8
Stop Bits	SB	1, 2	1
Parity	PY	None (N), Even (E), Odd (O)	ASCII (N) Modbus RTU (E)
Flow Control	FC	No (N), Yes (Y) 'Y' enables Hardware RTS-CTS handshake (CTS must be TRUE to send)	N
Interface	IF	RS232, RS422	RS232

This is an example string to setup the serial port:

PT=1 BD=19200 DB=8 SB=1 PY=N FC=N IF=RS232

Usage Rules:

a) If any parameters are missing or invalid values assigned in the string, the default value for the parameter(s) will be used.

b) All characters in the string are case insensitive (consistent with IEC 61131 spec).

c) The string is not order dependent (consistent with IEC 61131 spec).

d) Any extraneous information included in the string will be ignored.

CHAPTER 8 APPLICATION DEVELOPMENT

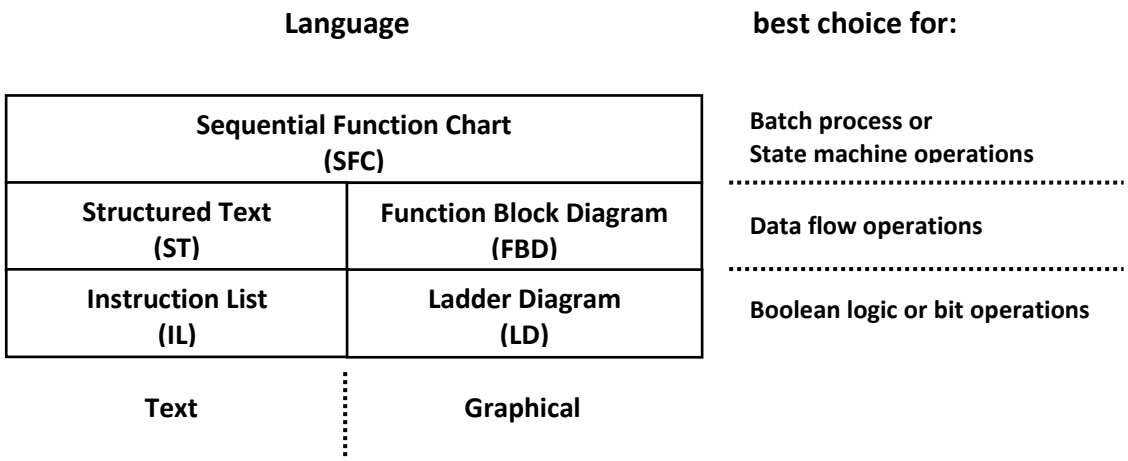
This section provides an overview of design, organization, development, and execution of 2500-ACP1 applications. Details for using the programming languages and instructions are provided in ***CTI Workbench*** on-line Help system.

8.1 IEC-61131 Concepts

The IEC-61131 environment provides features to address the deficiencies found in most PLCs:

- Support for textual and graphical programming languages to provide "best-fit" for each application. A single project may contain programs written in several different languages.
- Memory organization supports complex entities, i.e. data structures.
- Both logic and data is encapsulated into a single program unit (POU). This provides the "local data" concept where internal information cannot be modified by other parts of the program.
- Hierarchical design encourages duplication and re-use of proven software sections.
- Greatly enhances control over execution cycle.

The following programming languages may be used for application development:



8.2 Glossary of Terms

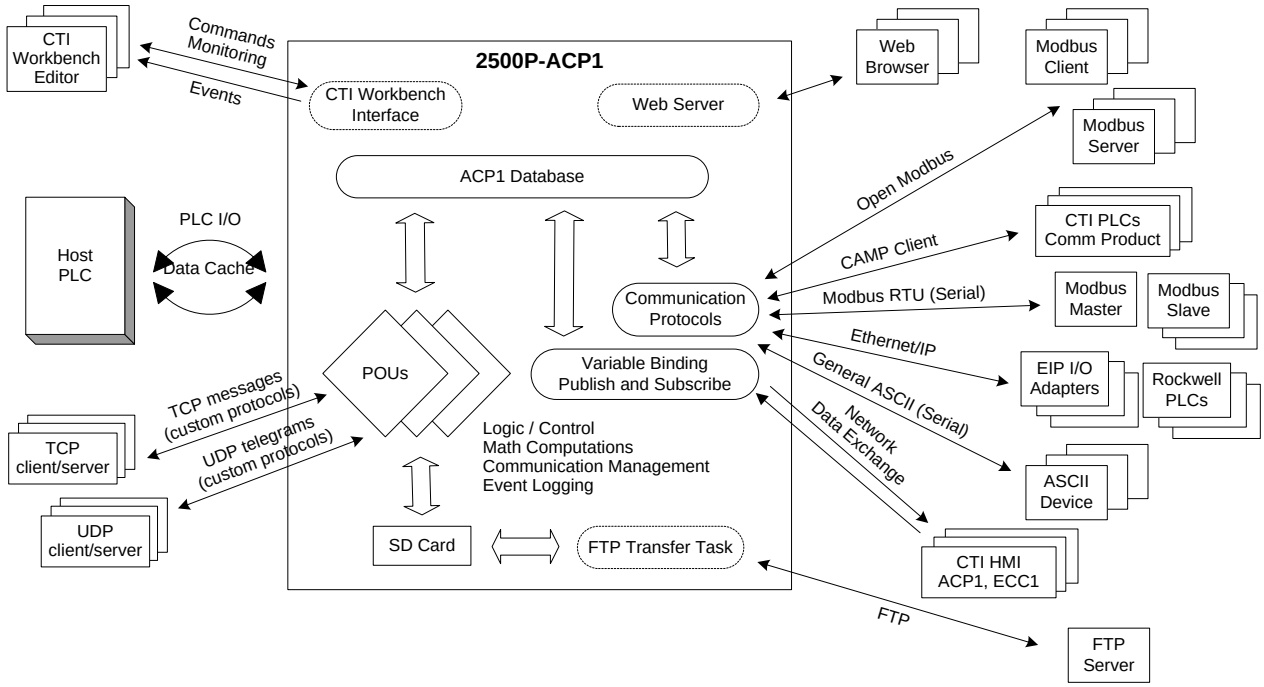
This section describes the items that comprise an ACP1 application and their function:

Item	Description
Data Type	Classification of memory location so that the length, possible values, and interpretation of data values can be determined. The run-time supports simple data types (single element such as BOOLEAN, INTEGER, or REAL) and complex data types derived from simple data types (Structures, Enumerations, Bit Fields, and Function Block instances).
Variable	Symbolic name and data type for database position(s) where the value is stored. Variables may be grouped into Arrays and accessed via index.
Instruction	Graphical symbol, text statement, or expression that performs an operation and/or represents the control flow (order of execution) through the logic program. Valid instructions are specific to the programming language.
Function	Operation that calculates a result based on the current value of the input parameters. A function is not linked to a specific instance and no internal data is retained when the operation completes.
Function Block	A group of private data and an algorithm specifying its operation. A Function Block has inputs, outputs, and internal variables and must have a unique name (or instance) declared each time it is used. When called, the Function Block executes the algorithm on the private data using the current value of the input parameters.
POU	Any one of the Program Organization Units (or program types) created by the user: Main Program, UDFB, Sub-Program, Exception Program, or SFC Child Program. A POU is made up of instructions, variables, constant expressions, and calls to other POUs.
Program	POU created in any IEC-61131 language that is called directly by the run-time system within the execution cycle. The Program then calls other POUs as defined by the application. One Program is required, but multiple Programs may exist in which case they are called sequentially in the order defined by the user.
UDFB	User Defined Function Block. POU written in any IEC language that is functionally equivalent to standard Function Block described above. It may be called by any other POU and may call any other UDFB, Sub-Program, Function, or Function Block.
Sub-Program	POU containing inputs, outputs, local data, and an algorithm written in FBD, LD, ST, or IL language that describes its operation. A Sub-Program is not instantiated and does not have a unique set of local data for each use. It may be called by any other POU and may call any other Sub-Program, Function, Function Block, or UDFB except SFC Child Program.
Exception Program	Program units for special processing and error handling. Exception Programs for the following system events are pre-defined: Startup (before first cycle), Shutdown (after last cycle), Divide by Zero, and Array Index out of Bounds.
Structured Text	Structured Text (ST) is a literal programming language consisting of a list of statements. Each statement represents an action and must end with a semicolon (";"). Presentation of the statements has no effect on the program operation as blank spaces and line breaks can be inserted at any point. ST is a structured language where execution flows from top to bottom, and "jump statements" are not allowed.

Item	Description
Instruction List	Instruction List (IL) is a program consisting of list of operations. Each operation (or instruction) has one or more operands that may include variables or constant expressions. Each instruction may include a label that provides the destination for jump instructions. It is possible to mix ST and IL languages within a single POU.
Function Block Diagram	Function Block Diagram (FBD) is a graphical language where basic operations (functions) are represented by a set of blocks. Inputs and outputs of the blocks are wired together with lines used to connect two logical points of the diagram (such as an input variable to input of block, output of a block to output variable, or output of a block to input of another block). Execution flow is from left to right and top to bottom of the diagram.
Sequential Function Chart	Sequential Function Chart (SFC) is a graphical language used to program processes that can be split into states (or steps). Main components of SFC are steps (containing actions), transitions (logic instructions), and connections (or directed links) between steps and transitions. A step becomes active when the preceding step is active and the connecting transition is TRUE. LD and/or ST is used on program actions and transition logic.
SFC Child Program	POU developed in SFC language that is controlled (started/stopped) from the action blocks of its parent SFC Program. A SFC Child Program may also have “children”.
Fieldbus Configuration	User definition of the operation and processing for each supported communication protocol and data transfer to/from ACP1 variables during run-time. Configuration options are specific to the protocol and function (such as Master or Slave).
Binding Configuration	User definition for the real-time Network Data Exchange among CTI devices supporting this protocol (2500P-ACP1, 2500P-ECC1, and 2500 Series HMI products). Variable Binding uses an event-based Publish/Subscribe model for high performance and low network traffic.
Project	Collection of components used to build an executable application. These components include POUs, Variables, Global Defines, Fieldbus Configurations, and Binding Configuration.
Execution Cycle	One complete set of operations performed by the ACP1 run-time based to the fieldbus configuration and instructions included in the application project. “Cycle Time” is defined as the time interval required to complete one execution cycle and may be set to Variable (execute as fast as possible) or Fixed (wait for expiration of timer before starting next cycle). This term is analogous to the “PLC scan”.
On-Line Change	Feature supported by ***CTI Workbench*** and the ACP1 run-time that allows an executing application to be changed “on the fly” without stopping it to download and initialize the new project.

8.3 Functional Overview

The following block diagram provides a functional overview of the 2500P-ACP1 module:



8.4 Application Design

An application for the 2500P-ACP1 module is called a Project. The following items must be understood, configured and/or programmed by the user before the Project can execute in the 2500P-ACP1 module:

- Project Settings
- Program Organization Units (POUs)
- Execution Cycle
- Variables
- Device Communications
- Variable Binding (Network Data Exchange)

8.4.1 Project Settings

The Project Settings dialog is used to specify details of the Project environment:

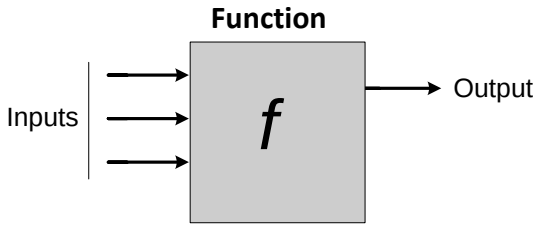
- Communication parameters for ***CTI Workbench*** to ACP1 module
- Run-time environment:
    - Cycle time – Variable (as fast as possible) or Fixed value
    - On-Line Change – When enabled, the program can be modified and replaced while the target application is running. However, certain types of edits are limited and/or restricted when On-Line Change is enabled.
- Compiler Options:
    - Code Generation – Select Debug mode (enables breakpoints and step-by-step execution) or Release mode (best performance)
    - Complex Data processing – This option should be disabled unless required since it degrades performance. It must be selected when certain complex variable types are used in the program. When enabled, all complex variables (arrays, structures, and function block instances) are place in a separate memory segment.

