ETHERNET SE IMPLEMENTATION FOR

# FULLY INTEGRATED SERVO MOTORS

## CLASS 6 SMARTMOTOR™ WITH COMBITRONIC™ TECHNOLOGY


Rev. E, February 2026

DESCRIBES THE CLASS 6 D- AND M-STYLE SMARTMOTOR™ SUPPORT FOR THE ETHERNET SERIAL ENCAPSULATION PROTOCOL

www.animatics.com

MOOG ANIMATICS

# Copyright Notice



Moog Animatics Class 6 SmartMotor™ Ethernet Serial Encapsulation Guide, SC80100017-001, Rev. E.



Please let us know if you find any errors or omissions in this manual so that we can improve it for future readers. Such notifications should contain the words "Ethernet Serial Encapsulation Guide" in the subject line and be sent by e-mail to: animatics_sales@moog.com. Thank you in advance for your contribution.

Contact Us:

Moog Animatics
1995 NC Hwy 141
Murphy, NC 28906
USA

Website: www.animatics.com

Email: animatics_sales@moog.com

# Table Of Contents

# Introduction

This chapter provides information on the purpose and scope of this manual. It also provides information on safety notation, related documents and additional resources.

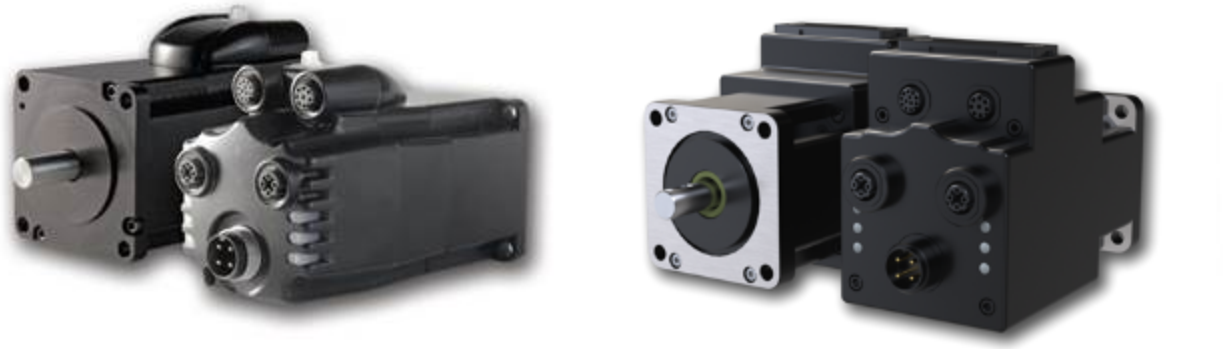
# Purpose

This guide describes the Moog Animatics Ethernet Serial Encapsulation protocol provided by the Class 6 EtherNet/IP (EIP) D-style and M-style SmartMotor™ integrated servos (see the next figures). It describes the major concepts that must be understood to integrate a SmartMotor follower[1] with an Ethernet Serial Encapsulation controller[1] device. It also includes a byte-by-byte discussion of the Moog Animatics Ethernet Serial Encapsulation protocol.

NOTE: The feature set described in this version of the manual refers to SmartMotor (-EIP option) firmware version: M-style 6.0.2.28 or later, D-style 6.4.2.54 or later, and netX (NXF) firmware version 3.3.0.3 or later.

NOTE: The "keepalive" feature, which resets broken connections, requires firmware version: M-style 6.0.2.41 or later, D-style 6.4.2.54 or later, and netX firmware (NXF) version 3.4.0.5 or later. Earlier firmware versions will not activate this feature.

The Ethernet Serial Encapsulation protocol was developed by Moog Animatics as a means for the Class 6 EtherNet/IP (EIP) motor to receive and send serial communications over Ethernet. In addition to this protocol, Moog Animatics offers a variety of other fieldbus protocol options for the Class 6 M-style and D-style motors, such as EtherNet/IP, Modbus, and more. Please contact Moog Animatics for details.



*Class 6 M-Style SmartMotor: MT (Left) vs. MT2 (Right)*

*Class 6 D-Style SmartMotor*

[1]Moog Animatics has replaced the terms "master" and "slave" with "controller" and "follower", respectively.

# Safety Information

This section describes the safety symbols and other safety information.

## Safety Symbols

The manual may use one or more of these safety symbols:

WARNING: This symbol indicates a potentially nonlethal mechanical hazard, where failure to comply with the instructions could result in serious injury to the operator or major damage to the equipment.

CAUTION: This symbol indicates a potentially minor hazard, where failure to comply with the instructions could result in slight injury to the operator or minor damage to the equipment.

NOTE: Notes are used to emphasize non-safety concepts or related information.

## Other Safety Considerations

The Moog Animatics SmartMotors are supplied as components that are intended for use in an automated machine or system. As such, it is beyond the scope of this manual to attempt to cover all the safety standards and considerations that are part of the overall machine/system design and manufacturing safety. Therefore, this information is intended to be used only as a general guideline for the machine/system designer.

It is the responsibility of the machine/system designer to perform a thorough "Risk Assessment" and to ensure that the machine/system and its safeguards comply with the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the site where the machine is being installed and operated. For more details, see Machine Safety on page 8.

### Motor Sizing

It is the responsibility of the machine/system designer to select SmartMotors that are properly sized for the specific application. Undersized motors may: perform poorly, cause excessive downtime or cause unsafe operating conditions by not being able to handle the loads placed on them. The *System Best Practices* document, which is available on the Moog Animatics website, contains information and equations that can be used for selecting the appropriate motor for the application.

Replacement motors must have the same specifications and firmware version used in the approved and validated system. Specification changes or firmware upgrades require the approval of the system designer and may require another Risk Assessment.

### Environmental Considerations

It is the responsibility of the machine/system designer to evaluate the intended operating environment for dust, high-humidity or presence of water (for example, a food-processing environment that requires water or steam wash down of equipment), corrosives or chemicals that may come in contact with the machine, etc. Moog Animatics manufactures specialized IP-rated motors for operating in extreme conditions. For details, see the *Moog Animatics Product Catalog*.

## Machine Safety

In order to protect personnel from any safety hazards in the machine or system, the machine/system builder must perform a "Risk Assessment", which is often based on the ISO 13849 standard. The design/implementation of barriers, emergency stop (E-stop) mechanisms and other safeguards will be driven by the Risk Assessment and the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the site where the machine is being installed and operated. The methodology and details of such an assessment are beyond the scope of this manual. However, there are various sources of Risk Assessment information available in print and on the internet.

NOTE: The next list is an example of items that would be evaluated when performing the Risk Assessment. Additional items may be required. The safeguards must ensure the safety of all personnel who may come in contact with or be in the vicinity of the machine.

