PROFINET® IMPLEMENTATION FOR

# FULLY INTEGRATED SERVO MOTORS

## CLASS 6 SMARTMOTOR™ WITH COMBITRONIC™ TECHNOLOGY

Rev. H, February 2026

DESCRIBES THE CLASS 6 SMARTMOTOR™ SUPPORT FOR THE PROFINET® PROTOCOL

www.animatics.com

MOOG ANIMATICS

# Copyright Notice



Moog Animatics Class 6 SmartMotor™ PROFINET Guide, Rev. H, PN: SC80100007-001.



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 "PROFINET 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 manual explains the Moog Animatics Class 6 SmartMotor™ support for the PROFINET® protocol. It describes the major concepts that must be understood to integrate a SmartMotorfollower with a PLC or other PROFINET controller[1]. However, it does not cover all the low-level details of the PROFINET protocol.

**NOTE:** The feature set described in this version of the manual refers to motor firmware 6.0.2.41 (Class 6 M) / 6.4.2.50 (Class 6 D) or later.

This manual is intended for programmers or system developers who understand the use of PROFINET. (The PROFINET v2.2 specifications are detailed in the following IEC publications: IEC61158-6-10 Ed2.0, IEC61158-5-10 Ed2.0 and IEC61784-2 Ed2.0.) Therefore, this manual is not a tutorial on those specifications or the PROFINET protocol. Instead, it should be used to understand the specific implementation details for the Moog Animatics SmartMotor. Additionally, examples are provided for the various modes of motion and accessing those modes through PROFINET to operate the SmartMotor.

The Command and Response Code chapter of this manual includes details about the specific commands available in the SmartMotor through the PROFINET protocol. The commands include those required by the specification and those added by Moog Animatics. For details, see Command and Response Codes on page 50. Also, see User Program Commands on page 35.

In addition to this manual, it is recommended that you visit the PROFINET/PROFIBUS website (at http://www.profibus.com), where you will find documentation, tutorials, and other useful resources.

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

*Class 6 D-Style SmartMotor*

# PROFINET Overview

PROFINET is an independent, open fieldbus standard that allows different manufacturers of automation products to communicate without special interface adjustments. Specifically, PROFINET, which is

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

optimized for high speed, is designed to communicate between control systems and distributed I/O at the device level.

Moog Animatics has defined a set of 8-bit command and response codes to be transmitted and received over PROFINET. For details, see Command Packet Codes to Motor Commands on page 51. These codes generally correspond to Class 5 and Class 6 SmartMotor™ commands. To set target position, for example, the "set target position" command code is transmitted together with the data consisting of the target position value.

The PROFINET SmartMotor is a SmartMotor with the addition of the PROFINET connectors and interface board, which then accepts commands as a follower over a PROFINET network. In addition to communicating over PROFINET, SmartMotor commands may be sent through other communication interfaces of the SmartMotor. Depending on the SmartMotor model, it may also communicate over RS-232, RS-485 and/or USB.

The Moog Animatics communications profile over PROFINET is intended to integrate well with a PLC that continuously transmits and receives cyclic data. The command and response codes achieve this through a handshaking mechanism.

Certain configuration data is held in nonvolatile storage in the SmartMotor. Therefore, the motor data EEPROM must be correctly initialized before PROFINET operation.

A PROFINET Generic Station Description (GSD) configuration file, which is an XML file (also referred to as a "GSDML" file), is necessary for the host to configure the PROFINET controller and to connect to the follower motor. Make sure you obtain the latest version of the file, which is available from the Moog Animatics website Download Center. For more details, see Software on page 9.

Document sections include Output and Input data formats (PROFINET cargo), a list of the Moog Animatics PROFINET command codes explained in terms of the equivalent SmartMotor commands, and a list of Moog Animatics PROFINET response codes explained in terms of the equivalent SmartMotor commands.

# Equipment Required

The section describes the required PROFINET hardware and software.

## Hardware

The following hardware is required:

- Moog Animatics PROFINET SmartMotor™
- Moog Animatics power supply or user-supplied equivalent
- Moog Animatics RS-485 or USB communications cable that is compatible with the SmartMotor
- User-supplied PC with the Microsoft Windows operating system
- User-supplied PLC with PROFINET controller or other PROFINET controller
- Moog Animatics PROFINET cable, or equivalent, to connect the PLC to the SmartMotor's industrial Ethernet port (for details, see Motor Connectors and Pinouts on page 1)

## Software

The following software is required:

- User-supplied PLC configuration software
- Moog Animatics SMI software (latest version), which is available on the Moog Animatics website Support > Downloads > Software tab at:

  www.animatics.com/support/downloads.software.html

- Moog Animatics PROFINET GSDML file, which is available on the Moog Animatics website Products > SmartMotor > Resources tab at:

  www.animatics.com/products/smartmotor.resources.html

  After opening that page, click Fieldbus Configurator Files > PROFIBUS.

  **NOTE:** The PROFINET GSD configuration file name will have the form "GSDML-Vx.x-MOOG ANIMATICS-SMC06DEV01-date.XML", where 'x.x' is the version and 'date' is the release date. Make sure you obtain the latest version of the file.