8.4.2 Program Organization Units (POUs)

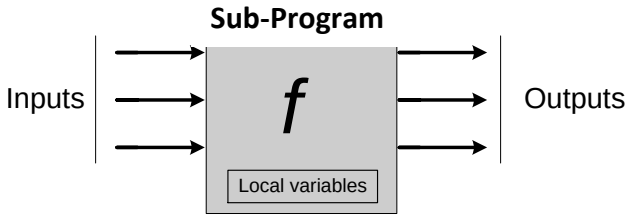
This section describes the organization of Program units that can be used to build a project.

Functions / Sub-Programs

A Function calculates a result based only on the input values. A Function allows multiple Inputs but only one Output parameter and no internal data. Functions are GLOBAL in scope and can be called from any POU. The run-time includes both IEC standard Functions and CTI Functions.



The Sub-Program is an extension to the IEC standard Function. A Sub-Program may include up to 256 Input and 256 Output Parameters and can be developed using FBD, LD, ST, or IL language. Local variables may be declared and used in the embedded logic code that can include calls to any other Function, Function Block, and/or UDFB. Since the Sub-Program executes like a Function, the Local variables are not instantiated. This means the same set of internal variables are used for all occurrences within the application.



Function Blocks / UDFBs

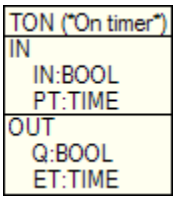
Function Blocks contain both code (logic) and private data (Inputs, Outputs, and Local variables). When called, the logic reads the Input Parameters and performs the specified operation on the private data. The run-time includes both IEC Standard and CTI Function Blocks.

The IEC-61131 model uses the Data Structure construct as a template or “data type” for the internal data. To use a Function Block, a unique variable having that data type must be declared for each occurrence of the Function Block in the application. This creates an ***instance*** of Function Block and its private variables.

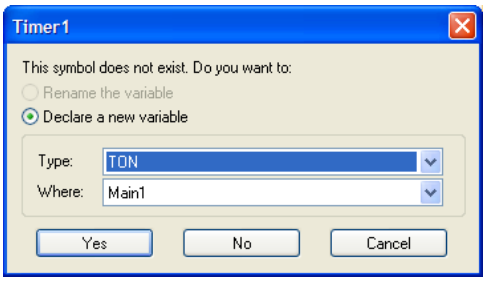
The UDFB is a user-developed program unit that is operationally equivalent to the standard Function Block described above. There are no restrictions in the development and usage of UDFBs. The logic can be constructed in any available language. A UDFB can be called from any other POU and its logic can call any Function, Function Block, and/or other UDFB. A maximum of 256 Input and 256 Output Parameters are supported.

The following examples show the declaration for an instance of IEC Standard Function Block “TON” :

In/Out Parameters



FB “Instance” Variable Declaration



Programs

The Program is highest-level POU in that it is called directly by the run-time as part of the execution cycle. An application must have at least one Program and can have a maximum of 32767. If more than one Program exists, they are executed according to the sequence defined in the Cycle configuration menu.

A Program is made up of logic instructions, data flow statements, expressions, and calls to other Functions, Sub-Programs, Function Blocks, and/or UDFBs. Programs can include Local variables as well as those variables with GLOBAL scope. Each time a Program is triggered by the run-time, it executes from start to finish before the execution cycle can be completed.

A Program can be developed using any available language, and Programs written in different languages can exist in the same application. When using SFC, a hierarchy of SFC Programs may be formed by creating one or more Child SFC Programs. Child SFC Programs are started/stopped in the Action Blocks of the parent Program. A Child SFC Program may also have “children” within the limit of 19 hierarchy levels.

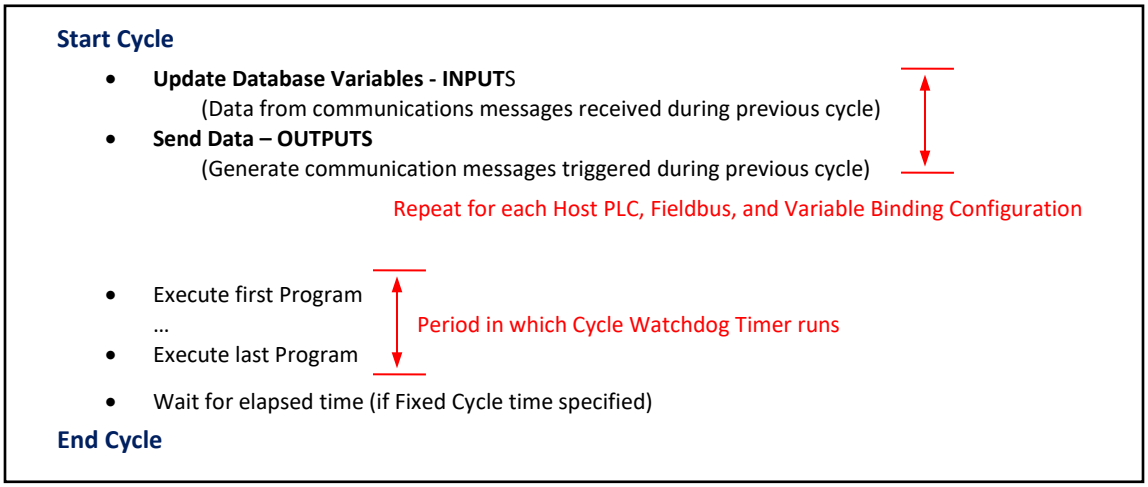
Exception Programs

Exception Programs allow the user to create logic routines to perform special processing and/or error handling when certain events are detected. The following exception events are supported in the 2500P-ACP1:

- Startup: Runs once before the first execution cycle when the application is started
- Shutdown: Runs once after the last execution cycle when the application is stopped
- Divide by Zero: Called when division by zero is detected
- Bad Array Index: Called when invalid array index position is used within the logic

8.4.3 Execution Cycle

The 2500P-ACP1 application is executed sequentially according to the following model:



Each configured communications driver (ACP1 Data Cache, ACP1 I/O, CAMP Client, Open Modbus Client, Open Modbus Server, Ethernet I/P Scanner, EtherNet/IP Adapter, EtherNet/IP Tag Client, and Variable Binding) is called each cycle and runs until all input/output messages are processed.

It is not required that each Program run during each execution cycle. The frequency and sequence order for the execution of each Program is set by the 'Cycle Configuration'. This gives the user total control over what Programs run during each execution cycle.

Each POU is assigned the same priority and runs to completion before the next program unit is called. This non-preemptive scheduling method provides a much simpler approach to system design since the execution sequence is determined by the order in which each program is called and the logic flow within the program.

The 2500P-ACP1 has an internal Cycle Watchdog timer to detect software anomalies in the application program that can result in the module "hanging" and becoming unresponsive. If the Cycle Watchdog timer expires during any execution cycle, the module is forced out of RUN mode and into a **Fatal Error** state.

This watchdog timer is fixed at 2000 msec (2 seconds) and starts when the first *Program* is called during each execution cycle and is cleared when the last *Program* in that cycle completes (see figure above). The cycle watchdog period does not include the time used to perform database updates, communication tasks, and system overhead functions.

The user must be careful when using loop instructions (such as WHILE, REPEAT, and FOR) not to block completion of a program unit while waiting on an event that is generated elsewhere in the program. This will result a Cycle Watchdog timeout error.

CAUTION:

The sequential execution method allows the user complete control over "when" each program unit is executed. However, the programmer must be careful not to create an infinite loop that blocks the completion of the program execution.

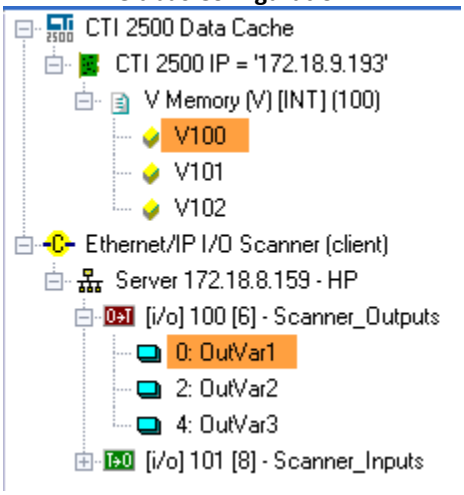
NOTE:

Communications to/from all field devices are processed at the beginning of each scan (similar to PLC I/O processing). Logic programs then run after communications processing is completed. This sequence order is important when the ACP1 acts as a "concentrator" or "gateway" device where data received from one field device is then transmitted to another device (usually using a different protocol).

This type of operation is implemented most often using CTI Data Cache to access PLC data which is then transmitted to one or more field devices. In this case, the logic program can control the data ONLY if a different application variable if used for each fieldbus interface.

In the following example, data read from "V100" in the PLC (using Data Cache) is then transmitted to EtherNet/IP field device (using the EIP I/O Scanner). Since we want to monitor the value sent to the field device to ensure it is never less than a preset minimum value, we cannot use the same "V100" variable. Instead, a different variable "OutVar1" is used so the program logic can limit the value when required.

Fieldbus Configuration:



Program Logic:

```
IF ( V100 < 25 ) THEN
    OutVar1 := 25;
ELSE OutVar1 := V100;
```

8.4.4 Variables

All data elements are addressed indirectly by “name” as opposed to the direct (or absolute) addressing method used in the Siemens® 505 and CTI 2500 Series® controllers. The name itself identifies the data, and a data type is then assigned to the variable so that the run-time can determine the data length and how to interpret the values stored at that memory location.

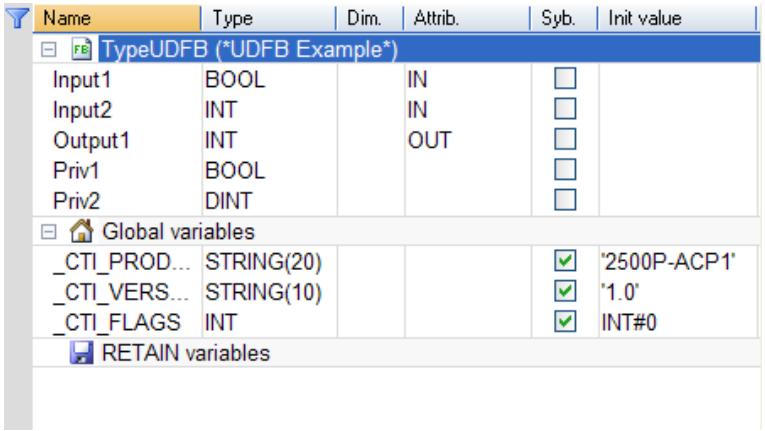
Most variables represent a single data element and are assigned a simple data type (i.e. BOOLEAN, INTEGER, or REAL). However, the user can create a custom type that combines various data types into a complex unit known as a Structure. The Structure can then be used to organize the information related to a device or machine into a data group represented by a single variable name.

Arrays of variables of all data types can be created so that common elements can be grouped together. One, two, and three-dimensional arrays may be used.

Three classes of variables are supported:

- Global: Internal variable known to all programs in the application
- Local: Internal variable known only to the POU where it resides
- Retain: Non-volatile Global variable used to restore critical variables to previously store states
  IMPORTANT NOTE: Retain variable storage on 2500P-ACP1 is limited to 11,500 bytes total.
  IMPORTANT NOTE: Retain variable storage is further limited to 992 bytes when using the F_SAVERETAIN and F_LOADRETAIN function blocks.