In general, the machine/system safeguards must:

- Provide a barrier to prevent unauthorized entry or access to the machine or system. The barrier must be designed so that personnel cannot reach into any identified danger zones.
- Position the control panel so that it is outside the barrier area but located for an unrestricted view of the moving mechanism. The control panel must include an E-stop mechanism. Buttons that start the machine must be protected from accidental activation.
- Provide E-stop mechanisms located at the control panel and at other points around the perimeter of the barrier that will stop all machine movement when tripped.
- Provide appropriate sensors and interlocks on gates or other points of entry into the protected zone that will stop all machine movement when tripped.
- Ensure that if a portable control/programming device is supplied (for example, a hand-held operator/programmer pendant), the device is equipped with an E-stop mechanism.

  NOTE: A portable operation/programming device requires *many* additional system design considerations and safeguards beyond those listed in this section. For details, see the safety standards specified by the governing authority (for example, ISO, OSHA, UL, etc.) for the site where the machine is being installed and operated.
- Prevent contact with moving mechanisms (for example, arms, gears, belts, pulleys, tooling, etc.).
- Prevent contact with a part that is thrown from the machine tooling or other part-handling equipment.
- Prevent contact with any electrical, hydraulic, pneumatic, thermal, chemical or other hazards that may be present at the machine.
- Prevent unauthorized access to wiring and power-supply cabinets, electrical boxes, etc.
- Provide a proper control system, program logic and error checking to ensure the safety of all personnel and equipment (for example, to prevent a run-away condition). The control system must be designed so that it does not automatically restart the machine/system after a power failure.
- Prevent unauthorized access or changes to the control system or software.

## Documentation and Training

It is the responsibility of the machine/system designer to provide documentation on safety, operation, maintenance and programming, along with training for all machine operators, maintenance technicians, programmers, and other personnel who may have access to the machine. This documentation must include proper lockout/tagout procedures for maintenance and programming operations.

It is the responsibility of the operating company to ensure that:

- All operators, maintenance technicians, programmers and other personnel are tested and qualified before acquiring access to the machine or system.
- The above personnel perform their assigned functions in a responsible and safe manner to comply with the procedures in the supplied documentation and the company safety practices.
- The equipment is maintained as described in the documentation and training supplied by the machine/system designer.

### Additional Equipment and Considerations

The Risk Assessment and the operating company's standard safety policies will dictate the need for additional equipment. In general, it is the responsibility of the operating company to ensure that:

- Unauthorized access to the machine is prevented at all times.
- The personnel are supplied with the proper equipment for the environment and their job functions, which may include: safety glasses, hearing protection, safety footwear, smocks or aprons, gloves, hard hats and other protective gear.
- The work area is equipped with proper safety equipment such as first aid equipment, fire suppression equipment, emergency eye wash and full-body wash stations, etc.
- There are no modifications made to the machine or system without proper engineering evaluation for design, safety, reliability, etc., and a Risk Assessment.

## Safety Information Resources

Additional SmartMotor safety information can be found on the Moog Animatics website; open the topic "Controls - Notes and Cautions" located at:

https://www.animatics.com/support/downloads/knowledgebase/controls---notes-and-cautions.html

OSHA standards information can be found at:

https://www.osha.gov/law-regs.html

ANSI-RIA robotic safety information can be found at:

http://www.robotics.org/robotic-content.cfm/Robotics/Safety-Compliance/id/23

UL standards information can be found at:

http://ulstandards.ul.com/standards-catalog/

ISO standards information can be found at:

http://www.iso.org/iso/home/standards.htm

EU standards information can be found at:

http://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/index_en.htm

# Additional Documents

The Moog Animatics website contains additional documents that are related to the information in this manual. Please refer to these lists.

## Related Guides

- *Class 6 D-Style SmartMotor™ Installation and Startup Guide*

  https://www.animatics.com/cl-6-d-style-install-startup-guide

- *Class 6 M-Style SmartMotor™ Installation and Startup Guide*

  https://www.animatics.com/cl-6-install-startup-guide

- *SmartMotor™ Developer's Guide*

  https://www.animatics.com/smartmotor-developers-guide

- *SmartMotor™ Homing Procedures and Methods Application Note*

  https://www.animatics.com/homing-application-note

- *SmartMotor™ System Best Practices Application Note*

  https://www.animatics.com/system-best-practices-application-note

In addition to the documents listed above, guides for fieldbus protocols and more can be found on the website: https://www.animatics.com/support/downloads.manuals.html

## Other Documents

- SmartMotor™ Certifications

  https://www.animatics.com/certifications.html

- *SmartMotor Developer's Worksheet*
  (interactive tools to assist developer: Scale Factor Calculator, Status Words, CAN Port Status, Serial Port Status, RMODE Decoder and Syntax Error Codes)

  https://www.animatics.com/support/downloads.knowledgebase.html

- *Moog Animatics Product Catalog*

  https://www.animatics.com/support/moog-animatics-catalog.html

# Additional Resources

The Moog Animatics website contains useful resources such as product information, documentation, product support and more. Please refer to these addresses:

- General company information:

  https://www.animatics.com

- Product information:

  https://www.animatics.com/products.html

- Product support (Downloads, How-to Videos, Forums and more):

  https://www.animatics.com/support.html

- Contact information, distributor locator tool, inquiries:

  https://www.animatics.com/contact-us.html

- Applications (Application Notes and Case Studies):

  https://www.animatics.com/applications.html

# System Connections and Status LEDs

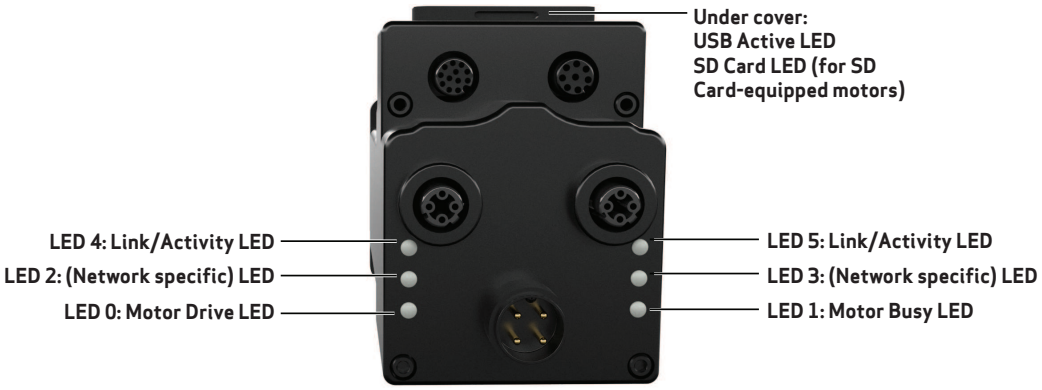
The chapter describes the system connections and the status LEDs.

NOTE: For information on your motor's connector pinouts and cables, refer to the corresponding SmartMotor Installation and Startup Guide.

NOTE: If you have set your PC's network adapter to a fixed IP address for temporary connections to SmartMotors with SMI, remember to return it to DHCP when done to avoid local area network connectivity issues.