# 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 11.

### 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*, which is available on the Moog Animatics website.

## 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*

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

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

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

- *SmartMotor™ Developer's Guide*

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

- *SmartMotor™ Homing Procedures and Methods Application Note*

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

- *SmartMotor™ System Best Practices Application Note*

  http://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*, which is available on the Moog Animatics website

  http://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:

  http://www.animatics.com

- Product information:

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

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

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

- Contact information, distributor locator tool, inquiries:

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

- Applications (Application Notes and Case Studies):

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

## PROFINET and PROFIBUS Resources

PROFINET and PROFIBUS are common standard maintained by PROFIBUS and PROFINET International (PI):

- PROFIBUS and PROFINET International (PI) website:

  http://www.profibus.com/

# Status LEDs

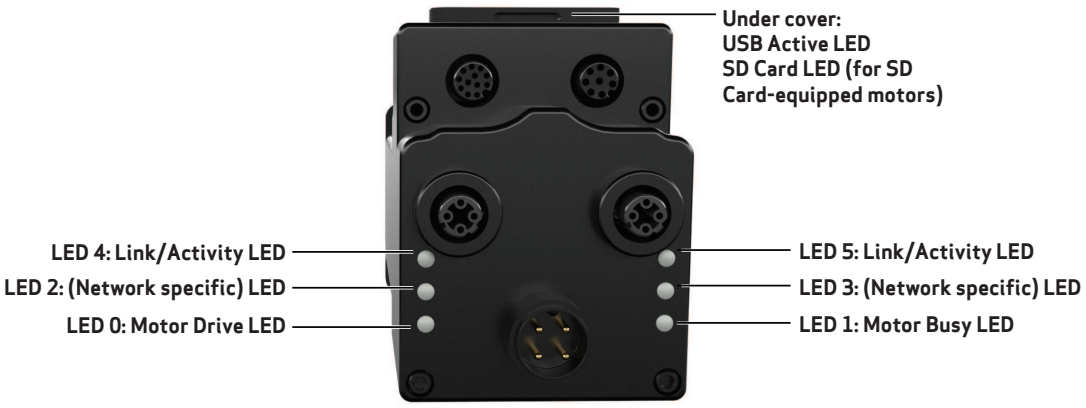
This chapter provides a description of the SmartMotor status LEDs.

**NOTE:** For information on the SmartMotor's connector pinouts and cable diagrams, 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.

# Status LEDs - Class 6 M-Style

This section describes the functionality of the Status LEDs on the Class 6 M-style 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.

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: PROFINET System Fail LED

| | |
|---|---|
| Off | No error |
| Flashing red | Network detected, configured, waiting for connection |
| Solid red | Application controller failure |

LED 3: PROFINET Bus Fail LED

| | |
|---|---|
| Off | No Error |
| Solid red | PROFINET Bus failed |

LED 4: Link/Activity LED

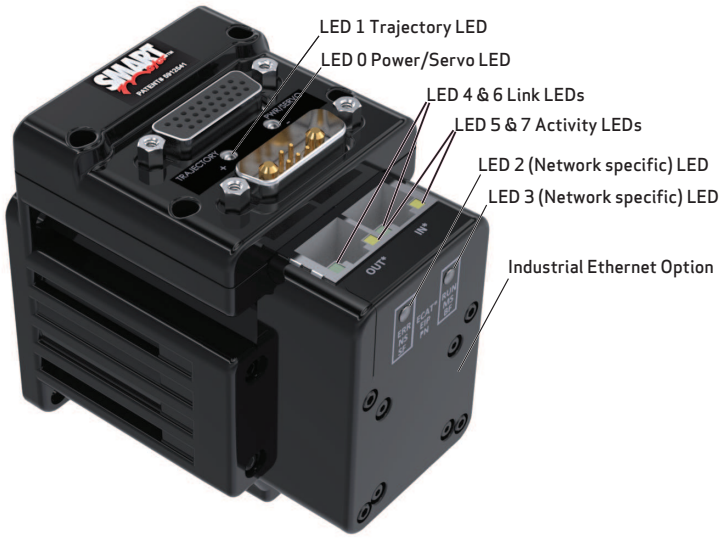
| | |
|---|---|
| Off | No/bad cable; no/bad Link port |
| Solid green | Link established |
| Blinking green | Activity |

LED 5: Link/Activity LED

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

# Status LEDs - Class 6 D-Style

This section describes the functionality of the Status LEDs on the Class 6 D-style 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: PROFINET System Fail LED

| | |
|---|---|
| Off | No error |
| Flashing red | Network detected, configured, waiting for connection |
| Solid red | Application controller failure |

LED 4 & 6 Link LEDs

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

Industrial Ethernet Option

LED 3: PROFINET Bus Fail LED

| | |
|---|---|
| Off | No Error |
| Solid red | PROFINET Bus failed |

LED 5 & 7 Activity LEDs

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

# PROFINET Configuration

The following sections describe how to configure your SmartMotor to communicate over PROFINET.

# Configure Motor with PC

Use the following procedure to configure the SmartMotor for communication with the PC. Refer to the figures in PROFINET Communication Example on page 23.

1. Connect the SmartMotor to the power supply.
2. If the motor is already configured, you may skip the balance of this procedure.
3. Connect the motor to the PC.
4. Launch the SmartMotor™ Interface (SMI) software, version 2.4.3.6 or later.