Variable Editor



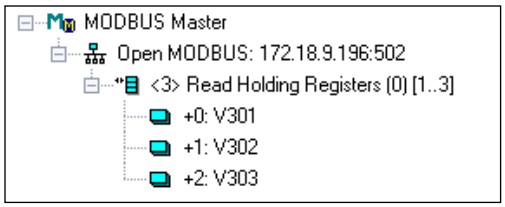
8.4.5 Device Communications

Communications between the 2500P-ACP1 run-time and field devices using any of the embedded protocol drivers are enabled and controlled via the **Fieldbus Configuration** editor. Protocols are enabled by selecting the appropriate protocol service from the list of supported drivers (see Chapter 7) and specifying the operational elements as shown below:

Fieldbus Configuration Tree

- Protocol / Service
  - Connection Parameters
    - Command / Data Block
      - Variables

Protocol Configuration Example



The variables specified included in the Protocol Configuration define the data to be:

- Embedded in designated messages transmitted to specified device
- Extracted from the messages received from the specified device
- Used for Diagnostics (status and error reporting) and/or Control (message triggering)

8.4.6 Variable Binding

Variable Binding provides an efficient means to exchange real-time data between devices connected to an Ethernet network through an event-based Publish and Subscribe model. Variable Binding network data exchange is currently supported by 2500P-ACP1, 2500P-ECC1, and 2500 Series® HMI products.

CTI Workbench includes two editors for configuring the network data exchange process. This configuration links each Published variable to its destination (Project and Variable name). An application can publish any variable that represents a single data element. Complex data types such as Structures, Strings, and Arrays cannot be exchanged via Variable Binding.

1) The **Global Binding Editor** provides a system-wide view of data exchange. This editor allows access to variables for all Projects within the active Workspace and automatically builds the Variable Binding configuration for each Project.

2) The **Binding Configuration Editor** allows the user to setup the data to be published and subscribed to (data published by other systems) on a local Project level. The Variable Binding configuration must be manually setup for each Project. This method must be used for network data exchange with systems not included in the same Workspace environment.

CHAPTER 9 UPDATING FIRMWARE

9.1 Overview

The CTI 2500P-ACP1 module stores the operating firmware in non-volatile flash memory. You can replace the current operating firmware with a different version to correct problems or add new features. During this procedure, the new firmware will be copied to controller RAM, verified, and then written to flash memory.

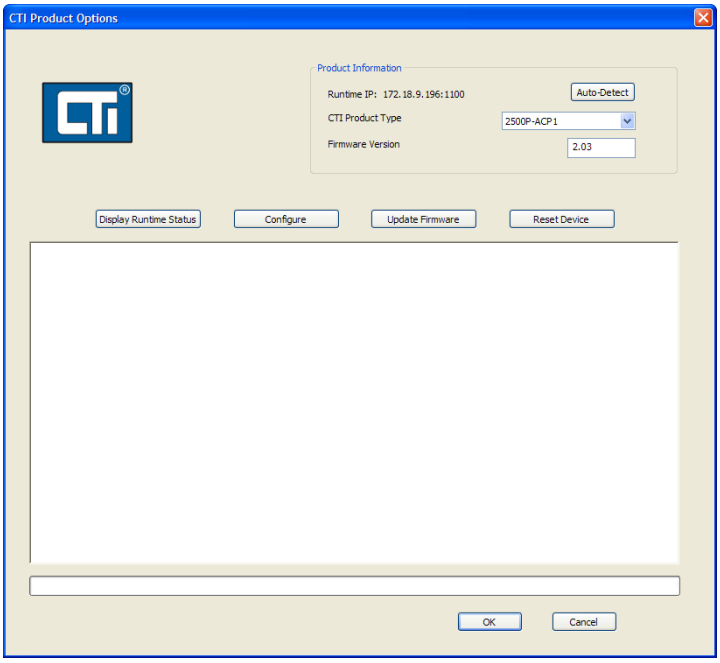
The firmware update method for each 2500P-ACP1 module is determined by the SW1 switch position as described in Section 3.1.2. When SW1 is CLOSED, the firmware is updated using ***CTI Workbench*** which transfers the firmware update file to the 2500P-ACP1 over an Ethernet link. This method performs an automated remote update operation where human intervention is not required to reset the module. This allows the firmware update to be completed from a remote location. An alternate method is available for installations where company policy or security concerns require a person be present when module firmware is updated. When SW1 is in the OPEN position, the "remote" method is disabled and firmware update must be carried out by using the SD card.

During the firmware update process, various status codes and error codes may be displayed on the 2500P-ACP1 front panel LED display. Firmware update status codes, which start with "**U"** followed by two numeric digits, indicate what is happening in the update process. See Section 9.4 for a list of the firmware update status codes. Firmware update error codes, which start with "**E"** followed by two numeric digits, are used to indicate the specific error that occurred. When an error occurs during firmware update, the process will stop. You must cycle power to the base to restart the firmware update procedure. See *APPENDIX A: ERROR CODES* for a list of error codes and corresponding corrective actions.

Prior to updating firmware, you will need to obtain a firmware update file for the 2500P-ACP1 module. This file can be downloaded from the CTI website http://www.controltechnology.com/downloads/ . After obtaining this file, you should save it to a file on your PC or on an accessible network drive.

9.2 Ethernet Firmware Update Method

The Ethernet firmware update method provides an efficient means for updating firmware on the product using the *CTI Product Options* utility within ***CTI Workbench*** as shown:



The steps in the firmware update process are described below.

1. Prepare for Firmware Update via Ethernet Connection

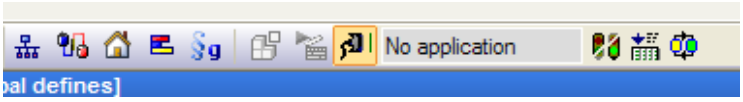
a. Download firmware update file from CTI website (see Section 9.1).

b. SW1 must be in the CLOSED (Ethernet Firmware Update Allowed) position.

c. Module must be installed in the base and power applied.

d. ACP1 application program must be halted.

When halted, *No Application* is displayed in status window as shown below:



NOTE:

Do not remove power from the module while the firmware update process is active.

2. Initiate Firmware Update Process

a. Select ***Update Firmware*** button in *CTI Product Options* screen.

b. Browse to folder and select the firmware update file.

c. If the module status is OK, the firmware update process is started.

d. The ACP1 module is reset, and the module is initialized to “firmware update” mode.

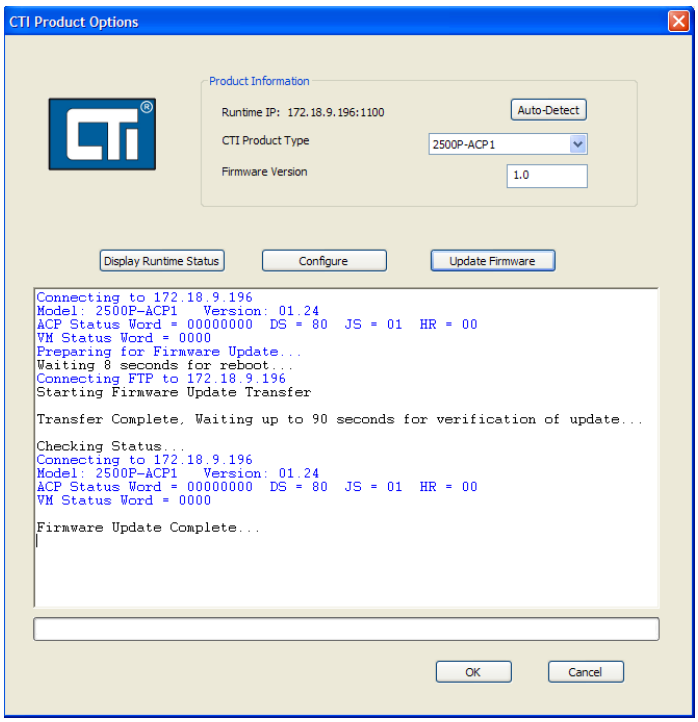
e. The firmware update file is downloaded to module. When the file transfer is completed successfully, a message is displayed indicating the file was successfully transferred.

f. After the file is transferred, the ACP1 module validates the file and, if successful, begins writing the firmware file to flash memory.

g. When complete, the module resets and is initialized to normal operation mode.

h. The status of the firmware update operation is reported to ***CTI Workbench***.

The following screen shows the results of a successful firmware update:



If an error occurs, an error status is returned to ***CTI Workbench***. An error code is displayed on the module LED display (see *APPENDIX A: ERROR CODES)* and the firmware update process is terminated. The user must go to the module location to diagnose and correct the error condition.

9.3 SD Card Firmware Update Method

The SD card firmware update method provides an alternate method of updating the ACP1 module firmware. This method requires the user to be present at the module while the firmware update operation is executed.

1. Prepare for Firmware Update using an SD Card

a. Download firmware update file from CTI website (see Section 9.1).

b. Disconnect power and remove the module from the base.

c. Remove any SD card installed in the product. Copy the firmware update file into the root directory or install new SD card containing the firmware update file in the root directory.

> ***NOTE:***
> ***The firmware update process will not work if there is more than one firmware update file in the root directory of the SD card.***

d. Set SW2 to the CLOSED (SD Card Firmware Update) position.

e. Install the Product into the base and apply power.

2. Monitor the Firmware Update Process

> ***NOTE:***
> ***Do not remove power from the module while firmware is being updated.***

When the ACP1 module starts up in SD card firmware update mode (with SW2 in CLOSED position), the module attempts to copy the firmware update file contents from the SD card to module RAM. Status code **U02** will be displayed on the LED Multi-Segment Display while this is taking place. If an error is encountered, such as a missing firmware update file, an error code is displayed and the firmware update process is terminated (see APPENDIX A: ERROR CODES). The firmware update process must be restarted after the problem is corrected.

Once the file has been copied to RAM and verified, the ACP1 module will proceed with replacing the current firmware in flash. While flash memory is being rewritten, various status codes will be displayed on the front panel Multi-Segment Display. If an error is encountered while updating flash, an error code is displayed and the firmware update process terminated. The error must be corrected before the firmware update process can be restarted. When the firmware update process is successfully completed, status code **U00** will be displayed.

3. Return the ACP1 Module to the Normal Operating Mode

a. Disconnect power and remove the ACP1 module from the base,

b. Replace the firmware update SD card with the operating SD card (if required).

c. Set SW2 to the OPEN (SD Card Normal Operation) position.

d. Re-install the module into the base and apply power.

9.4 Firmware Update Status Codes

The following status codes indicate progress in updating firmware. Under normal circumstances, many actions happen so fast that individual status codes cannot be recognized.

Status Code	Status
U00	Module firmware successfully updated
U01	Waiting for firmware file transfer to begin
U02	Firmware file transfer in progress
U03	Validating downloaded Image File header in DRAM
U04	Validating downloaded Image File in DRAM
U05	Building the CTI Image File in DRAM.
U06	Erasing the application data communications area
U07	Erasing the FLASH area to hold the CTI Image File
U08	Writing the CTI Image File to FLASH
U09	Verifying the CTI Image File was written to FLASH properly
U10	Searching for an Image Trailer in FLASH
U11	Searching for a Boot Loader Image in the image file
U12	Validating the Boot Loader Image found in FLASH
U13	Copying the Boot Loader Image from FLASH to DRAM and verifying copy
U14	Erasing the area in FLASH to hold the downloaded Boot Loader
U15	Writing the downloaded Boot Loader into FLASH

APPENDIX A: ERROR CODES

Operational Error Codes

Operational Error Codes are detected by the ACP1 module firmware and indicated on the front panel. While any error is active, the Status LED blinks ON/OFF. The Error Code is presented on the front panel multi-segment LED display by displaying "Err" and then the 3-digit number corresponding to the error condition.

It is possible that multiple errors may be active simultaneously. This condition is indicated by blinking the right-most decimal point in the display. In these instances, the highest priority error code shall be displayed. Once this error has been cleared, the next highest priority error shall be displayed.

The following tables include error code, description, and recovery steps for all error conditions.