# Understanding the Status LEDs

The next figure and tables describe the functionality of the Ethernet Serial Encapsulation Status LEDs on the Class 6 M-style EIP SmartMotor.




SD Card LED (for SD Card-equipped motors)

| | |
|---|---|
| Off | No card, bad or damaged card |
| Blinking green | Busy, do not remove card |
| Solid green | Card detected |
| Solid red | Card with no SmartMotor data |

See the topic "Understanding the SD Card" for details. It is in the *Class 6 M-Style Installation and Startup Guide*.

USB Active LED

| | |
|---|---|
| Flashing green | Active |
| Flashing red | Suspended |
| Solid red | USB power detected, no configuration |

If the USB port is plugged in at power up, it flashes for ~4 seconds, turns solid red until it is detected through SMI, then it returns to flashing

LED 0: Motor Drive LED

| | |
|---|---|
| Off | No power |
| Solid green | Drive on |
| Blinking green | Drive off, no faults |
| Triple red flash | Watchdog fault |
| Solid red | Faulted or no drive enable input |
| Alt. red/green | In boot load; needs firmware |

**LED 0 and 1 Status on Power-up:**

- With no program and the travel limit inputs are low:
  LED 0 solid red; motor in fault state due to travel limit fault
  LED 1 off
- With no program and the travel limits are high:
  LED 0 solid red for 500 milliseconds then flashing green
  LED 1 off
- With a program that only disables travel limits:
  LED 0 red for 500 milliseconds then flashing green
  LED 1 off

LED 1: Motor Busy LED

| | |
|---|---|
| Off | Not busy |
| Solid green | Drive on, trajectory in progress |
| Flashing # red | Flashes fault code (see below) when Drive LED is solid red |

**Fault Codes:** pauses for 2 sec before flashing the code

| Flash | Description |
|---|---|
| 1 | NOT Used |
| 2 | Bus Voltage |
| 3 | Over Current |
| 4 | Excessive Temperature |
| 5 | Excessive Position |
| 6 | Velocity Limit |
| 7 | dE/Dt - First derivative of position error is excessive |
| 8 | Hardware Positive Limit Reached |
| 9 | Hardware Negative Limit Reached |
| 10 | Software Positive Travel Limit Reached |
| 11 | Software Negative Travel Limit Reached |

LED 2: EtherNet/IP Network Status LED

| | |
|---|---|
| Off | No power or no IP address |
| Flashing red/grn | Power-up self test |
| Flashing green | No connections |
| Solid green | Connected |
| Flashing red | Connection timeout |
| Solid red | Duplicate IP |

LED 4: Link/Activity LED

| | |
|---|---|
| Off | No/bad cable; no/bad Link port |
| Solid green | Link established |
| Blinking green | Activity |

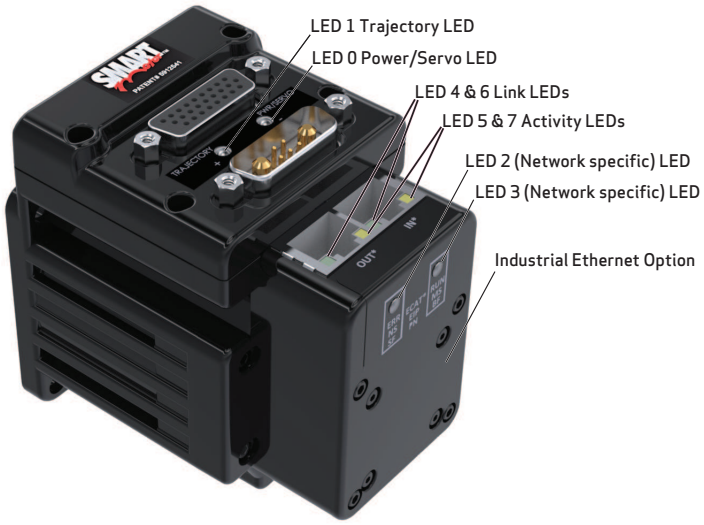
LED 3: EtherNet/IP Module Status LED

| | |
|---|---|
| Off | No power |
| Flashing red/grn | Power-up self test |
| Flashing green | Standby |
| Solid green | Device operational |
| Flashing red | Minor fault |
| Solid red | Major fault |

LED 5: Link/Activity LED

| | |
|---|---|
| Off | No/bad cable; no/bad Link port |
| Solid green | Link established |
| Blinking green | Activity |

The next figure and tables describe the functionality of the Ethernet Serial Encapsulation Status LEDs on the Class 6 D-style EIP SmartMotor.




LED 0: Power/Servo LED

| | |
|---|---|
| Off | No power |
| Solid green | Drive on |
| Blinking green | Drive off, no faults |
| Triple red flash | Watchdog fault |
| Solid red | Faulted or no drive enable input |
| Alt. red/green | In boot load; needs firmware |

**LED 0 and 1 Status on Power-up:**

- With no program and the travel limit inputs are low:
  LED 0 solid red; motor in fault state due to travel limit fault
  LED 1 off
- With no program and the travel limits are high:
  LED 0 solid red for 500 milliseconds then flashing green
  LED 1 off
- With a program that only disables travel limits:
  LED 0 red for 500 milliseconds then flashing green
  LED 1 off

LED 1: Trajectory LED

| | |
|---|---|
| Off | Not busy |
| Solid green | Drive on, trajectory in progress |
| Flashing # red | Flashes fault code (see below) when Power/Servo LED is solid red |

**Fault Codes:** pauses for 2 sec before flashing the code

| Flash | Description |
|---|---|
| 1 | NOT Used |
| 2 | Bus Voltage |
| 3 | Over Current |
| 4 | Excessive Temperature |
| 5 | Excessive Position |
| 6 | Velocity Limit |
| 7 | dE/Dt - First derivative of position error is excessive |
| 8 | Hardware Positive Limit Reached |
| 9 | Hardware Negative Limit Reached |
| 10 | Software Positive Travel Limit Reached |
| 11 | Software Negative Travel Limit Reached |

Industrial Ethernet Option

LED 2: EtherNet/IP Network Status LED

| | |
|---|---|
| Off | No power or no IP address |
| Flashing red/grn | Power-up self test |
| Flashing green | No connections |
| Solid green | Connected |
| Flashing red | Connection timeout |
| Solid red | Duplicate IP |

LED 4 & 6 Link LEDs

| | |
|---|---|
| Off | No/bad cable; no/bad Link port |
| Solid green | Link established |

Industrial Ethernet Option

LED 3: EtherNet/IP Module Status LED

| | |
|---|---|
| Off | No power |
| Flashing red/grn | Power-up self test |
| Flashing green | Standby |
| Solid green | Device operational |
| Flashing red | Minor fault |
| Solid red | Major fault |

LED 5 & 7 Activity LEDs

| | |
|---|---|
| Off | No activity |
| Blinking amber | Activity |

# Using Ethernet Serial Encapsulation

This chapter describes how to enable Ethernet Serial Encapsulation communications with your SmartMotor, along with information on supported function codes, input registers and holding registers.