## User Program Requirements

No user program is specifically required by the Class 6 PROFINET SmartMotor.

## Required Nonvolatile EEPROM Values

The nonvolatile settings can be entered using the SMI software's Terminal window. For details on using the Terminal window, see the SMI software online help.

After the configuration settings have been entered, cycle the SmartMotor's power for the new configuration to take effect.

To change the nonvolatile station name for PROFINET within a user program, see the following code example:

```
...
SNAME("mymotor1")
a=ETH(0)
IF(a&2)
  Z       'Execute reset if Station Name changed
ENDIF
...
```

# Configure PLC with PC

Use the following procedure to configure the PLC for communication with the PC. Refer to the figures in PROFINET Communication Example on page 23.

**NOTE:** You may skip this section if the PLC is already configured.

1. Using the PLC configuration software running in a PC, load the SmartMotor's GSDML (XML) file, set it up as a PROFINET device from the catalog, and define the correct Station Name. For more details on the GSDML file, see Software on page 9.
2. Determine the location of the PLC memory to exchange three words (six bytes) of PROFINET output to the motor and the seven words (fourteen bytes) of input from the SmartMotor. The GSDML file defines the three output words and seven input words, but it does not specify where this is located in the PLC memory. That location is determined by the configuration tools supplied by the PLC manufacturer.

# Configure SmartMotor to PROFINET

Use the following procedure to configure the SmartMotor to PROFINET. Refer to the Status LEDs - Class 6 M-Style on page 16 or Status LEDs - Class 6 D-Style on page 17.

1. Verify the corresponding Link LED is ON (green) with possible occasional flashing, which indicates there is communication traffic.
2. After connecting the motor, the System Fail LED should go from solid red to flashing red, which indicates it is waiting for an I/O controller.
3. After the I/O controller makes a connection, the System Fail LED turns off.

## PLC Sends Commands to Motor

Program the PLC or modify by hand the PLC memory areas, as described below, to send the desired commands over PROFINET and communicate with the motor.

The following are sequences of commands sent, which show all the intermediary PROFINET packet output data states.

**NOTE:** Bold characters indicate changes in the PLC memory output buffer and input buffer values.

## Network Data Format Example

Each byte below is represented as two hexadecimal characters. For example, 7A represents hex 7A or decimal 122.

| COMMAND FROM I/O CONTROLLER | | | | | | RESPONSE FROM SMART MOTOR | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Cmd Code** | **Resp Code** | **Data** | | **Cmd Code Ack** | **Resp Code Ack** | **Resp Data** | **Status Word** | **Measured Position** | **Pos Error** |
| 00 | 7A | 0000 0000 | ..... | 00 | 00 | 0000 0000 | 0680 | 0000 0000 | 0000 |

The following are the SmartMotor's Status Word response bit definitions (the response shown above is 0680).

| Bit | Description |
|---|---|
| 0 | Busy Trajectory |
| 1 | Historical + limit (hardware and software limit) |
| 2 | Historical - limit (hardware and software limit) |
| 3 | Index report available for the rising edge of internal encoder |
| 4 | Position wraparound occurred |
| 5 | Position error fault |
| 6 | Temperature limit fault |
| 7 | Drive off |
| 8 | Index input active |
| 9 | + limit active (hardware and software limit) |
| 10 | - limit active (hardware and software limit) |
| 11 | Communication error of any type |
| 12 | Network user bit, defined by ETHCTL(12,x) command, see User Program Commands on page 35 |
| 13 | Command error (includes math and array errors) |
| 14 | Peak overcurrent occurred |
| 15 | Drive ready |

## PLC Memory

Each byte below is represented as two hexadecimal characters. For example, 0680 represents hex 680 or decimal 134.

| Output to follower motor: | Input from follower motor: |
|---|---|
| 3 two-byte words out | 7 two-byte words in |
| 0000 0000 0000 | 0000 0000 0000 0086 0000 0000 0000 |

A status word of 0x0680 (which breaks down to the bits 0000 0110 1000 0000) indicates the servo is off, the left and right limits have been activated, and the drive is not ready.

## Sequence to Set Report Data to Motor Clock

| Command Code | Response Code | Data | Motor Command |
|---|---|---|---|
| | 0x7A | | RCLK |

Insert response code 0x**7A** in the output buffer, which is being transmitted continuously (i.e., cyclically) by the controller to the follower motor. See Command Packet Codes to Motor Commands on page 51 to find response code RCLK and its value, hex 7A.

| | |
|---|---|
| 00**7A** 0000 0000 | 0000 0000 0000 0680 0000 0000 0000 |

Wait for response code acknowledge in the input buffer, which is being received continuously (i.e., cyclically) by the controller as a response from the follower motor. The clock data begins being cyclic updates.

| | |
|---|---|
| 007A 0000 0000 | 00**7A** 0000 03A1 0680 0000 0000 0000 |

As time goes on, the clock data is updated.