Startup Errors

Error Code	Description	Comments	Recovery
010	No Network Configuration in flash. Use default IP Address.	Uses default network settings	Configure module via Web Server. See Section 5.1.2.
020	Ethernet Port Switch Error	Unable to configure switch	Reset the module. If error persists contact CTI Support.
030	Manufacturing Data Error	Checksum error detected	Reset the module. If error persists contact CTI Support.
040	OS Component Start Error	Module startup error	
120	SD Card Not Accessible	Unformatted or unsupported type of SD Card	Reformat SD card for FAT file system and reset module. If the error persists, replace the SD Card.
130	SD Card Not Found	SD Card cannot be accessed, missing or unsupported type.	Insert a compatible SD card See Section 3.1.1.
140	Configuration File Not Found	The ACP1 configuration file (acp1.ini) was not found on SD Card in /acp1 folder	Configure module via Web Server. See Section 5.1.2.
160	Invalid Configuration File	The contents of the configuration file are not valid for ACP1 module	
170	Application Program Not Found	Application program file cannot be found on the SD card	Connect ***CTI Workbench*** to module and download application program.
175	Incompatible or Unlicensed Application Program	Program is not compatible with module run-time version, or program was compiled with unlicensed installation of ***CTI Workbench***	Re-compile program with licensed version of ***CTI Workbench***. '*Firmware Version*' value entered in 'Project Settings' must match firmware version in ACP1 module.
180	ACP1 Run-time Start Error	Unable to start ACP1 execution engine	Reset the module. If error persists contact CTI Support.
190	System Exception Error	Fatal Error detected. See Event Log for details.	Correct error and download new program or reset module.

CTI Data Cache Interface Errors

Error Code	Description	Comments	Recovery
210	TCP Connection Failure to Host PLC	Data Cache interface is unable to connect to Host PLC.	Ensure the Host PLC is online. Ensure there is a network path between the ACP1 module and the Host PLC. Confirm the configured Host PLC IP Address matches the one being used Confirm that the ACP1 module network settings are compatible with the Host PLC IP Address and subnet mask (see *APPENDIX B*).
220	Host PLC Firmware Incompatible	The Host Controller firmware version and ACP1 firmware versions are not compatible.	Ensure that the ACP1 and Host PLC are updated to compatible firmware versions.
230	Host PLC Registration Failed	The ACP1 module cannot register because the Host Controller is already connected to maximum number of AFM modules.	If you are replacing an ACP1 module, ensure that the module to be replaced is disconnected before connecting the replacement module.
240	Host PLC Link Inactive	The ACP1 module is not communicating with the Host Controller.	Ensure the Host PLC is online. Ensure that the module and the Host Controller are connected to the same local area network. If a VLAN is used, ensure that the Host PLC and module are on the same VLAN.
310	Host PLC Fatal Error	The Host PLC is in Fatal Error state. Data Cache variables are not being updated.	Clear the Host PLC Fatal Error.
320	Host PLC in Program Mode	Host PLC logic program is not running.	Change PLC mode to RUN.
330	Inaccessible PLC Address	Unable to access one or more PLC memory locations specified in the Data Cache configuration.	Compare the Host PLC memory configuration with the memory types/offsets specified for Data Cache variable mapping. Change Host Controller memory configuration or revise the Data Cache variable mapping.
335	No Memory Buffers Available for Data Transfer	Number of variable “write data” requests to PLC exceeded number of available buffers. Data transfer is delayed one PLC scan.	Error clears automatically on the ACP1 execution cycle. Increase CPU time slice for ACP1 and set “fixed” ACP1 cycle time.

CTI 2500 I/O Interface Errors

Error Code	Description	Comments	Recovery
365	Host PLC I/O Output Disable detected by module	Indicates PLC has stopped I/O communications with ACP1. Usually caused by mismatch between ACP1 application program and PLC '*I/O Login'* configuration for ACP1 module slot.	Ensure PLC is operational (no Fatal Error state) and CTI 2500P-ACP1 I/O configuration matches Host PLC '*I/O Login*' for module slot where the ACP1 is installed. Module automatically resets FPGA sub-system and attempts to recover.
370	Host PLC I/O Data Exchange Timeout	Indicates ACP1 module has not detected PLC I/O data transfer within 30 second interval. Can be the result by a mismatch between ACP1 application program and PLC '*I/O Login'* configuration for ACP1 module slot or ACP1 sub-system failure.	
380	Uncorrectable ECC Memory Error	ACP1 detected an uncorrectable memory error during I/O data transfer. Data transfer from PLC is delayed one PLC scan.	Error is corrected when memory cell is written during the next I/o data transfer.

Run-Time Errors

Error Code	Description	Comments	Recovery
390	Scan Cycle Overrun (Watchdog Timeout)	An execution cycle in the ACP1 application exceeded the 2000 msec timeout period. May be caused by logic "loops".	Connect ***CTI Workbench*** to correct error and exit ERROR mode. The application must be stopped and restarted. Error can also be cleared by resetting ACP1 module (but logic error will still exist in program).
392	Invalid Array Index	A logic error in the ACP1 application caused an invalid array index to be accessed.	
395	SD Card Write Protected	Detected when write to SD Card operation is attempted by program download or 'write to file' function called by the application program.	Set SD Card switch to disable write protection and reset module.
398	SD Card Free Space Warning	Used disk space on SD Card is periodically calculated when FTP File Transfer is enabled. Indicates the percentage of used space exceeds *Warning Level* set in ACP1 Web Server.	Use ACP1 Web Server *File Management* page to transfer files to FTP Server to delete user files.
398	SD Card Disk Full	Indicates available disk space on SD Card was insufficient to store data required by 'write to file' operation.	
400	Data Exchange Subscriber Error	Data Exchange Subscriber encountered a startup error or is unable to connect to one or more publishers.	Ensure publishers are online Ensure that the correct IP address has been configured for each subscriber to publisher session.
405	FTP Server Authorization Error	Indicates ACP1 module was unable to login the FTP Server using the specified *IP Address*, *User ID,* and *Password*.	Correct FTP Server login credentials entered in ACP1 Web Server *Module Configuration* page.
408	FTP Pending File Count Warning	Indicates number of files queued for transfer to FTP Server meets or exceeds the *File Count Warning Level*.	Ensure FTP Transfer is enabled. Correct connection problem to FTP Server.
410	FTP Transfer Queue Full	Indicates number of files queued for transfer to FTP Server has reached maximum limit of 100 files.	
412	FTP File Transfer Error	Error was detected during file transfer to FTP Server. Data is retained on SD Card and file transfer retry is automatic.	Check status of FTP Server. Error is cleared when a file is successfully transferred to FTP Server.

Device Communication Errors

Error Code	Description	Comments	Recovery
415	CAMP Client Error	CAMP Client encountered a startup error or is unable to connect to one or more remote network devices	Confirm all remote devices are online and a network path exists between the ACP1 module and the remote devices. Check the configured IP Address for the CAMP Server devices.
420	Modbus Server Error	Modbus Server encountered a startup error.	Cycle power to the ACP1 module. Recompile and download the application program to the ACP1. If the problem continues contact CTI Support.
425	Modbus Client Error	Modbus Client encountered a startup error or is unable to connect to one or more Modbus Server devices.	Confirm all remote devices are online and a network path exists between the ACP1 module and the remote devices. Check the configured IP Address for the Modbus Server devices.
435	Data Exchange Error	Data Exchange Subscriber is unable to connect to one or more Publishers.	Check IP Address for each Publisher device set in *Global Binding* editor. Ensure Publishers are online and network path exists between the ACP1 and the remote devices.
460	EtherNet/IP Scanner Error	EIP I/O Scanner is unable to transfer I/O data with one or more I/O Adapters.	Confirm all EIP Adapters are online and a network path exists between the ACP1 module and devices. Check IP Address and I/O Assembly information in I/O Scanner fieldbus configuration.
465	EtherNet/IP Adapter Error	EIP Adapter encountered a startup error.	Cycle power to the ACP1 module. Recompile and download the application program to the ACP1. If the problem continues contact CTI Support.
470	EtherNet/IP Tag Client Error	EIP Tag Client is unable to read/ write data from one or more Servers.	Confirm all EIP Servers are online and a network path exists between the ACP1 module and devices. Check IP Address and specified tags to ensure names, data type, and length matches existing Controller tags in EIP Tag Client Server device. Add *Diagnostic/Control* variables to read General and Extended Error status.

500	MQTT Client Error	MQTT Client could not make a TCP connection to MQTT Broker or failed to complete a Publish/Subscribe service request.	Confirm MQTT Broker Is online and a network path exists between ACP1 module and device. Check the configured IP Address for the MQTT Broker device. Use *MQTTStatus* function block to determine the cause of the error.

Protocol Error Codes

Protocol Error Codes are returned by the protocol tasks. They indicate problems in sending or receiving messages via the designated protocol. Server protocol tasks return error codes to their clients within the protocol messages. Client protocol tasks allow Protocol Error Codes, including those returned by servers, to be mapped to a Host Controller memory address.

Similarly, Publisher protocol tasks may return error codes to their Subscribers. Subscriber protocol tasks allow error codes to be mapped to a Host controller memory address.

CAMP Client Error Codes

The following table describes the errors that CAMP Client may detect in the reply request or in attempting to communicate with another device that implements a CAMP Server.

Error Code		Description	Corrective Action
Decimal	**Hex**		
107	6B	Read Word Count Error	The number of words returned in the reply does not equal the number of words requested. This indicates a server error. Contact CTI Support
123	7B	Missing Delimiter Detected	The reply is missing a message delimiter. This Indicates a server error. Contact CTI Support.
116	74	Request contains Invalid Checksum	Usually results from transmission error. Retry the request. This may result from a checksum generation error. Contact CTI Support.
124	7C	Reply contains Invalid Checksum	
117	75	Request contains Invalid Command Type	This indicates an error in CTI Protocol Manager. Contact CTI Support.
125	7D	Reply contains Invalid Command Type	
118	76	Invalid Character in Request	The protocol accepts only characters 0-9, A-F, ?, [,] This indicates a Protocol Manager error. Contact CTI Support.
126	7E	Invalid Character in Reply	
119	77	Odd Number Of Characters In Request	The protocol requires that the message contain an even number of characters. This indicates a Protocol Manager error. Contact CTI Support.
127	7F	Odd Number Of Characters In Reply	
128	80	Invalid Error Type in Reply	The reply contains an invalid error type character. This indicates a server error. Contact CTI Support.
129	81	Word Count Error in Request	The number of data words in the message does not match the number of words specified. This indicates a Protocol Manager error. Contact CTI Support.
130	82	Word Count Error in Reply	
131	83	Write Request to Invalid Address 0	The CAMP Client Write Request message specified an invalid address offset of “0”. This indicates a CAMP Client error. Contact CTI Support.

Error Code		Description	Corrective Action
Decimal	Hex		
132	84	Write Request Word Count Error	The number of data words written does not match the number of words reported in the reply. This indicates a server error. Contact CTI Support. This indicates a CAMP Client error. Contact CTI Support.
143	8F	Read Request contains Word Count of 0	A Read Request message contains a “0” in the “number of data words” field. This indicates a CAMP Client error. Contact CTI Support.
144	90	Unknown Address Class	An unknown PLC memory type was specified in the request message. This indicates a CAMP Client error. Contact CTI Support.
145	91	Read Request contains Word Count greater than 256	A Read Request message contains an invalid Word Count value greater than 256. This indicates a CAMP Client error. Contact CTI Support.
146	92	Message ID Mismatch	The Message ID of the reply does not match the Message ID of the command message. This could occur if the reply arrives after a timeout and is interpreted as a reply to the next request. If this error persists, increase the timeout period.
152	98	Unable to connect to remote device	This could be caused by one of the following: • The remote device is offline or powered down. • The maximum number of TCP connections to the remote device has been reached, • The IP Address specified in the configuration does not match the IP address of the remote device, • There is no network path between the ACP1 module and the remote device.
155	9B	Error Reading Ethernet	The CAMP Client is unable to read data from the Ethernet interface. If this error persists, contact CTI Support.
156	9C	Error Writing Ethernet	The CAMP Client is unable to write data to the Ethernet Interface. If this error persists, contact CTI Support.
157	9D	Timeout Error	A reply from the remote device was not received within the timeout period specified in the configuration. This could be caused by one of the following: • The remote device is offline (when using UDP). • The IP Address specified in the configuration does not match the IP Address of the remote device (when using UDP). • There is no network path between the ACP1 module and the remote device (when using UDP). • The configured timeout value is too small. It does not allow enough time for the remote device to respond.