# Ethernet Serial Encapsulation Description

Ethernet Serial Encapsulation is a protocol developed by Moog Animatics that allows host software, such as SMI, to communicate via serial commands over Ethernet. The Moog Animatics Class 6 EtherNet/IP (EIP) SmartMotor supports communication with a PLC, HMI, or other host device over Ethernet.

Unlike Modbus RTU serial communication, the OCHN command is not needed or used for Ethernet Serial Encapsulation communication. In fact, once the motors are connected to the Ethernet network, they will be able to communicate with the Ethernet Serial Encapsulation controller if DHCP is used, or they will simply need a static IP address if DHCP is not being used.

## TCP Port

There is a single instance of the TCP port that acts like a serial command parser. Note that a second concurrent attempt to connect will be rejected. However, if the first connection is closed by the client, then the motor will accept another connection.

CAUTION: There is no security check or other method available to disable the TCP port. Therefore, the network is assumed to be "friendly" (i.e., secure, ready and safe to connect to).

### TCP Port 10001

The TCP port uses ASCII commands to create a "pipeline" to the same handling as a serial port by:

- Prepending with a byte of the value: 0x80
- Terminating with a byte of the value: 0x20 (a space)
- Terminating responses with 0x0d

Example:

- Command: 0x80RPA0x20 = "0x80RPA " (note there is a space at the end).
- Response: 102330x0d = the number 10233 with the byte value 0x0d at the end.

## UDP Port

Optionally, and unrelated to the TCP connection, a broadcast to port 30718 over UDP can be used to discover the SmartMotor(s).

### UDP port 30718

***Send a UDP packet broadcast:***

To address 255.255.255.255, port 30718 with the BCAST flag:

1. Send the packet from your port 30718
2. Send this data:
    - Datagram length: four bytes
    - Content (from first to last): 0x00, 0x00, 0x00, 0xf6

***Listen for response back to your port 30718:***

1. Check that data length received is 30 bytes.
2. Check that the first four bytes (0-3) are:

   0x00, 0x00, 0x00, 0xf7
3. For the remaining bytes:
   - Bytes 4-23 should all be 0
   - Bytes 24-29 represent the MAC ID of the motor

## Detecting the Motors in SMI

When using the Detect Motors feature in SMI to detect the SmartMotor(s), they will be detected as "Ethernet". Note that:

- The ability to change the SmartMotor IP address through the SMI menus and dialogs will not function. However, you can use SMI software's Terminal window to issue commands that set the IP address. See the SMI software's online help for details on the Terminal window.
- The SmartMotor "webpage" feature, also used to set the IP address, will not function.

Therefore, the IP address must be set through either:

- The IP Control command (IPCTL), or
- The SmartMotor's USB port

For details, refer to the next section.

## Setting the IP Address

As mentioned previously, for Ethernet Serial Encapsulation on the SmartMotor, the IP address can be either static or dynamic (DHCP). The default operation is dynamic addressing. For applications requiring a fixed IP address, it must be set using the IP control command (IPCTL) through either:

- The Ethernet port, or
- The SmartMotor's USB port

The IPCTL command allows you to change the IP address of the SmartMotor. The default setting is "0.0.0.0" for IP address, subnet mask, and gateway disabled/automatic. Three function codes (0, 1, and 2) are available for setting a specific IP address, a specific subnet mask, and/or a specific gateway address, respectively. It uses the form:

IPCTL(function,"string")

- function is one of these codes:

| function | Description |
|---|---|
| 0 | Set IP address |
| 1 | Set subnet mask |
| 2 | Set gateway |

- "string" is formatted as an IP address and entered as a string

For example:

```
IPCTL(0,"192.168.0.10") 'Set the IP address to 192.168.0.10
```

For more details on the IPCTL command, see the *SmartMotor™ Developer's Guide*.. For details on the Ethernet and USB ports, see the Installation and Startup Guide for your Class 6 SmartMotor.

# Supported/Not-Supported Functionality

## Supported Functionality

A small set of functions are supported for access to these basic SmartMotor operations:

- Report commands: RPA, RSP, etc.

  Report commands are shown in the *SmartMotor™ Developer's Guide*. They can be identified by looking at the command lists (near the end of the guide) for commands that begin with a superscript "R" character, for example: $^{R}$PA, $^{R}$SP, $^{R}$PC etc.

  Commands that begin with an "R" character that *is not* superscript *are not* report commands, for example: RANDOM=, RESUME, RETURN, RETURNI, RUN, RUN?.

  NOTE: Also, refer to the Not-Supported Functionality section.

  For more details, refer to Example with a String Response on page 23 and Example with a Numeric Response on page 24.

- Assign a variable: a=400

  For more details, refer to Example of Assigning a Variable on page 25.

- Motion commands

  For the Class 6 M-styleSmartMotor, all motion commands listed in the Motion Control section of the *SmartMotor™ Developer's Guide* will work with the exception of these commands (these are supported on the Class 6 D-style SmartMotor):

  - ADTS
  - ATS
  - DTS
  - VTS
  - PTS
  - PRTS
  - GS
  - PRTSS
  - PTSS
  - RPTSD
  - RPTST
  - TSWAIT

- User program download, run and upload.

  For more details, refer to Example of Program Downloading, Running and Uploading on page 26.

- Channel 2 is always open (see Not-Supported Functionality)

- For Class 6 M-style, the TCP keepalive feature requires firmware 6.0.2.41 or higher with netX firmware (NXF) version 3.4.0.5 or higher; for Class 6 D-style, it requires firmware 6.4.2.54 or higher with netX firmware (NXF) version 3.4.0.5 or higher.)

## Not-Supported Functionality

The following SmartMotor functions are not supported:

- Serial "data" mode is not supported (for example, RGETCHR)

  Report commands specific to the serial port, like RGETCHR, RGETCHR1, etc. (reporting the state of the actual serial ports) will report a value without problem. However, commands like RGETCHR2, RLEN2 are not supported specific to the state of the serial encapsulation channel itself; i.e., you can’t put the serial encapsulation into "data" mode.

  Any command that can be performed on the serial port should also work through Ethernet Serial Encapsulation. There are a few operations, however, that don't apply to this environment:

  - Setting or reading the baud rate specific to that port
  - OCHN, serial interpolation time sync
  - Serial addressing

- Open channel and change channel (OCHN and CCHN, respectively) commands are not supported

  NOTE: Channel 2 is always open.

# Ethernet Serial Encapsulation Communications Setup

This topic contains Ethernet Serial Encapsulation communications setup information.