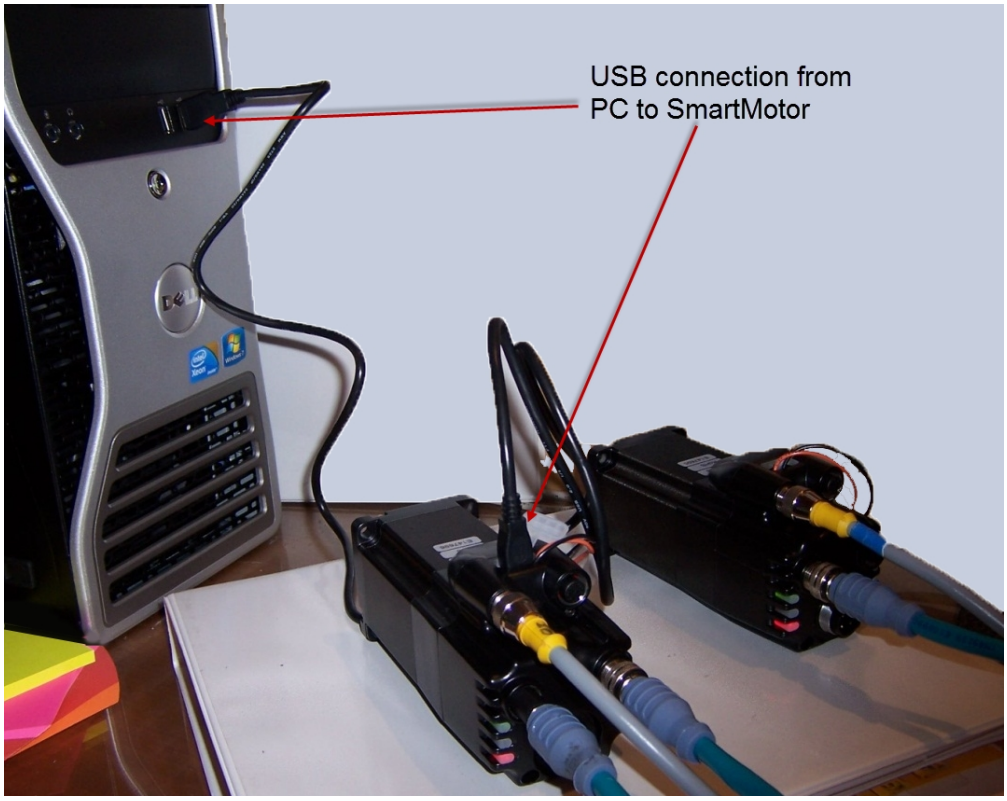
| | |
|---|---|
| 007A 0000 0000 | 007A **0001 B01A** 0680 0000 0000 0000 |

# PROFINET Communication Example

The following example illustrates PROFINET communications. It sends commands from a PLC over PROFINET to cause the SmartMotor to continually report its changing clock value to the PLC. The value is displayed by the PLC registers containing the PROFINET data received from the motor. It changes as the updated clock value is received.

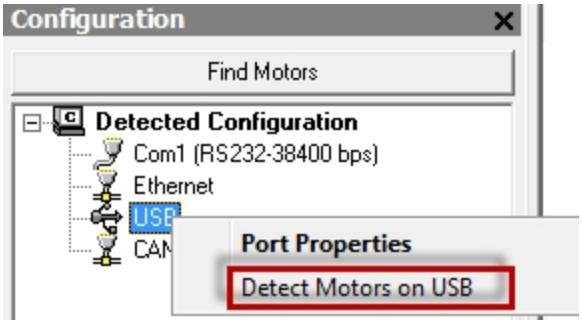
To create a PROFINET connection to the SmartMotor:

1. Install the SMI software. For more details, see the *Moog Animatics Class 6 SmartMotor™ Installation & Startup Guide*.
2. Connect control power to the 12-pin connector.
    a. Pin 11 is 24 Volt control power.
    b. Pin 12 is Ground or 24 Volt low.
3. Connect a USB cable from the PC to the USB connector on the SmartMotor. Refer to the following figure.