Error Code		Description	Corrective Action
Decimal	Hex		
172	AC	Read Request contains Invalid Address	The CAMP Server could not access one or more PLC Address locations specified in the request message. This usually occurs when the request exceeds the maximum address available in the host controller. Consider the following corrective actions: • Change the client request to change the address to be read • If the client is reading a block of data addresses, ensure that the number of addresses requested does not cause the maximum address to be exceeded. • Change the Host Controller memory configuration so that the requested addresses are available.
173	AD	Write Request contains Invalid Address	
187	BB	CAMP Server Host PLC in Fatal Error	The CAMP Server could not complete the request because its Host PLC is in Fatal Error state.
188	BC	CAMP Server Unable to Communicate with Host PLC	The CAMP Server could not complete the request due to failed communications with its Host PLC.
189	BD	CAMP Server is Not Enabled	The CAMP Server could not complete the request because it is not enabled.
190	BE	Request to Write 32-bit data contains Invalid Word Count	A CAMP Client Request to write 32-bit data words (REAL or Long Integer values) contains incorrect data. Two data words are required for each 32-bit value.
191	BF	Request to Write Data to a Read-only PLC Address	The PLC memory type specified in Address field of a CAMP Client Write Request specifies a Read-only address in the PLC.
198	C6	Request for Data that is not included in Data Cache while Host PLC unavailable	The 2500P-ECC1 CAMP Server could not complete the Read Request because one or more PLC Address locations are not included in the current Data Cache configuration and cannot be read from Host PLC. This can occur due to failed communications with Host PLC or when the Host PLC is in Fatal Error.
199	C7	CAMP Request Limit has been reached	The CAMP message contains a number of request packets in excess of the allowed limit. This indicates a CAMP Client error. Contact CTI Support.
200	C8	CAMP Server Data Cache Full	The 2500P-ECC1 CAMP Server could not complete the request because the PLC Address locations are not currently included in the Data Cache and is unable to add additional points to the Data Cache. This should occur very rarely but is possible when several CAMP Clients are communicating with a single CAMP Server. Items are removed from the Data Cache after 60 seconds if they are not accessed. Retry after a short timeout. If this error persists, contact CTI Support.

Modbus Server Error Codes

The following error codes may be returned by the Modbus Server.

Error Code	Description	Comments
1	Unsupported Modbus function	The function code received in the query is not supported by the server. This can result from the following conditions: • The client is requesting a function code not supported by the ACP1 Modbus Server. See Section 7.1.3 for a list of supported function codes. • The configured data blocks do not support the requested function code. For example, the client is attempting to read input registers but no data block of containing input registers exists.
2	Invalid Modbus address	The Modbus query attempts to access an address that is not within the range of Modbus addresses that have been configured for the requested Modbus data type. This can result from the following conditions: • The client is erroneously requesting an unsupported address. • The server data block is configured incorrectly. The starting address of the data block is in error or the size is too small.
3	Invalid Modbus value	A value contained in the query data field is not an allowable value for the slave. This indicates a fault in the message structure of the Modbus request so that the value in a particular field is not valid for its purpose. This error indicates a problem with the Modbus Client.
4	Modbus Server failure	An unrecoverable error occurred while the server was attempting to perform the requested action. If this error persists, contact CTI Support.

Modbus Client Error Codes

Error Code	Description	Comments
0	Request Successful	No Error
1	Unsupported Modbus function	The function code in the query is not supported by the Modbus slave device. Some Modbus devices may return this error if the function is not allowed for the data you are attempting to access, for example attempting to write data to which you have only read access. You should review user documentation for the device and make corrections as necessary.
2	Invalid Modbus address	The Modbus query attempted to access an address that is not a valid Modbus address for the device. Some Modbus slave devices may also return this code to indicate that that access is inhibited (for example, write protected).
3	Invalid Modbus value	A value contained in the query data field is not an allowable value for the Modbus slave device.
4	Modbus Server failure	An unrecoverable error occurred while the Modbus device was attempting to perform the requested action.
6	Modbus Server Busy	The Modbus slave device was unable to service the request because it was busy with other tasks.
8	Data Parity Error	This error is not expected, but is included for completeness. This error indicates that the extended memory area failed to pass a consistency check. The ACP1 Modbus Client does not support FC20 or FC21 that can return these error codes.
10	Invalid Gateway Path	This response is returned by a Modbus Ethernet-to- Serial gateway. It indicates that the gateway was unable to allocate a path to the device. This typically means the Unit ID (used to select the path) does not match the address of a slave on the serial network.
11	Gateway Target Failed	This response is returned by a Modbus Ethernet-to- Serial gateway. It indicates that no response was obtained from the target device. The device may not be connected, may be powered down, or the serial parameters may be configured incorrectly.
128	Communication Timeout	This indicates that the target Modbus device failed to respond within the designated timeout period.
129	Bad CRC16	Indicates a data transmission error. Reported only by Modbus slave devices attached to serial links. Modbus TCP/IP protocol does not contain a Modbus CRC.
130	Lost Connection	The TCP connection to the device has been lost.

EtherNet/IP Tag Client Error Codes

The EtherNet/IP Tag Client allows the ACP1 to read/write Controller tags in a RSLogix PLC. “Tags” are one or more data elements of a single Data Type. When more than one data element is included, the PLC tag is configured as an array with enough elements to store all of the data. The ACP1 can access PLC tags up to 500 bytes in length. It is possible to configure larger tags in the PLC, but these “extended-length” tags are not supported by the ACP1.

The following tables list the errors that may be returned by the EIP Tag Client. Error codes are listed in Hex format with decimal equivalents shown in parentheses.

Read Tag Errors

General Status	Extended Status	Comments
0x04 (4)	0x0000 (0)	Syntax Error in decoding Network Path
0x05 (5)	0x0000 (0)	Request Path Destination is unknown. This usually indicates the specified tag name does not exist in the Controller scope.
0x06 (6)	0x0000 (0)	Insufficient Packet Space. The number of bytes allocated for storage of response data is too small.
0xFF (255)	0x2105 (8453)	Access beyond end of object. The number of elements requested + offset exceeds number of elements in Tag.

Write Tag Errors

General Status	Extended Status	Comments
0x04 (4)	0x0000 (0)	Syntax Error in decoding Network Path
0x05 (5)	0x0000 (0)	Request Path Destination is unknown. This usually indicates the Tag Name does not exist in the Controller scope.
0x10 (16)	0x2101 (8449)	Device State Conflict: Keyswitch position. The requestor is attempting to change force data in HARD RUN mode.
0x10 (16)	0x2102 (8450)	Device State Conflict: Safety status. The controller is in a state where Safety Memory cannot be modified.
0xFF (255)	0x2105 (8453)	Access beyond end of object. The number of elements requested + offset exceeds number of elements in Tag.
0xFF (255)	0x2107 (8455)	Data type of tag is invalid or not supported.

CIP General Status Codes

The following table list all of the General Status codes that can be returned by the CIP protocol. These codes may be reported in the I/O Scanner and/or Tag Client diagnostic error variables. This list can be helpful for debugging connection issues, especially when analyzing data from a network capture program (i.e. Wireshark).

General Status Code		Description	Comments
Hex	Dec		
0x00	0	Success	Requested service was successfully performed.
0x01	1	Connection failure	A connection related service failed along the request path. When detected, the Extended Status value is returned
0x02	2	Resource unavailable	Resources needed for the requested service were unavailable.
0x 03	3	Invalid parameter value	See Status Code 20 (the preferred value for this condition)
0x04	4	Path segment error	The path segment identifier or segment syntax was not understood by the processing node.
0x 05	5	Path destination unknown	The path references an object class, instance, or structure element that does not exist in processing node.
0x06	6	Partial transfer	Only part of the expected data was transferred.
0x07	7	Connection lost	The messaging connection was lost.
0x08	8	Service not supported	The requested service is not supported for the specified Class/Instance.
0x09	9	Invalid attribute value	Invalid attribute data was detected by processing node.
0x0A	10	Attribute list error	An attribute in Get_Attribute_List or Set_Attribute_List response has a non-zero status.
0x0B	11	Already in requested mode/state	The object is already in the mode/state requested by the service.
0x0C	12	Object state conflict	The object cannot perform the requested service in the current mode/state.
0x0D	13	Object already exists	The requested instance of object to be created already exists.
0x0E	14	Attribute not settable	A request to modify a non-changeable attribute was received.
0x0F	15	Privilege violation	A permission privilege check failed.
0x10	16	Device state conflict	The current mode/state prohibits execution of the requested service.
0x11	17	Reply data too large	The reply data is larger than the allocated response buffer.
0x12	18	Fragmentation of a primitive value	The service specified an operation that will fragment a primitive value (such as half of a REAL data type).
0x13	19	Not enough data	The service request did not include enough data to perform the specified operation.
0x14	20	Attribute unsupported	The attribute specified in the request is not supported by the destination run-time system.
0x15	21	Too much data	The service request supplied more data than expected.
0x16	22	Object does not exist	The specified object does not exist in the device.
0x17	23	Service fragmentation sequence inactive	The fragmentation sequence for this service is not currently active for this operation.
0x18	24	No stored attribute data	The attribute data of this object was not saved prior to the requested service.

General Status Code		Description	Comments
Hex	Dec		
0x19	25	Store operation failure	The attribute data of this object was not saved due to an operational failure during the attempt.
0x1A	26	Routing failed: request packet too large	The request packet was too large for retransmission on specified network and was aborted by the routing device.
0x1B	27	Routing failed: response packet too large	The response packet was too large for retransmission on specified network and was aborted by the routing device.
0x1C	28	Missing attribute list entry data	The service did not supply an attribute needed to perform the requested operation.
0x1D	29	Invalid attribute list	The service is returning the list of attributes with state information for the invalid attributes.
0x1E	30	Embedded service error	An embedded service operation resulted in an error.
0x1F	31	Vendor specific error	A vendor specific error has occurred. The Extended Status field of the Error Response specifies the error condition.
0x20	32	Invalid parameter	A parameter associated with the request does not meet the specification requirements.
0x21	33	Write-once value/media already written	An attempt to write to a write-once media (WORM drive or PROM) or modify a non-changeable value failed.
0x22	34	Invalid reply received	The response service code does not match the request service code or reply message is shorter than minimum reply size.
0x23	35	Buffer overflow	The message received is larger than the receiving buffer size. The entire message is discarded.
0x24	36	Message format error	The format of the received message is not supported.
0x25	37	Key failure in path	The Key Segment included as the first segment in path does not match the destination device. The object specific status indicates which part of the key check failed.
0x26	38	Path size invalid	The size of the path sent with the Service Request is invalid: either too much or too little routing data was included.
0x27	39	Unexpected attribute in list	An attempt was made to set an attribute that cannot be modified at this time.
0x28	40	Invalid Member ID	The Member ID included in the request does not exist in the specified Class/Instance/Attribute.
0x29	41	Member not settable	A request to modify a non-changeable member was received.
0x2A	42	Group 2 Only server general failure	The error is only reported by DeviceNet Group 2 Only servers with <=4K code space in place of “Service not supported”, “Attribute not supported”, or “Attribute not settable “ errors.
0x2B	43	Unknown Modbus error	A CIP to Modbus translator received an unknown Modbus Exception Code.
0x2C	44	Attribute not read	A request to get a non-readable attribute was received.
0x2D	45	Instance not deleted	The requested object instance cannot be deleted.
0x2E	46	Service not supported for specified path	The object does not the designated application path for the specified service.
0x2F-0xCF	47-317	Reserved	Reserved by CIP for future extensions.

General Status Code		Description	Comments
Hex	Dec		
0xD0-0xFF	318-255	Served for Object Class specific errors	Object Class specific errors when none of the other Error Codes in this table accurately reflect the error that was encountered.

CIP UCMM Extended Status Codes

The following table list all of the Extended Status codes that can be returned by the CIP Unconnected Message Manager (UCMM). These status codes are generally returned during processing of a "Forward_Open" request from an EtherNet/IP Client when attempting to establish an I/O connection with a Server.

These errors may be reported in the I/O Scanner diagnostic error variables.