For a typical Ethernet Serial Encapsulation application:

- Ethernet Serial Encapsulation requires a Class 6 M-style or D-style SmartMotor with the "-EIP" option. Verify that you have the correct motor.
- Verify the type of motor addressing being used. Note that:
    - For dynamic IP (DHCP) addressing (SmartMotor default), there is no need to set an IP address on the motor.
    - For static IP addressing, you will need to set a static IP address on the motor. For more details, see Setting the IP Address on page 17.
- There is no need to open the Ethernet Serial Encapsulation port, it is already open by default (using TCP port 10001). Therefore, no special program is needed.
- There is no need for a node ID—the IP address serves as the motor's identification. Note that the Node ID is typically assumed to be "0" in Ethernet Serial Encapsulation.

| Command | Purpose | Value | Non-Volatile Setting |
|---|---|---|---|
| ETHCTL(100, value) | Enable/disable ports | -1 default<br>0 disable TCP communications port and UDP discovery port.<br>1 enable TCP communications port only<br>2 enable UDP discovery port only<br>3 enable TCP communications port and UDP discovery port. (default) | Yes |
| ETHCTL(110, value) | Keepalive time for Ethernet serial encapsulated connection. | -1 default (3 seconds)<br>0 disable<br>1-127 keepalive time (seconds) | Yes |

# Ethernet Serial Encapsulation Sample Command Sequences

This topic contains some sample Ethernet Serial Encapsulation command sequences. These examples show the data sent from and received by SMI software communicating with a SmartMotor. For these examples, an open-source software (Wireshark network protocol analyzer) is used to show the communications between SMI software and the SmartMotor.

NOTE: There are various utilities available for this purpose. Therefore, Moog Animatics does not endorse any particular one—the selection depends on the requirements of your application.

For each of the sections:

- Section title = action being performed
- Output = formatted byte stream sent from controller to the SmartMotor
- Input = formatted byte stream received by the controller from the SmartMotor

For the tables:

NOTE: A table is provided to illustrate the parts of the byte sequence only. The byte sequence must be transmitted as a stream of bytes shown in the Output/Input strings above the table (i.e., no pause or null for the blank cells).

These items unique to the UDP Discovery Example:

- Operation Code (Bytes 0-3) = specifies the operation being performed
- Reserved (Bytes 4-23) = reserved bytes
- MAC address (Bytes 24-29) = the MAC address of the responding motor

These items are common to TCP Command Examples:

- Prefix = required stream prefix (Output always 80; Input = N/A)
- Command = Output byte string representing the desired command
- Response = Input byte string representing the response from the motor
- Terminator = required stream terminator (Output always 20; Input = 0d)

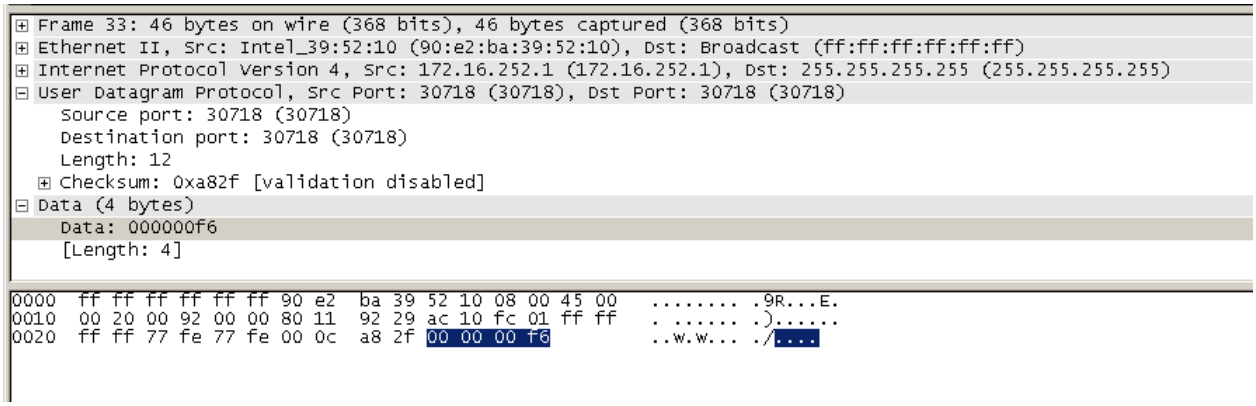
# UDP (User Datagram Protocol) Discovery Example

Output: 00 00 00 f6

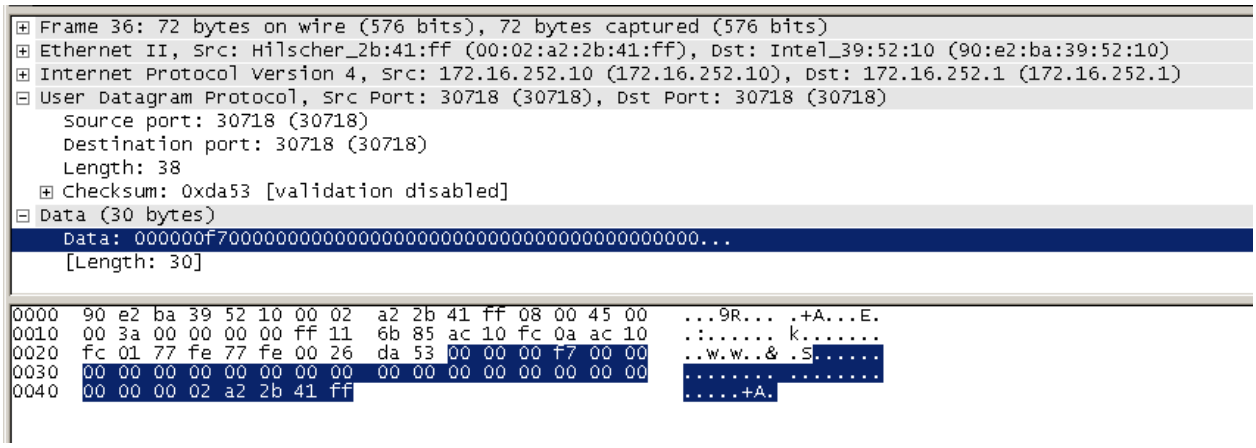
Input: 00 00 00 f7 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 a2 2b 41 ff

| | UDP Packet Data Payload | | |
|---|---|---|---|
| | **Operation Code (Bytes 0-3)** | **Reserved (Bytes 4-23)** | **MAC address (Bytes 24-29)** |
| Output | 00 00 00 f6 | | |
| Input | 00 00 00 f7 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 02 a2 2b 41 ff |
| A table is provided to illustrate the parts of the byte sequence only. The byte sequence must be transmitted as a stream of bytes shown in the Output/Input strings above the table (i.e., no pause or null for the blank cells). | | | |

***Output:***




***Input:***



# TCP (Transmission Control Protocol) Command Examples

## Example with a String Response

This is a TCP example with a string response; the RSP command replies with "06250/6.0.2.30"