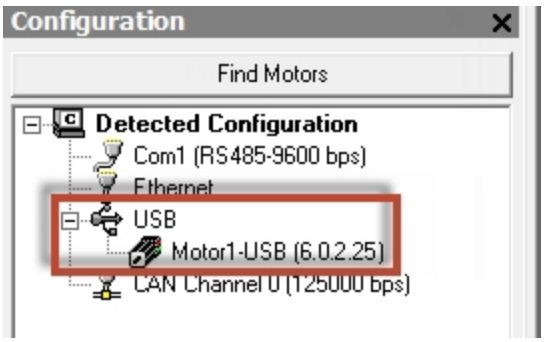



*USB Connection from PC to SmartMotor*

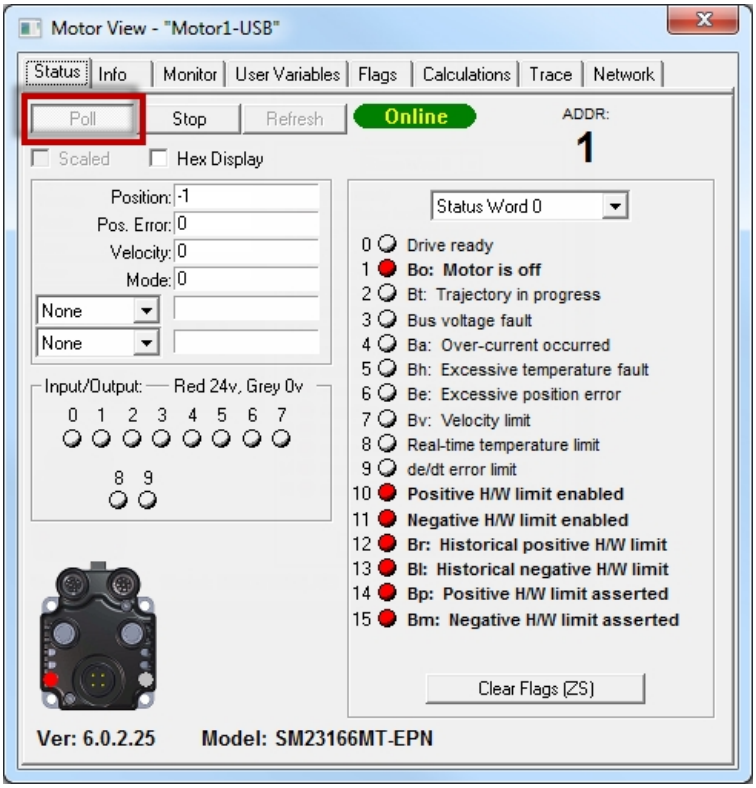
4. In the SMI software Configuration window, right-click the USB category and select Detect Motors on USB from the menu.



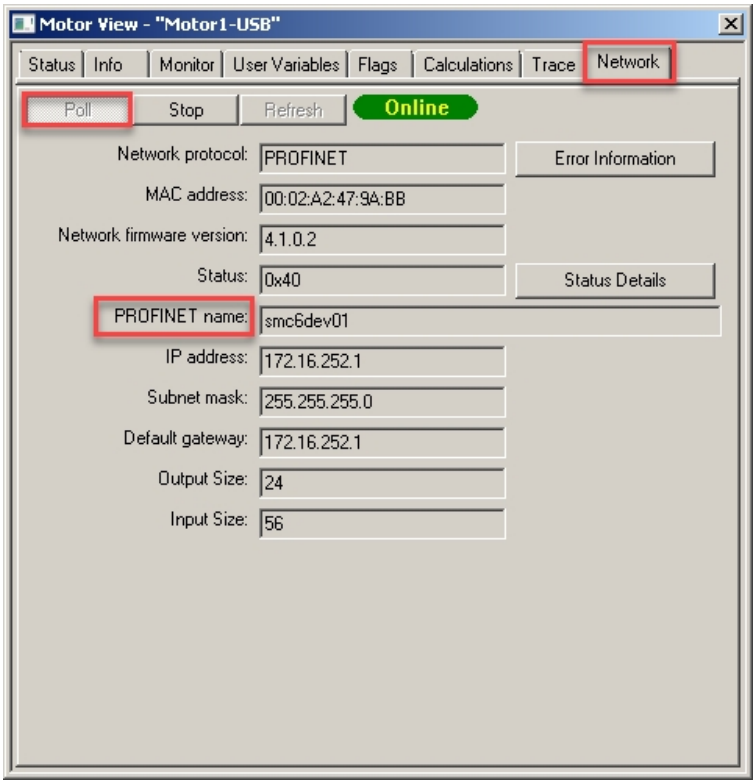

When detection has completed, Motor 1 will be shown under the USB network.