UCMM Status Code		Description	Comments
Hex	Dec		
0x100	256	Connection in Use or Duplicate Forward Open	The Originator is attempting to make a connection to a Target to which the Originator has already established a connection. This could be caused by poor network traffic. Check the cabling, switches and connections.
0x103	259	Transport Class or Trigger Not Supported	The Target does not support Class 1 / Class 3 transports or specified trigger-type. CTI products support only Cyclic triggers for I/O messages.
0x106	262	Ownership Conflict	The connection cannot be established since the required resources are already allocated to a different connection. An example of this would be only one exclusive owner connection can control an I/O module output point. Check to see if other Scanner devices are connected or verify that Multicast is supported by the Adapter (if Multicast is selected). This error could be caused by poor network traffic. Check the cabling, switches and connections.
0x107	263	Connection Not Found	This occurs if a device sends a Forward Close on a connection that is inactive. This could occur if one of these devices has powered down or if the connection timed out due to poor network traffic. Check the cabling, switches and connections.
0x108	264	Invalid Network Connection Parameter	One of the parameters specified in the Forward Open (such as Connection ID, Connection Type, Connection Priority, Redundant Owner, or Fixed/Variable) is not supported.
0x109	265	Invalid Connection Size	The Target or Router does not support the specified connection size. This could occur when the assembly size configured for the Target does not match the required size for a fixed connection. CTI products support only fixed connection sizes. Check the manufacturer documentation to verify the connection size required by the device. Note that most devices specify this value in terms of bytes.

UCMM Status Code		Description	Comments
Hex	Dec		
0x10A-0x10F	266-271	Reserved	Reserved by CIP
0x110	272	Target for Connection Not Configured	The Target has received a request for connection that has not been configured. This could occur if the device has powered down or if the connection timed out. This could be caused by poor network traffic. Ethernet/IP
0x111	273	RPI Not Supported	The Target does not support the specified Requested Packet Interval (RPI) for O->T or T->O connection. This status code can also indicate an unsupported connection timeout produced by the connection timeout multiplier. Although CTI products allow RPI to be set as low as 1 msec, I/O data cannot be produced or processed any faster than the product cycle time of the application.
0x112	274	RPI Values Not Acceptable	The Requested Packet Interval (RPI) values in the Forward Open message are outside the range required by the Target, or the Target is producing at a different interval.
0x113	275	Out of Connections	All CIP connections available to Target are already in use. The maximum number of connections supported by the Connection Manager is already in use.
0x114	276	Vendor ID or Product Code Mismatch	The compatibility bit was set in the Forward Open message but the Product Code or Vendor ID specified in the electronic key logical segment does not match the Product Code or Vendor ID of the device.
0x115	277	Device Type Mismatch	The compatibility bit was set in the Forward Open message but the Device Type specified in the electronic key logical segment does not match the Device Type of the device.
0x116	278	Revision Mismatch	The compatibility bit was set in the Forward Open message but the major/minor revisions specified in the electronic key logical segment do not correspond to a valid device revision.
0x117	279	Invalid Produced or Consumed Application Path	The Connection ID specified for the O->T or T->O connection in incorrect or not supported by the Target.
0x118	280	Invalid Configuration Application Path	The Connection ID specified the Configuration data is incorrect or not supported by the Target.
0x119	281	Non-Listen Only Connection Not Opened	The Originator requested a Listen-only connection when a Non-Listen Only connection is not already established. CTI products do not support Listen-only connections as Scanner or Adapter.
0x11A	282	Target Object Out of Connections	The maximum number of connections supported by this instance of the Target object has been exceeded.
0x11B	283	Production Inhibit Time is greater than RPI Value	The T->O RPI is smaller than the T->O Production Inhibit Time. Consult the manufacturer's documentation as to the minimum rate that data can be produced and configure the RPI to greater than this value.

UCMM Status Code		Description	Comments
Hex	Dec		
0x11C	284	Transport Class Not Supported	The Transport Class requested in the Forward Open is not supported by the Server. Only Class 1 and Class 3 messages are supported by the CTI products.
0x11D	285	Production Trigger Not Supported	The Server does not support the trigger to produce messages requested in the Forward Open is not supported. In Class 1, only Cyclic and Change of state are supported in
0x11E	286	Direction Not Supported	The Direction requested in the Forward Open is not supported by the Target.
0x11F	287	Invalid O->T Network Connection FIXVAR	The O-> T Fixed/Variable flag specified in the Forward Open is not supported. Only Fixed is supported in CTI products.
0x120	288	Invalid T->O Network Connection FIXVAR	The T-> O Fixed/Variable flag specified in the Forward Open is not supported. Only Fixed is supported in CTI products.
0x121	289	Invalid O->T Network Connection Priority	The O-> T Network Connection Priority specified in the Forward Open is not supported. CTI products support Low, High, Scheduled, and Urgent Priority settings.
0x122	290	Invalid T->O Network Connection Priority	The T->O Network Connection Priority specified in the Forward Open is not supported. CTI products support Low, High, Scheduled, and Urgent Priority settings.
0x123	291	Invalid O->T Network Connection Type	The O-> T Network Connection Type (Unicast or Multicast) specified in the Forward Open is not supported. Most devices support only Unicast for O->T (Output) data. CTI products support both Unicast and Multicast connections.
0x124	292	Invalid T->O Network Connection Type	The T->O Network Connection Type (Unicast or Multicast) specified in the Forward Open is not supported. CTI products support both Unicast and Multicast connections.
0x125	293	Invalid O->T Network Connection Redundant Owner Flag	The O->T Network Connection Redundant Owner flag specified in the Forward Open is not supported. Only Exclusive owner connections are supported in CTI products.
0x126	294	Invalid Configuration Size	The Configuration data sent in the Forward Open does not match the size specified or is not supported by the Adapter. The Target usually returns an additional value that specifies the maximum size allowed for this data. This information is displayed on the EtherNet/IP statistics page of the CTI product web server
0x127	295	Invalid Originator to Target Size	The O->T (Output data) size specified in the Forward Open does not match what is expected by the Target. Consult the documentation of the Adapter device to verify the size. Note if the Run/Idle header is included in the O->T data size. The Run/Idle Header can be enabled/disabled as needed to match the Adapter setting when a CTI product is acting as the I/O Scanner. When a CTI product is serving as the Adapter, it always requires the Run/Idle header to be included in the O->T data. If the I/O Scanner does not have the option to send the Run/Idle Header, then 4 additional bytes must be added to the O->T size (with first 4 bytes unused).

UCMM Status Code		Description	Comments
Hex	Dec		
			Some devices publish the expected T->O size as an additional value when the error is returned. If available, this information is displayed on the EtherNet/IP statistics page of the CTI product web server. NOTE: The Target device may return this error if the O->T Connection ID is invalid for I/O Message data.
0x128	296	Invalid Target to Originator Size	The T->O (Input data) size specified in the Forward Open does not match the Target configuration. Consult the documentation of the Adapter device to verify the size. Note if the Run/Idle header is included in the T->O data size. The Run/Idle Header can be enabled/disabled as needed to match Adapter settings when a CTI product is acting as the I/O Scanner. When CTI product is serving as the Adapter, it does not include the Run/Idle header in the T->O data. If the I/O Scanner does not have the option to exclude the Run/Idle Header from T->O data, then 4 additional bytes must be added to the T->O size (with first 4 bytes unused). Some devices publish the expected O->T size as an additional value when the error is returned. If available, this information is displayed on the EtherNet/IP statistics page of the CTI product web server. NOTE: The Target device may return this error if the T->O Connection ID is invalid for I/O Message data.
0x129	297	Invalid Configuration Application Path	The Configuration Instance contained a size other than zero for Adapter where Configuration data sent with the Forward Open message is not supported.
0x12A	298	Invalid Consuming Application Path	The Consuming (O->T) Path is not present in the Forward Open message from the I/O Scanner, or the O->T Connection ID is incorrect for the Adapter configuration.
0x12B	299	Invalid Producing Application Path	The Producing (T->O) Path is not present in the Forward Open message from the I/O Scanner, or the T->O Connection ID is incorrect for the Adapter configuration.
0x12C	300	Configuration Symbol Does Not Exist	The Originator attempted to connect to a configuration tag name that does not exist in the Target.
0x12D	301	Consuming Symbol Does Not Exist	The Originator attempted to connect to a consuming tag name that does not exist in the Target.

UCMM Status Code		Description	Comments
Hex	Dec		
0x12E	302	Producing Symbol Does Not Exist	The Originator attempted to connect to a producing tag name that does not exist in the Target.
0x12F	303	Inconsistent Application Path Combination	The data specifying the Producing, Consuming, and Configuration paths is inconsistent.
0x130	304	Inconsistent Consume Data Format	The contents of the data does not match the format for the Consumed data segment.
0x131	305	Inconsistent Produce Data Format	The contents of the data does not match the format for the Produced data segment.
0x132	306	Null Forward Open Function Not Supported	The Target does not support the function requested in the NULL Forward Open message. This could include functions such as “Ping device”, “Configure device application”, etc.
0x133	307	Connection Timeout Multiplier Not Acceptable	The Connection Multiplier specified in the Forward Open message is not supported by the Target. Consult the manufacturer’s documentation for the acceptable timeout and multiplier values. CTI products use a fixed Connection Multiplier of ‘4’.
0x203	515	Connection Timed Out	An I/O message was received on a connection that has already timed out. Connections timeout when no I/O message is received in the time period specified by RPI Rate X ‘Connection Multiplier’ (fixed at ‘4’ for CTI products). This could be caused by poor network traffic. Check the cabling, switches and connections.
0x204	516	Unconnected Send Timed Out	An Unconnected Send message has been sent, and no response is received within the specified timeout period. This usually indicates poor network traffic. Check the cabling, switches and connections.
0x205	517	Unconnected Send Parameter Error	The Connection Tick Time/Connection Timeout period specified in the Forward Open or Forward Close message is not supported.
0x206	518	Message Too Large for UCMM	The Unconnected Send message is too large to be sent on the network.
0x207	519	UCMM Acknowledge without Reply	The Acknowledge message to an Unconnected Send was received, but no data response occurred.
0x208-0x300	520-768	Reserved	Reserved by CIP
0x301	769	No Buffer Memory Available	The Connection memory buffers in the Target device are full. This can usually be corrected by reducing the frequency of the messages being sent to the device and/or reducing the number of connections to the device
0x302	770	Network Bandwidth Not Available for Data	The Producer device cannot support the specified RPI rate when the connection has been configured with Schedule priority. Reduce the message rate (increase RPI value) or consult the manufacturer's documentation.
0x303	771	No Consumed Connection ID Filter Available	The Consumer device does not have an available consumed_connection_id filter.

UCMM Status Code		Description	Comments
Hex	Dec		
0x304	772	Not Configured to Send Scheduled Priority Data	A device configured for a Scheduled Priority message cannot send the data at the scheduled time slot.
0x305	773	Schedule Signature Mismatch	The connection scheduling information in the Originator does not match the scheduling information in the Target.
0x306	774	Schedule Signature Validation Not Possible	The connection scheduling information in the Originator cannot be validated by the Target network.
0x307-0x310	775-778	Reserved	Reserved by CIP
0x311	779	Port Not Available	The Port number specified in a Port Segment is not available or does not exist. Consult the documentation of the device to verify the correct port number.
0x312	780	Link Address Not Valid	The Link address specified in the Port Segment is not valid for the Target network. Consult the documentation of the device to verify the Port configuration.
0x315	789	Invalid Segment in Connection Path	The Target cannot decode the Connection Path. This could be caused by an unrecognized or unexpected Segment Type (such as Configuration Instance) in the 'Forward Open' request.
0x316	790	Forward Close Service Connection Mismatch	The Connection path in the Forward Close message does not match the Connection Path configured in the Target.
0x317	791	Scheduling Not Specified	The Schedule Network Segment was not present or the value was invalid.
0x318	792	Link Address to Self Invalid	The Link Address in the Port Segment points back to the Originator device.
0x319	793	Secondary Resource Unavailable	This occurs in a Redundant system when the Secondary connection request is unable to duplicate the Primary connection request. CTI products do not support Redundant Owner connections.
0x31A	794	Rack Connection Already Established	A module connection is refused because some of the data requested is already included in an existing rack connection.
0x31B	795	Module Connection Already Established	A rack connection is refused because some of the data requested is already included in an existing module connection.
0x31C	796	Miscellaneous	This error is used when there is no other applicable code for the error condition. Consult the device documentation or contact CTI Support if this error persist.
0x31D	797	Redundant Connection Mismatch	At least one of these parameters does not match when attempting to establish a Redundant Owner connection: O->T RPI, O->T Connection parameters, T->O RPI, T->O Connection parameters, Transport Type, and Trigger. CTI products do not support Redundant Owner connections.
0x31E	798	No more User Configurable Link Resources Available in the Producing Module	The Target device has no more available Consumer connections available for a Producer.