Output: 80 52 53 50 20

Input: 30 36 32 35 30 2f 36 2e 30 2e 32 2e 33 30 0d

| | TCP Stream | | | |
|---|---|---|---|---|
| | **Prefix** | **Command** | **Response** | **Terminator** |
| Output | 80 | 52 53 50 ("RSP") | | 20 |
| Input | | | 30 36 32 35 30 2f 36 2e 30 2e 32 2e 33 30 ("06250/6.0.2.30") | 0d |
| A table is provided to illustrate the parts of the byte sequence only. The byte sequence must be transmitted as a stream of bytes shown in the Output/Input strings above the table (i.e., no pause or null for the blank cells). | | | | |

***Output:***

```
⊞ Frame 44: 59 bytes on wire (472 bits), 59 bytes captured (472 bits)
⊞ Ethernet II, Src: Intel_39:52:10 (90:e2:ba:39:52:10), Dst: Hilscher_2b:41:ff (00:02:a2:2b:41:ff)
⊞ Internet Protocol Version 4, Src: 172.16.252.1 (172.16.252.1), Dst: 172.16.252.10 (172.16.252.10)
⊟ Transmission Control Protocol, Src Port: 58203 (58203), Dst Port: scp-config (10001), Seq: 1, Ack: 1, Len: 5
    Source port: 58203 (58203)
    Destination port: scp-config (10001)
    [Stream index: 3]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 6    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x504d [validation disabled]
  ⊞ [SEQ/ACK analysis]
⊟ Data (5 bytes)
    Data: 8052535020
    [Length: 5]
```

```
0000  00 02 a2 2b 41 ff 90 e2  ba 39 52 10 08 00 45 00   ...+A... .9R...E.
0010  00 2d 00 99 40 00 80 06  00 00 ac 10 fc 01 ac 10   .-..@... ........
0020  fc 0a e3 5b 27 11 58 c6  fc 76 00 00 19 6f 50 18   ...['.X. .v...oP.
0030  fa f0 50 4d 00 00 80 52  53 50 20                  ..PM...R SP
```

***Input:***

```
⊞ Frame 45: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)
⊞ Ethernet II, Src: Hilscher_2b:41:ff (00:02:a2:2b:41:ff), Dst: Intel_39:52:10 (90:e2:ba:39:52:10)
⊞ Internet Protocol Version 4, Src: 172.16.252.10 (172.16.252.10), Dst: 172.16.252.1 (172.16.252.1)
⊟ Transmission Control Protocol, Src Port: scp-config (10001), Dst Port: 58203 (58203), Seq: 1, Ack: 6, Len: 15
    Source port: scp-config (10001)
    Destination port: 58203 (58203)
    [Stream index: 3]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 16    (relative sequence number)]
    Acknowledgement number: 6    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
    Window size value: 2915
    [Calculated window size: 2915]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x6fb9 [validation disabled]
  ⊞ [SEQ/ACK analysis]
⊟ Data (15 bytes)
    Data: 30363235302f362e302e322e33300d
    [Length: 15]
```

```
0000  90 e2 ba 39 52 10 00 02  a2 2b 41 ff 08 00 45 00   ...9R... .+A...E.
0010  00 37 00 02 00 00 ff 06  6b 91 ac 10 fc 0a ac 10   .7...... k.......
0020  fc 01 27 11 e3 5b 00 00  19 6f 58 c6 fc 7b 50 18   ..'..[.. .oX..{P.
0030  0b 63 6f b9 00 00 30 36  32 35 30 2f 36 2e 30 2e   .co...06 250/6.0.
0040  32 2e 33 30 0d                                    2.30.
```

## Example with a Numeric Response

This is a TCP example with a numeric response as a string; the RPA command reports the value 1105.

Output: 80 52 50 41 20
Input: 31 31 30 35 0d

| | TCP Stream* | | | |
|---|---|---|---|---|
| | **Prefix** | **Command** | **Response** | **Terminator** |
| Output | 80 | 52 50 41 ("RPA") | | 20 |
| Input | | | 31 31 30 35 ("1105") | 0d |
| A table is provided to illustrate the parts of the byte sequence only. The byte sequence must be transmitted as a stream of bytes shown in the Output/Input strings above the table (i.e., no pause or null for the blank cells). | | | | |

***Output:***

```
⊞ Frame 706: 59 bytes on wire (472 bits), 59 bytes captured (472 bits)
⊞ Ethernet II, Src: Intel_39:52:10 (90:e2:ba:39:52:10), Dst: Hilscher_2b:41:ff (00:02:a2:2b:41:ff)
⊞ Internet Protocol Version 4, Src: 172.16.252.1 (172.16.252.1), Dst: 172.16.252.10 (172.16.252.10)
⊟ Transmission Control Protocol, Src Port: 51490 (51490), Dst Port: scp-config (10001), Seq: 1, Ack: 1, Len: 5
    Source port: 51490 (51490)
    Destination port: scp-config (10001)
    [Stream index: 11]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 6    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0x504d [validation disabled]
  ⊞ [SEQ/ACK analysis]
⊟ Data (5 bytes)
    Data: 8052504120
    [Length: 5]
```

```
0000  00 02 a2 2b 41 ff 90 e2  ba 39 52 10 08 00 45 00   ...+A... .9R...E.
0010  00 2d 02 b2 40 00 80 06  00 00 ac 10 fc 01 ac 10   .-..@... ........
0020  fc 0a c9 22 27 11 a2 01  3e 19 00 00 19 85 50 18   ...''... >.....P.
0030  fa f0 50 4d 00 00 80 52  50 41 20                  ..PM...R PA
```

***Input:***

```
⊞ Frame 707: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
⊞ Ethernet II, Src: Hilscher_2b:41:ff (00:02:a2:2b:41:ff), Dst: Intel_39:52:10 (90:e2:ba:39:52:10)
⊞ Internet Protocol Version 4, Src: 172.16.252.10 (172.16.252.10), Dst: 172.16.252.1 (172.16.252.1)
⊟ Transmission Control Protocol, Src Port: scp-config (10001), Dst Port: 51490 (51490), Seq: 1, Ack: 6, Len: 5
    Source port: scp-config (10001)
    Destination port: 51490 (51490)
    [Stream index: 11]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 6    (relative sequence number)]
    Acknowledgement number: 6    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
    Window size value: 2915
    [Calculated window size: 2915]
    [Window size scaling factor: -2 (no window scaling used)]
  ⊞ Checksum: 0xfbf7 [validation disabled]
  ⊞ [SEQ/ACK analysis]
⊟ Data (5 bytes)
    Data: 313130350d
    [Length: 5]
```

```
0000  90 e2 ba 39 52 10 00 02  a2 2b 41 ff 08 00 45 00   ...9R... .+A...E.
0010  00 2d 00 94 00 00 ff 06  6b 09 ac 10 fc 0a ac 10   .-...... k.......
0020  fc 01 27 11 c9 22 00 00  19 85 a2 01 3e 1e 50 18   ..'..".. ....>.P.
0030  0b 63 fb f7 00 00 31 31  30 35 0d 00               .c....11 05..
```