5. Double click Motor1 to open the Motor View tool. Click Poll to update the Status.



6. Select the Network tab and then Poll again. Note the default PROFINET name.



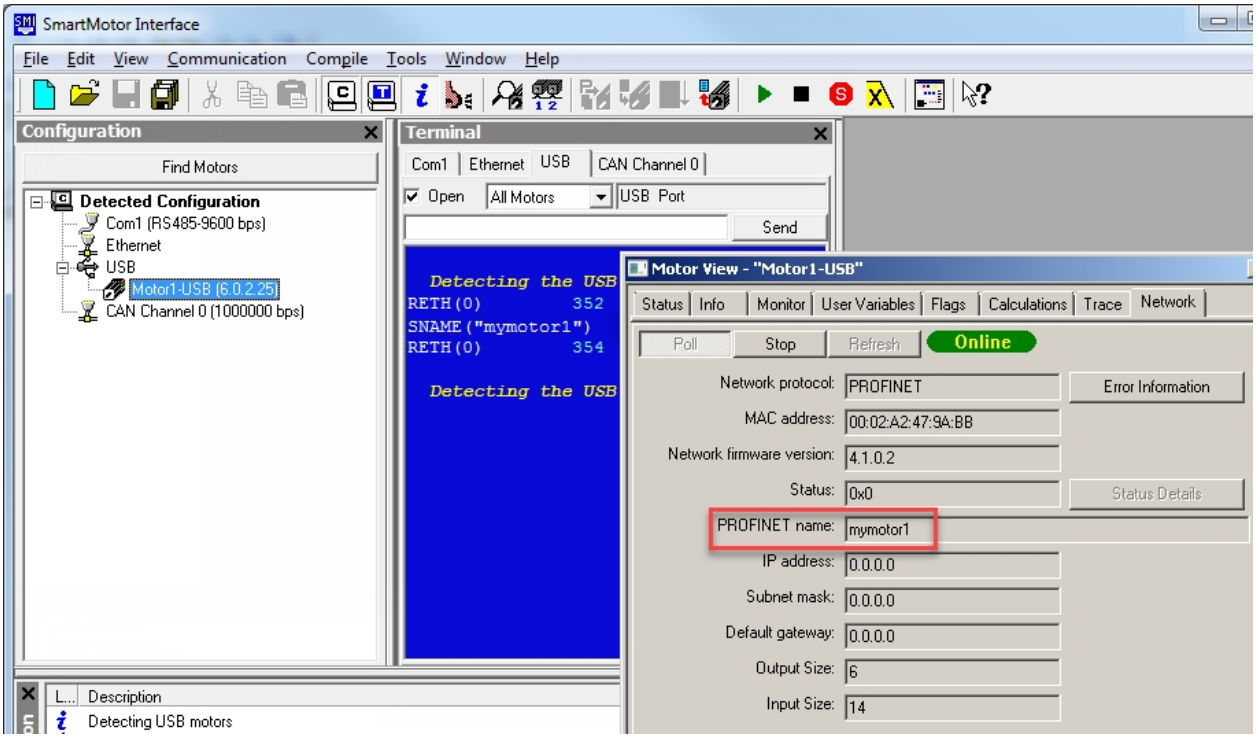

7. Set the station name. Refer to the Terminal window in the following figure.
    a. Execute RETH(0) to get the current Ethernet interface status bit (352 decimal = 160 hex).
    b. Type SNAME("mymotor1") into the Terminal window.

       Use the following PROFINET name conventions:

       - Characters a-z (lowercase only, uppercase is not permitted)
       - Numbers 0-9, but cannot start with a number
       - No underscores or other special characters; the hyphen and period are permitted, but not as the first or last character
       - No more than 127 characters total or 63 characters as a name component within the device name (e.g., a character string between two periods); may be further limited by your configuration software
       - Cannot be formatted as an IP address (dotted-decimal notation)
       - Cannot begin with the characters "port-nnn-", where "nnn" are three numeric characters 0-9 (e.g., port-735-)

    c. Execute RETH(0) to get the updated Ethernet interface status bit (354 decimal = 162 hex). After the station name has changed, the status for the report from RETH(0) should indicate a PROFINET status configuration change on Bit 1 (zero based). Refer to User Program Commands on page 35.

       **NOTE:** The new station name in the following figure (in the red box) won't be shown until power is cycled and the motor is redetected. Refer to the next step.




*Entering and Verifying the Station Name*

8. Cycle motor power to use the new configuration and station name. Refer to the red box in the previous figure.
9. Configure your PLC through its serial port using a PC that is running your PLC configuration software.
    a. Load the motor's PROFINET GSDML file.
    b. Assign and display the PLC registers associated with the motor's PROFINET input and output data.
10. Connect the PROFINET cable to the PLC and the SmartMotor.
11. Power cycle the SmartMotor to initialize it with the configured values.
12. Enter the PROFINET motor response code to report the motor clock in the PLC PROFINET data registers (i.e., in the "3 words out", the second byte is the motor response code).
    a. Using a PC that is running your PLC software, and with your PLC online, enter the PROFINET response code 122 decimal, x7A hex into the "response code" field.
    b. Watch the clock value being updated in your PLC PROFINET input registers "7 words in", bytes 2-5.

For examples of sending command sequences and communication handshaking, refer to Sample Command Sequences on page 27.

# Sample Command Sequences

This chapter contains sample PROFINET command sequences.

# Overview

These sequences illustrate:

- Disabling limits from preventing motion
- Turning the shaft in torque mode
- Moving a relative distance
- Command and response codes
- Handshaking of messages

## Command and Response Codes

The command and response codes are described in Command Packet Codes to Motor Commands on page 51 The symbolic command and response codes are listed, along with their values and the related SmartMotor™ command. See Output and Input Packets on page 41 for further explanation of how to use the command and response codes.

## Handshaking of Messages

Handshaking of output message changes is included in the protocol to ensure coherence in the packet. See Output and Input Packets on page 41 for an explanation of handshaking.

## Disabling Limits from Preventing Motion

At power up, if limit switches are not connected to the motor, the electrical state of the limit pins will default to indicate that the motor is at the limits. This will prevent motion unless the limits are disabled and any limit faults are cleared.

These commands may be included in the user program that is downloaded to the motor and runs at power up. If the user program does *not* include these commands or the limits are not held inactive at power-up, before attempting to turn the motor shaft, you must perform the command sequence described in Disable Limits and Clear Fault Status on page 29.

## Turning the Motor Shaft

After disabling the limits and clearing any faults, the shaft may be turned using the following command sequences:

- Initiate Mode Torque on page 31
- Initiate Relative Position Move on page 33

These sequences are described in following sections.

# Disable Limits and Clear Fault Status

## Commands

| Command Code | Response Code | Data | Resulting SmartMotor Command |
|---|---|---|---|
| 0x01 | | 0x30 | EIGN(2) |
| 0x01 | | 0x33 | EIGN(3) |
| 0x01 | | 0x44 | ZS |

## PLC Memory

**Output to follower motor:**

3 words out

0000 0000 0000

**Input from follower motor:**

7 words in

0000 0000 0000 0680 0000 0000 0000

| Cmd Code | Resp Code | Data | | Cmd Code Ack | Resp Code Ack | Resp Data | Status Word | Measured Position | Pos Error |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 7A | 0000 0000 | ..... | 00 | 00 | 0000 0000 | 0680 | 0000 0000 | 0000 |

## Disable positive limit, command EIGN(2)

Insert command EIGN(2) data = 0x30 in the output buffer, which is being transmitted continuously (i.e., cyclically) by the controller to the follower motor.