UCMM Status Code		Description	Comments
Hex	Dec		
0x31F	799	No User Configurable Link Resources Configured in the Producing Module	The Target device has no Consumer connections configured for a Producer connection.
0x800	2048	Network Link Offline	The Network Link Path is invalid or not available.
0x801-0x80F	2049-2063	Reserved	Reserved by CIP
0x810	2064	No Target Application Data Available	The Target application has no valid data to produce.
0x811	2065	No Originator Application Data Available	The Originator application has no valid data to produce.
0x812	2066	Node Address has changed since the Network was scheduled	A Router has a different node address than the value configured in the original connection.
0x813	2067	Not Configured for Off-Subnet Multicast	The Producer has been requested to support a Multicast connection for a Consumer on a different subnet, and the Producer does not support this functionality.
0x814	2068	Invalid Produced/Consumed Data format	The format of the produced and/or consumed data is not consistent with Information in the data segment. Status Codes 0x130 and 0x131 are typically used to report this situation in most devices.

Network Data Exchange Subscriber Error Codes

The following error codes may be returned by the Network Data Exchange (Variable Binding) Subscriber.

Error Code	Description	Comments
0	No Error	
1	Re-establishing lost connection	The Subscriber is attempting to reconnect to the Publisher.
2	TCP Error Occurred	The Subscriber experienced a non-recoverable TCP error, causing the connection to be closed.
3	Timeout Error	Nothing has been received from the Publisher within the expected time interval.
4	Other error detected	General connection error

NOTE:

If a connection is interrupted, the Network Data Exchange Subscriber will automatically attempt to reconnect. Consequently, the error codes will typically continue to cycle until the connection is good.

If your Host Controller logic is performing an action based on error, you should condition the action when the controller memory address associated with the error status is non-zero rather than the monitoring individual error codes.

MQTT Client Error Codes

The following error codes may be returned by the ***MQTTStatus*** function block.

MQTT Standard Error Codes returned by MQTT Broker/Server:

Error Code	Description	Comments
0	No Error	
1	Connection Refused : Unacceptable Protocol Version	The server does not support the version level of the MQTT protocol requested by the client.
2	Connection Refused: Identifier rejected	The Client ID is not allowed by the broker/server. This usually means the specified Client ID is not included in the list of clients authenticated to connect to the broker/server.
3	Connection Refused: Server is unavailable	The network connection has been made of the MQTT broker service is unavailable.
4	Connection Refused: Bad username or password	Verify the Client ID, Username, and Password entered in MQTT Client configuration matches data in the broker password file.
5	Connection Refused: Not authorized	This error is returned when client attempts to access an unauthorized (or restricted) topic, service (publish or subscribe), or quality of service on the broker.

MQTT Status Codes set by ACP1:

Status Code	Description	Comments
1000	Waiting for connection	Connection is in progress.
1001	TCP connection error	The MQTT Client was unable to make a TCP connection to the specified IP address and port number.
1002	Connection timeout	The MQTT Client could not communicate with the specified IP address.
1003	Invalid telegram received	An unexpected telegram not matching MQTT specifications was received via port 1883.
9999	Invalid Connection ID	The invalid Connection ID was entered in the MQTT Client configuration. The Connection ID must exist (at least one char in length) and is restricted to a maximum length of 23 characters when using MQTT V3.1.

Firmware Update Error Codes

If an error occurs during the firmware update procedure, the procedure will stop and wait for corrective action. An error code will be displayed on the Multi-Segment Display. The following table describes the error codes and the corrective action.

Error Code	Description	Corrective Action
E16	Firmware update failed	Rerun the firmware update procedure. If the error persists, contact CTI Support
E17	Unable to write the firmware update status flag.	Rerun the firmware update procedure. If the error persists, contact CTI Support.
E18 – E25	Unable to locate or execute the firmware file loader	Contact CTI Support.
E26	An error was encountered when attempting to open the firmware update file	Reformat the SD Card, copy the firmware file to the SD Card, and rerun the firmware update procedure.
E27	The firmware update file could not be read.	Delete the firmware update file, re-copy the firmware update to the SD Card, and rerun the firmware update procedure. If the error persists, try using a different SD Card for the firmware update.
E28	The SD Card directory could not be read	Reformat the SD Card, copy the firmware update file to the SD Card, and rerun the firmware update procedure.
E29	No firmware update (.ffl) file could be found on the SD Card or more than one firmware update file was found.	One and only one firmware update file is allowed in the root directory of the SD Card. Take the necessary action to ensure that the desired firmware update file (and only that file) is located in the root directory. Once this has been done, rerun the firmware update procedure.
E30	Unsupported SD card inserted	The ACP1 module supports an SDSC (SD Standard Capacity) or an SDHC (SD High Capacity) card.
E31	Unused	
E32	SD Card memory region is not available	Rerun the firmware update procedure. If this error persists, contact CTI Support.
E33 – E73	Error erasing, writing, or verifying, flash.	Retry the firmware download procedure. If the error persists, contact CTI Support.
E74 - E79	The file transferred to the ACP1 module does not appear to be a valid ACP1 Firmware Update File.	Ensure that the file you transferred is a firmware update file. If so, retry the firmware update procedure. If not, obtain a valid firmware update file and repeat the firmware update procedure. Also check Firewall settings on pc/laptop. Try temporarily disabling the Firewall just during firmware download.
E80	An error occurred while the firmware update file was being transferred to the controller.	**Ethernet Transfer Method** The network path between the PC and the ACP1 module may have been disrupted. Retry the firmware update procedure.

Error Code	Description	Corrective Action
		SD Card Transfer Method The SD Card may have failed. Retry the firmware update procedure. If the error persists, retry using a different SD card.
E81 - E99	An error occurred while processing the firmware update file. This usually indicates the file is corrupted.	Retry the firmware update procedure. If the error repeats, obtain a valid firmware update file and rerun the firmware update procedure.
E168 – E199	An error occurred while updating the FPGA firmware. This usually indicates the file is corrupted.	Retry the firmware update procedure. If the error repeats, obtain a valid firmware update file and rerun the firmware update procedure. If the error still occurs, contact CTI Support.

APPENDIX B: IP ADDRESS INFORMATION

IP Address Nomenclature

IP Address

Every host interface on a TCP/IP network is identified by a unique IP Address. This address is used to uniquely identify the host device, such as a workstation or communications module, and the network to which the host belongs.

Each IP Address consists of 32 bits, divided into four 8 bit entities called *octets*. An IP Address is expressed in *dotted notation,* with each octet expressed as its decimal equivalent. See the example below.

Notation	Octet 1	Octet 2	Octet 3	Octet 4
Binary	11000000	11011111	10110001	00000001
Decimal	192	223	177	1

Although an IP Address is a single value, it contains two types of information: the *Network ID* and the *Host ID*. The Network ID identifies the IP network to which the host belongs. The Host ID identifies a specific IP host on that IP network. All IP hosts on a particular local area network must have the same network ID. Each IP host on a particular local area network must use a unique Host ID.

Address Classes

The Internet community originally defined network classes to accommodate networks of varying sizes. The network class can be discerned from the first octet of its IP Address.

The following table summarizes the relationship between the first octet of a given address and its Network ID and Host ID fields. It also identifies the total number of Network IDs and Host IDs for each address class that participates in the Internet addressing scheme.

Class	First Octet Value*	Network ID	Host ID	Number of networks	Number of hosts per net
A	1-126	First Octet	Last 3 Octets	126	16,777,214
B	128-191	First 2 Octets	Last 2 Octets	16,384	65,534
C	192-223	First 3 Octets	Last Octet	2,097,151	254

* Address 127 is reserved for loopback testing and inter-process communication on the local computer; it is not a valid network address. Addresses 224 – 239 are used for Class D (IP multicast).

Subnet Mask

Used alone, the designation of network classes is very inflexible. For example, a Class A network assigns a large number of host devices to the same IP network; potentially reducing performance, limiting topology, and compromising network security. An additional entity, the Subnet Mask, provides means of dividing a large IP network into a collection of smaller networks called subnets.

The Subnet Mask is a collection of 32 bits that distinguish the network ID portion of the IP address from the host ID. Network masks are implemented by assigning a 1 to bits that belong to the network ID and 0's to the bits that belong to the host ID. Like the IP Address, the resulting 32-bit value is expressed in dotted decimal notation. See the example below.

Bits for Network Mask				**Network Mask in Dotted Decimal**
11111111	00000000	00000000	00000000	255.0.0.0 (default class A subnet mask)
11111111	11111111	00000000	00000000	255.255.0 (default class B subnet mask)
11111111	11111111	11110000	00000000	255.255.240.0 (subnetted class B network)
11111111	11111111	11111111	00000000	255.255.255.0 (default class C subnet mask)

For example: when the IP address is 172.54.177.97 and the subnet mask is 255.255.255.0, the Network ID is 172.54.177 and the Host ID is 97.

NOTE

*The binary representation of a Network Mask must be a single continuous block 1's followed by a contiguous block of zeroes. When entering the Network Mask in dotted decimal notation, you must ensure that this requirement is maintained. For example, a network mask of 255.247.0.0 is not valid because the binary equivalent (11111111111**0**111000000000000000) violates this rule.*

The Network Mask must allow at least two bits of host address. In addition, a network mask which causes the derived host ID to be 0 or a broadcast address (all Host ID bits set to 1) should not be used.

Using the Subnet Mask

For Class A, B, and C IP addresses, the IP Host uses the Subnet Mask to determine where to send an IP message. After deriving the Network ID and Host ID portion of the IP Address using the Subnet Mask, the IP Host compares the Network ID of the destination IP Address with the Network ID of the Host IP Address. If the Network IDs are the same, the message is sent to another Host on the local network. If the Network IDs are different, the message is sent to an IP Gateway, for routing to another network, if possible.

When you are configuring the IP Address of devices that must communicate on a local network, you must ensure that:

- The Subnet Mask of all devices is the same.
- The Network ID of all hosts is the same.
- The Host ID of each host is different.

If you are using Subnet Masks that are aligned with the IP Address octets, this can easily be done by examining the dotted decimal values. The octets of the IP Address where the corresponding octet of the Subnet Mask is 255 belong to the Network ID and the octets of the IP Address where the corresponding octet of the Subnet Mask is 0 belong to the Host ID.

For example, where the IP Address is 127.18.40.3 with a Subnet Mask of 255.255.0.0, the Network ID is 127.18 and the Host ID is 40.3.

IP Address	127	18	40	3
Subnet Mask	255	255	0	0
Network ID	127	18		
Host ID			40	3

However, if you are using a Subnet Mask that does not align with the octet boundaries, this is more difficult. You will need to perform a bitwise “and” calculation to arrive at the Network address. See the following illustration.

Assuming an IP Address of 127.18.40.3 and a Subnet Mask of 255.255.240.0, the following table illustrates the bitwise “and” operation. In essence, wherever the Subnet Mask bit is one, the corresponding IP Address bit is part of the Network ID.

Item	**Dotted Decimal**	**Binary Equivalent**			
		1st Octet	2nd Octet	3rd Octet	4th Octet
IP Address	127.18.40.3	**01111111**	**00010010**	**0010**1000	00000011
Subnet Mask	255.255.240.0	**11111111**	**11111111**	**1111**0000	00000000
Derived Network Address	127.18.32.0	**01111111**	**00010010**	**0010**0000	00000000

An easier way to determine this is to compare only non-aligned Subnet Mask octet with the corresponding octet of the IP Address. For example, since the Subnet Mask of the first two octets is 255.255, the first two octets of the Network ID are the same as the dotted decimal values (127.18) of the IP Address. However, since the third octet of the subnet mask is not 255 or 0, you must perform a bitwise ”and” calculation using the third octet of the IP Address and Subnet Mask.