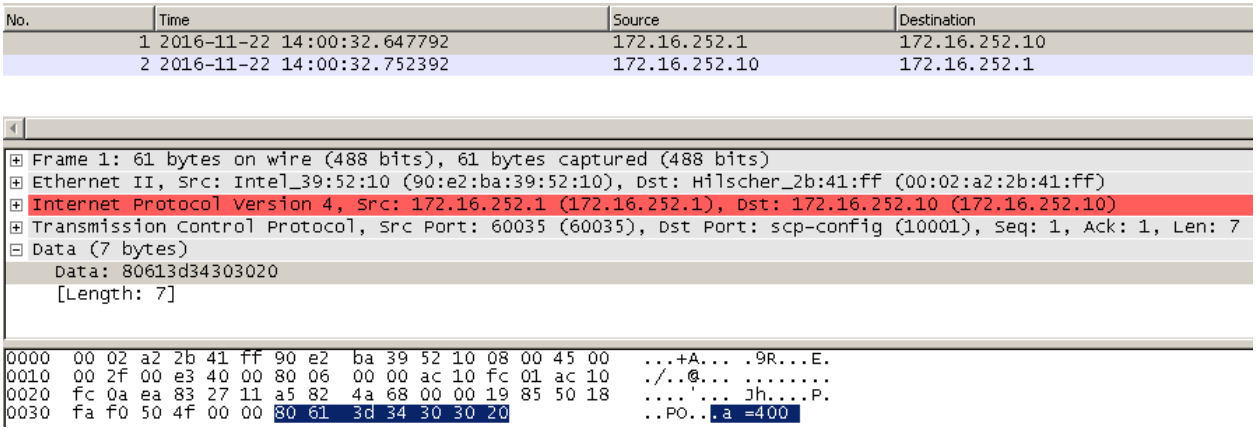
## Example of Assigning a Variable

This is a TCP example of assigning a numeric value to a variable: a=400.

Output: 80 61 3D 34 30 30 20
Input: 31 31 30 35 0d

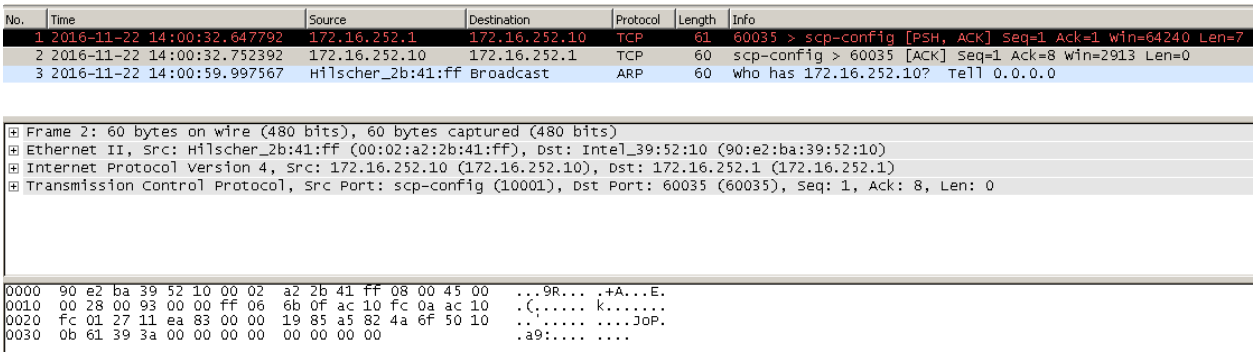
| | TCP Stream* | | | |
|---|---|---|---|---|
| | **Prefix** | **Command** | **Response** | **Terminator** |
| Output | 80 | 61 3D 34 30 30 ("a=400") | | 20 |
| Input | N/A - See Input figure below | | | |
| A table is provided to illustrate the parts of the byte sequence only. The byte sequence must be transmitted as a stream of bytes shown in the Output/Input strings above the table (i.e., no pause or null for the blank cells). | | | | |

### *Output:*




### *Input:*

The next figure shows the SmartMotor ACKnowledgement back to the PC/controller. However, it doesn't contain any data to respond to; it is just to keep the TCP connection alive.



## Example of Program Downloading, Running and Uploading

This is a TCP example of downloading, running and uploading a program:

```
' My test program

WHILE 1
     PRINT7("Hi there Ethernet host",#13)
     WAIT=1000
LOOP

END
```

### *Downloading the Program*

| | | |
|---|---|---|
| Output: | 80 4C 4F 41 44 20 | LOAD command sent to motor |
| Input: | 06 | ACK from motor |
| Output: | 24 36 30 30 30 30 30 31 31 36 44 30 30 30 30 34<br>30 30 30 30 30 30 30 30 30 30 30 24 57 30 30 33 | 32 character block of program download |
| Input: | 06 | ACK from motor |
| Output: | 43 4C 45 20 31 0A 50 52 49 4E 54 32 28 22 48 69<br>20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 | 32 character block of program download |
| **Input:** | 06 | ACK from motor |
| **Output:** | 68 6F 73 74 22 2C 23 31 33 29 0A 57 41 49 54 3D<br>31 30 30 30 0A 24 4C 30 30 30 30 50 0A 45 4E 44 | 32 character block of program download |
| **Input:** | 06 | ACK from motor |
| **Output:** | 0A FF 24 74 65 73 74 20 65 74 68 65 72 6E 65 74<br>2E 73 6D 78 00 20 20 20 20 20 20 10 0B 16 13 07 | 32 character block of program download |
| **Input:** | 06 | ACK from motor |
| Output: | 02 87 00 00 00 59 08 | Remaining character block of program download |
| Output: | FF FF 20 | Output, but no reply expected |
| Output: | 80 52 43 4B 53 20 | RCKS command |
| Input: | 30 30 30 30 30 30 20 30 30 31 31 36 44 20 50 0D | Two checksum values and P (pass) |

*SMI Serial Data Analyzer (shows Output and Input)*

```
>80 4C 4F 41 44 20                                              | ..LOAD..
>06                                                             | ..
>24 36 30 30 30 30 30 31 31 36 44 30 30 30 30 34 30 30 30 30    | $600000116D000040000
 30 30 30 30 30 30 30 24 57 30 30 33 06 43 4C 45 20 31 0A 50    | 0000000$W003..CLE..1..P
 52 49 4E 54 32 28 22 48 69 20 74 68 65 72 65 20 45 74 68 65    | RINT2("Hi..there..Ethe
 72 6E 65 74 20 06 68 6F 73 74 22 2C 23 31 33 29 0A 57 41 49    | rnet....host",#13)..WAI
 54 3D 31 30 30 30 0A 24 4C 30 30 30 30 50 0A 45 4E 44 06 0A    | T=1000..$L0000P..END....
 FF 24 74 65 73 74 20 65 74 68 65 72 6E 65 74 2E 73 6D 78 00    | ..$test..ethernet.smx..
 20 20 20 20 20 20 10 0B 16 13 07 06 02 87 00 00 00 59 08       | ...............................Y..
>FF FF 20                                                       | ......
>80 52 43 4B 53 20                                              | ..RCKS..
>30 30 30 30 30 30 20 30 30 31 31 36 44 20 50 0D                | 000000..00116D..P..
```