0000 0000 00**30** 0000 0000 0000 0680 0000 0000 0000

Set command code 0x01 in the output buffer.

**01**00 0000 0030 0000 0000 0000 0680 0000 0000 0000

Wait for a command code acknowledge in the input buffer, which is being received continuously (i.e., cyclically) by the controller as a response from the follower motor.

0100 0000 0030 **01**00 0000 0000 0480 0000 0000 0000

The command code acknowledges the motor has received the command.

Clear the command code in the output buffer (handshake) to prepare for the next command.

**00**00 0000 0030 0100 0000 0000 0480 0000 0000 0000

Wait for acknowledgment of the cleared command code.

0000 0000 0030 **00**00 0000 0000 0480 0000 0000 0000

## Disable negative limit, command EIGN(3)

Insert command EIGN(3) data = 0x33 in the output buffer, which is being transmitted continuously (i.e., cyclically) by the controller to the follower motor.

0000 0000 00**33** 0000 0000 0000 0480 0000 0000 0000

Set command code 0x01in the output buffer.

| | |
|---|---|
| **01**00 0000 0033 | 0000 0000 0000 0480 0000 0000 0000 |

Wait for command code acknowledge in the input buffer, which is being received continuously (i.e., cyclically) by the controller as a response from the follower motor.

| | |
|---|---|
| 0100 0000 0033 | **01**00 0000 0000 **0080** 0000 0000 0000 |

The command code acknowledges the motor has received the command.

Clear the command code in the output buffer (handshake) to prepare for the next command:

| | |
|---|---|
| **00**00 0000 0033 | 0100 0000 0000 0080 0000 0000 0000 |

Wait for acknowledgment of the cleared command code.

| | |
|---|---|
| 0000 0000 0033 | **00**00 0000 0000 0080 0000 0000 0000 |

## Clear fault status, command ZS

Insert command ZS data = 0x44 in output buffer, which is being transmitted continuously (i.e., cyclically) by the controller to the follower motor.

| | |
|---|---|
| 0000 0000 00**44** | 0000 0000 0000 **0086** 0000 0000 0000 |

Set command code 0x01in the output buffer.

| | |
|---|---|
| **01**00 0000 0044 | 0000 0000 0000 0086 0000 0000 0000 |

Wait for command code acknowledge in the input buffer, which is being received continuously (i.e., cyclically) by the controller as a response from the follower motor. Fault status is reported cleared to 0x**0080**.

| | |
|---|---|
| 0100 0000 0044 | **01**00 0000 0000 **0080** 0000 0000 0000 |

The command code acknowledges the motor has received the command.

Clear command code in output buffer (handshake) to prepare for the next command.

| | |
|---|---|
| **00**00 0000 0044 | 0100 0000 0000 0080 0000 0000 0000 |

Wait for acknowledgment of the cleared command code.

| | |
|---|---|
| 0000 0000 0044 | **00**00 0000 0000 0080 0000 0000 0000 |

# Initiate Mode Torque

## Commands

| Command Code | Response Code | Data | Resulting SmartMotor Command |
|---|---|---|---|
| 0x94 | 0xA2 | 3072 (0x0c00) | T=3072<br>RVA (polled motor response) |
| 0x01 | 0xA2 | 0x21 | MT<br>RVA (polled motor response) |
| 0x01 | | 0x0C | G (begin motion) |

## PLC Memory

| Output to follower motor: | Input from follower motor: |
|---|---|
| 3 words out | 7 words in |
| 0000 0000 0000 | 0000 0000 0000 0080 0000 0000 0000 |

## Set torque value, specify the response data

This will command T=3072 and specify the response data to be the current velocity.

Begin to set torque T=3072 by putting **x 00 00 0C 00** in output data.

0000 **0000 0C00** 0000 0000 0000 0080 0000 0000 0000

Insert command code 0x**94** and response code 0x**A2**.

**94A2** 0000 0C00 0000 0000 0000 0080 0000 0000 0000

Wait for acknowledge in input buffer:

94A2 0000 0C00 **94A2** 0000 0000 0080 0000 0000 0000

Now, T=3072 (0x0c00), and the response data value will be velocity. Clear the command code output buffer (handshake) to prepare for the next command.

**00**A2 0000 0C00 94A2 0000 0000 0080 0000 0000 0000

Wait for acknowledgment of command code clear in input buffer.

00A2 0000 0C00 **00**A2 0000 0000 0080 0000 0000 0000

## Initiate torque mode, command MT

Insert command 0x21 data to begin torque mode.

00A2 **0000 0021** 00A2 0000 0000 0080 0000 0000 0000

Insert command code 0x01.