This can be accomplished by using the windows calculator (Scientific View in Windows XP or Programmer view in Windows 7). Using this example, you would enter the value of the third octet (40), click on the "and" button, enter the Subnet Mask (240), and then click on the "=" button. The result, in this case, is 32. Thus the Network Address is 127.18.32.0.

Selecting an IP Address

In order to perform its functions, the 2500P-ACP1 requires a fixed IP Address. If you are connecting to an existing network, you should obtain an unused static IP Address and the Network Subnet Mask from the network administrator.

If you are establishing your own IP Addresses, you should select IP Addresses from a block of 'private" addresses established by the Internet Assigned Numbers Authority (IANA). The private address blocks are:

- 10.0.0.0 through 10.255.255.255 (Class A)
- 172.16.0.0 through 172.31.255.255 (Class B)
- 192.168.0.0 through 192.168.255.255 (Class C)

These addresses will not be forwarded by the Internet backbone routers; therefore, you are free to use any address in this group as long as it does not conflict with the usage by your local organization.

Selecting a Multicast Address

The address range of 239.0.0.0 thru 239.255.255.255 has been designated as an administratively scoped Multicast Address space (RFC 2365). Addresses in this range are designated for use by private multicast domains. They do not conflict with other multicast address spaces that are explicitly assigned by Internet Assigned Numbers Authority (IANA). Within this range, addresses 239.255.0.0 thru 239.255.255.255 are designated for the IPV4 multicast local scope.

If you are choosing a Multicast Address for a new factory floor application, you should choose a Multicast Address in the IPV4 local scope range (239.255.0.0 thru 239.255.255.255) unless you have a specific reason to do otherwise. You should verify there is no conflict with other Multicast Addresses being used locally.

In case you are using the 2500P-ACP1 module in an existing multicast application that uses a Multicast Address outside of the administratively scoped address space, the configuration program allows you to enter the complete range of assignable multicast addresses (224.0.0.1 thru 239.255.255.255).

For a current list of IANA assigned multicast addresses, see the IANA website:
www.iana.org/assignments/multicast-addresses/

APPENDIX C: SERIAL PORT DETAILS

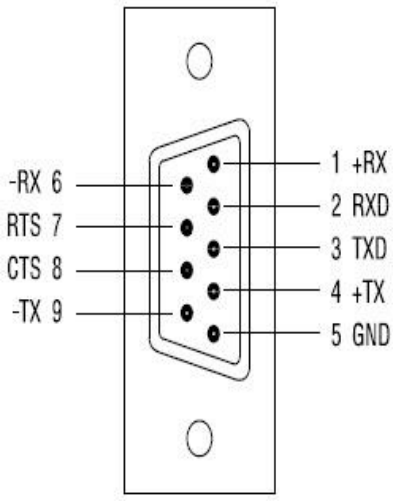
Protocols

The 2500P-ACP1 provides communications protocols for connection of RS-232/RS-422 compatible devices to the serial port. Only one serial port protocol may be active at any time.

Function	Description
Modbus RTU Master	Allows ACP1 to function as Modbus RTU Master Supports up to 32 different Modbus Requests to be configured
Modbus RTU Slave	Allows ACP1 to function as Modbus RTU Slave
General ASCII	Sends and receives ASCII characters. Can be used to interface to proprietary serial protocol messages. Data is transmitted and received based on the application logic.

Hardware

The male DB9 connector on the front panel provides the serial port interface. The cable used for connection to the external device must attach to the pins used by the selected electrical interface.



RS-232 (Subset) Pinout:

Pin	Signal	Description
2	RXD	Receive Data
3	TXD	Transmit Data
5	GND	Signal Ground
7	RTS	Request to Send (optional)
8	CTS	Clear to Send (optional)

RS-422 Pinout:

Pin	Signal	Description
1	+RX	Receive Data (+)
4	+TX	Transmit Data (+)
6	-RX	Receive Data (-)
9	-TX	Transmit Data (-)

Software Configuration

All serial port parameters, including the electrical interface (RS232 or RS422), are set by software configuration via ***CTI Workbench***. Parameters are specified by an ASCII string containing descriptors and associated values. This string is used to setup the serial port as required by serial protocol drivers.

The following table lists serial port setup parameters:

Parameter	Descriptor	Valid Values	Default Value
Port	PT	ALWAYS = 1	1
Baud Rate	BD	1200, 2400, 4800, 900, 19200, 38400, 57600, 115200	9600
Data Bits	DB	7, 8	8
Stop Bits	SB	1, 2	1
Parity	PY	None (N), Even (E), Odd (O)	ASCII (N) Modbus RTU (E)
Flow Control	FC	No (N), Yes (Y) 'Y' enables Hardware RTS-CTS handshake (CTS must be TRUE to send)	N
Interface	IF	RS232, RS422	RS232

This is an example string to setup the serial port:

PT=1 BD=19200 DB=8 SB=1 PY=N FC=N IF=RS232

Usage Rules:

a) If any parameters are missing or invalid values assigned in the string, the default value for the parameter(s) will be used.

b) All characters in the string are case insensitive (consistent with IEC 61131 spec).

c) The string is not order dependent (consistent with IEC 61131 spec).

d) Any extraneous information included in the string will be ignored.

APPENDIX D: PRODUCT SPECIFICATIONS

Hardware Specifications

Module Size: Single Wide I/O

Backplane Power Consumption: 5 watts @ 5 VDC

Operating Temperature: 0º to 60º C (32º to 185º F)

Storage Temperature: -40º to 85º C (-40º to 185º F)

Humidity: 0% to 95%, non-condensing

LIMITED PRODUCT WARRANTY

Warranty. Control Technology Inc. ("CTI") warrants that this CTI Industrial Product (the "Product") shall be free from defects in material and workmanship for a period of one (1) year from the date of purchase from CTI or from an authorized CTI Industrial Distributor, as the case may be. Repaired or replacement CTI products provided under this warranty are similarly warranted for a period of 6 months from the date of shipment to the customer or the remainder of the original warranty term, whichever is longer. This Product and any repaired or replacement products will be manufactured from new and/or serviceable used parts which are equal to new in the Product. This warranty is limited to the initial purchaser of the Product from CTI or from an authorized CTI Industrial Distributor and may not be transferred or assigned.

2. Remedies. Remedies under this warranty shall be limited, at CTI's option, to the replacement or repair of this Product, or the parts thereof, only after shipment by the customer at the customer's expense to a designated CTI service location along with proof of purchase date and an associated serial number. Repair parts and replacement products furnished under this warranty will be on an exchange basis and all exchanged parts or products become the property of CTI. Should any product or part returned to CTI hereunder be found by CTI to be without defect, CTI will return such product or part to the customer. The foregoing will be the exclusive remedies for any breach of warranty or breach of contract arising therefrom.

3. General. This warranty is only available if (a) the customer provides CTI with written notice of a warranty claim within the warranty period set forth above in Section 1 and (b) CTI's examination of the Product or the parts thereof discloses that any alleged defect has not been caused by a failure to provide a suitable environment as specified in the CTI Standard Environmental Specification and applicable Product specifications, or damage caused by accident, disaster, acts of God, neglect, abuse, misuse, transportation, alterations, attachments, accessories, supplies, non-CTI parts, non-CTI repairs or activities, or to any damage whose proximate cause was utilities or utility-like services, or faulty installation or maintenance done by someone other than CTI.

4. Product Improvement. CTI reserves the right to make changes to the Product in order to improve reliability, function or design in the pursuit of providing the best possible products.

5. Exclusive Warranty. THE WARRANTIES SET FORTH HEREIN ARE CUSTOMER'S EXCLUSIVE WARRANTIES. CTI HEREBY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED. WITHOUT LIMITING THE FOREGOING, CTI SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, COURSE OF DEALING, AND USAGE OF TRADE.

6. Disclaimer and Limitation of Liability. TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, CTI WILL NOT BE LIABLE FOR ANY BUSINESS INTERRUPTION OR LOSS OF PROFIT, REVENUE, MATERIALS, ANTICIPATED SAVINGS, DATA, CONTRACT, GOODWILL OR THE LIKE (WHETHER DIRECT OR INDIRECT IN NATURE) OR FOR ANY OTHER FORM OF INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND. CTI'S MAXIMUM CUMULATIVE LIABILITY RELATIVE TO ALL OTHER CLAIMS AND LIABILITIES, INCLUDING OBLIGATIONS UNDER ANY INDEMNITY, WHETHER OR NOT INSURED, WILL NOT EXCEED THE COST OF THE PRODUCT(S) GIVING RISE TO THE CLAIM OR LIABILITY. CTI DISCLAIMS ALL LIABILITY RELATIVE TO GRATUITOUS INFORMATION OR

ASSISTANCE PROVIDED BY, BUT NOT REQUIRED OF CTI HEREUNDER. ANY ACTION AGAINST CTI MUST BE BROUGHT WITHIN EIGHTEEN (18) MONTHS AFTER THE CAUSE OF ACTION ACCRUES. THESE DISCLAIMERS AND LIMITATIONS OF LIABILITY WILL APPLY REGARDLESS OF ANY OTHER CONTRARY PROVISION HEREOF AND REGARDLESS OF THE FORM OF ACTION, WHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY) OR OTHERWISE, AND FURTHER WILL EXTEND TO THE BENEFIT OF CTI'S VENDORS, APPOINTED DISTRIBUTORS AND OTHER AUTHORIZED RESELLERS AS THIRD-PARTY BENEFICIARIES. EACH PROVISION HEREOF WHICH PROVIDES FOR A LIMITATION OF LIABILITY, DISCLAIMER OF WARRANTY OR CONDITION OR EXCLUSION OF DAMAGES IS SEVERABLE AND INDEPENDENT OF ANY OTHER PROVISION AND IS TO BE ENFORCED AS SUCH.

7. Adequate Remedy. The customer is limited to the remedies specified herein and shall have no others for a nonconformity in the Product. The customer agrees that these remedies provide the customer with a minimum adequate remedy and are its exclusive remedies, whether based on contract, warranty, tort (including negligence), strict liability, indemnity, or any other legal theory, and whether arising out of warranties, representations, instructions, installations, or non-conformities from any cause. The customer further acknowledges that the purchase price of the Product reflects these warranty terms and remedies.

8. Force Majeure. CTI will not be liable for any loss, damage or delay arising out of its failure (or that of its subcontractors) to perform hereunder due to causes beyond its reasonable control, including without limitation, acts of God, acts or omissions of the customer, acts of civil or military authority, fires, strikes, floods, epidemics, quarantine restrictions, war, riots, acts of terrorism, delays in transportation, or transportation embargoes. In the event of such delay, CTI's performance date(s) will be extended for such length of time as may be reasonably necessary to compensate for the delay.

9. Governing Law. The laws of the State of Tennessee shall govern the validity, interpretation and enforcement of this warranty, without regard to its conflicts of law principles. The application of the United Nations Convention on Contracts for the International Sale of Goods shall be excluded.

REPAIR POLICY

In the event that the Product should fail during or after the warranty period, a Return Material Authorization (RMA) number can be requested orally or in writing from CTI main offices. Whether this equipment is in or out of warranty, a Purchase Order number provided to CTI when requesting the RMA number will aid in expediting the repair process. The RMA number that is issued and your Purchase Order number should be referenced on the returning equipment's shipping documentation. Additionally, if the product is under warranty, proof of purchase date and serial number must accompany the returned equipment. The current repair and/or exchange rates can be obtained by contacting CTI's main office at 1-800-537-8398 or go to www.controltechnology.com/support/repairs/.

When returning any module to CTI, follow proper static control precautions. Keep the module away from polyethylene products, polystyrene products and all other static producing materials. Packing the module in its original conductive bag is the preferred way to control static problems during shipment. Failure to observe static control precautions may void the warranty.