### Running the Program

| | | |
|---|---|---|
| Output: | 80 52 55 4E 20 | RUN command |
| | 48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F 73 74 0D | |
| Input: | . . . | "Hi there..." messages from program loop |
| | 48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F 73 74 0D | |
| Output: | 80 45 4E 44 20 | END command |

*SMI Serial Data Analyzer (shows Output and Input)*

```
>80 52 55 4E 20                                           | ..RUN..
>48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F | Hi..there..Ethernet..ho
 73 74 0D                                                 | st..
>48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F | Hi..there..Ethernet..ho
 73 74 0D                                                 | st..
>48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F | Hi..there..Ethernet..ho
 73 74 0D                                                 | st..
>48 69 20 74 68 65 72 65 20 45 74 68 65 72 6E 65 74 20 68 6F | Hi..there..Ethernet..ho
 73 74 0D                                                 | st..
>80 45 4E 44 20                                           | ..END..
```

### Uploading the Program

| | | |
|---|---|---|
| Output: | 80 55 50 4C 4F 41 44 20 | UPLOAD command sent to motor |
| Input: | 57 48 49 4C 45 20 31 0A | 16 character block of upload from motor |
| Output: | 06 | ACK from host |
| Input: | 50 52 49 4E 54 32 28 22 | 16 character block of upload from motor |
| Output: | 06 | ACK from host |
| **Input:** | 48 69 20 74 68 65 72 65 | 16 character block of upload from motor |
| **Output:** | 06 | ACK from host |
| **Input:** | 20 45 74 68 65 72 6E 65 | 16 character block of upload from motor |
| **Output:** | 06 | ACK from host |
| **Input:** | 74 20 68 6F 73 74 22 2C | 16 character block of upload from motor |
| Output: | 06 | ACK from host |
| Input: | 23 31 33 29 0A 57 41 49 | 16 character block of upload from motor |
| Output: | 06 | ACK from host |
| Input: | 54 3D 31 30 30 30 0A 4C | 16 character block of upload from motor |
| **Output:** | 06 | ACK from host |
| **Input:** | 4F 4F 50 0A 45 4E 44 0A | 16 character block of upload from motor |

*SMI Serial Data Analyzer (shows Output and Input)*

```
>80 55 50 4C 4F 41 44 20                                  | ..UPLOAD..
>57 48 49 4C 45 20 31 0A                                  | WHILE..1..
>06                                                       | ..
>50 52 49 4E 54 32 28 22                                  | PRINT2("
>06                                                       | ..
>48 69 20 74 68 65 72 65                                  | Hi..there
>06                                                       | ..
>20 45 74 68 65 72 6E 65                                  | ..Etherne
>06                                                       | ..
>74 20 68 6F 73 74 22 2C                                  | t..host",
>06                                                       | ..
>23 31 33 29 0A 57 41 49                                  | #13)..WAI
>06                                                       | ..
>54 3D 31 30 30 30 0A 4C                                  | T=1000..L
>06                                                       | ..
>4F 4F 50 0A 45 4E 44 0A                                  | OOP..END..
```

# Troubleshooting

This table provides troubleshooting information for solving common problems. For additional support resources, see the Moog Animatics Support page at:

http://www.animatics.com/support.html

| Issue | Cause | Solution |
|---|---|---|
| Communication and Control Issues | | |
| Motor control power light does not illuminate. | Control power is off, disconnected or incorrectly wired. | Check that control power is connected to the proper pins and turned on. For connection details, see the installation and startup guide for your SmartMotor. |
| | Motor has routed drive power through drive-enable pins. | Ensure cabling is correct and drive power is not being delivered through the wrong connector. |
| Motor does not communicate with SMI. | Transmit, receive or ground pins are not connected correctly. | Ensure that transmit, receive and ground are all connected properly to the host PC. |
| | Motor program is stuck in a continuous loop or is disabling communications. | To prevent the program from running on power up, use the Communications Lockup Wizard located on the SMI software Communications menu. |
| Motor not detected or not communicating through TCP (Ethernet port in SMI). | IP address and/or netmask not set. | See IPCTL command, or check DHCP server. |
| | Feature disabled | See ETHCTL(100,<value>) command. |
| Motor disconnects from SMI sporadically. | COM port buffer settings are too high. | Adjust the COM port buffer settings to their lowest values. |
| | Poor connection on serial cable. | Check the serial cable connections and/or replace it. |
| | Power supply unit (PSU) brownout. | PSU may be too high-precision and/or undersized for the application, which causes it to brown-out during motion. Make moves less aggressive, increase PSU size or change to a linear unregulated power supply. |
| Red PWR SERVO light illuminated. | Critical fault. | To discover the source of the fault, use the Motor View tool located on the SMI software Tools menu. |
| Common Faults | | |
| Bus voltage fault. | Bus voltage is either too high or too low for operation. | Check servo bus voltage. If motor uses the DE power option, ensure that both drive and control power are connected. |
| Overcurrent occurred. | Motor intermittently drew more than its rated level of current. Does not cease motion. | Consider making motion less abrupt with softer tuning parameters or acceleration profiles. |
| Excessive temperature fault. | Motor has exceeded temperature limit of 85°C. Motor will remain unresponsive until it cools down below 80°C. | Motor may be undersized or ambient temperature is too high. Consider adding heat sinks or forced air cooling to the system. |
| Excessive position error. | The motor's commanded position and actual position differ by more than the user-supplied error limit. | Increase error limit, decrease load or make movement less aggressive. |
| Historical positive/negative hardware limit faults. | A limit switch was tripped in the past. | Clear errors with the ZS command. |
| | Motor does not have limit switches attached. | Configure the motor to be used without limit switches by setting their inputs as general use. |

| Issue | Cause | Solution |
|---|---|---|
| Programming and SMI Issues | | |
| Several commands not recognized during compiling. | Compiler default firmware version set incorrectly. | Use the Compiler default firmware version option in the SMI software Compile menu to select a default firmware version closest to the motor's firmware version. In the SMI software, view the motor's firmware version by right-clicking the motor and selecting Properties. |

# TAKE A CLOSER LOOK

Moog Animatics, a sub-brand of Moog Inc. since 2011, is a global leader in integrated automation solutions. With over 30 years of experience in the motion control industry, the company has U.S. operations and international offices as well as a network of Automation Solution Providers worldwide.

Moog Animatics
1995 NC Hwy 141
Murphy, NC 28906
United States

Email: animatics_sales@moog.com

For Animatics product information, visit **www.animatics.com**

For more information or to find the office nearest you, email **animatics_sales@moog.com**



Moog AnimaticsClass 6SmartMotor™Ethernet Serial Encapsulation Guide, Rev. E
SC80100017-001