**01**A2 0000 0021 00A2 0000 0000 0080 0000 0000 0000

Wait for command code 1 acknowledgment.

| | |
|---|---|
| 01A2 0000 0021 | **01**A2 0000 0000 0080 0000 0000 0000 |

Insert command code 0x00.

| | |
|---|---|
| **00**A2 0000 0021 | 01A2 0000 0000 0080 0000 0000 0000 |

Wait for command code 0 acknowledgment.

| | |
|---|---|
| 00A2 0000 0021 | **00**A2 0000 0000 0080 0000 0000 0000 |

Insert command 0x0C data to initiate open-loop motion.

| | |
|---|---|
| 00A2 0000 **000C** | 00A2 0000 0000 0080 0000 0000 0000 |

Insert command code 0x01.

| | |
|---|---|
| **01**A2 0000 000C | 00A2 0000 0000 0080 0000 0000 0000 |

When the command is received by the motor, the motor shaft will begin turning if it is not in a fault state.

Wait for command code acknowledgment in the input buffer.

| | |
|---|---|
| 01A2 0000 000C | **01**A2 0000 0000 0080 0000 0000 0000 |

Velocity becomes nonzero, and it is reported as 0x**00 14 00 00** in this example. Status changes are reported as 0x**0009** in this example. Position becomes nonzero, and it is reported as 0x**00 00 00 A2** in this example.

| | |
|---|---|
| 01A2 0000 000C | 01A2 **0014 0000 0009 0000 00A2** 0000 |

Insert command code 0x00 to clear the command code output buffer (handshake) to prepare for the next command. The position is continually updated. Velocity is a filtered value measured in:

encoder counts per sample period x 65,536

| | |
|---|---|
| **00**A2 0000 000C | 01A2 0014 0000 0009 **0000 02EE** 0000 |

Wait for the command code clear acknowledge in the input buffer.

| | |
|---|---|
| 00A2 0000 0000 | **00**A2 0014 0000 0009 **0000 05DC** 0000 |

Set data to 0.

| | |
|---|---|
| 0000 0000 0000 | 00A2 0014 0000 0009 0000 05DC 0000 |

# Initiate Relative Position Move

## Commands

| Command Code | Response Code | Data | Resulting SmartMotor Command |
|---|---|---|---|
| 0x64 | | 255 (0xff) | ADT=255 |
| 0xA3 | | 100000 | VT=100000 |
| 0x01 | | 0x1D | Change to Mode Position (MP) |
| | 0xA2 | | RVA (polled motor response) |
| 0x03 | | 10000 | PRT=10000 G |

## PLC Memory

**Output to follower motor:**
3 words out
0000 0000 0000

**Input from follower motor:**
7 words in
0000 0000 0000 0080 0000 0000 0000

## Set acceleration value, command ADT=255

Begin to set ADT=255 by putting x**00 00 00 FF** in output data.

0000 **0000 00FF** 0000 0000 0000 0080 0000 0000 0000

Insert command code 0x64 and response code 0xA2.

**64A2** 0000 00FF 0000 0000 0000 0080 0000 0000 0000

Wait for acknowledge in input buffer.

64A2 0000 00FF **64A2** 0000 0000 0080 0000 0000 0000

Now, ADT=255, and the response data value will be velocity. Clear the command code output buffer (handshake) to prepare for the next command.

**00**A2 0000 00FF 64A2 0000 0000 0080 0000 0000 0000

Wait for acknowledge of command code clear in input buffer.

00A2 0000 00FF **00**A2 0000 0000 0080 0000 0000 0000

## Set maximum velocity value, command VT=100000

Insert code commanded velocity of VT=100000 = 0x**0001 86A0**.

00A2 **0001 86A0** 00A2 0000 0000 0080 0000 0000 0000

Insert command code 0x**A3** to set VT=100000.

**A3**A2 0001 86A0 00A2 0000 0000 0080 0000 0000 0000

Wait for command code acknowledge in the input buffer.

A3A2 0001 86A0 **A3**A2 0000 0000 0080 0000 0000 0000

Insert command code 0x**00**.

**00**A2 0001 86A0 A3A2 0000 0000 0080 0000 0000 0000

Wait for command code acknowledge in the input buffer.

00A2 0001 86A0 **00**A2 0000 0000 0080 0000 0000 0000

Insert data 0x**0000 001D** for MP when command is 1.

00A2 0000 001D 00A2 0000 0000 0080 0000 0000 0000

Insert command code 0x**01**.

**01**A2 0001 86A0 00A2 0000 0000 0080 0000 0000 0000

Wait for command code acknowledge in the input buffer.

01A2 0001 86A0 **01**A2 0000 0000 0080 0000 0000 0000

Insert command code 0x**00**.

00A2 0001 86A0 **01**A2 0000 0000 0080 0000 0000 0000

## Make a relative position move

Insert data for a relative move of 10,000 counts = 0x**0000 2710**.

00A2 **0000 2710** 00A2 0000 0000 0080 0000 0000 0000

Insert command code value 0x03.

**03**A2 0000 2710 00A2 0000 0000 0080 0000 0000 0000

Wait for command code acknowledge in the input buffer.

03A2 0000 2710 **03**A2 0000 0000 0080 0000 0000 0000

The motor performs its move. While the trajectory is in the slew phase, you will see something like:

03A2 0000 2710 03A2 0001 86AD 0009 0000 CA23 0011

which is the following input data:

| | |
|---|---|
| command code acknowledge | 03 |
| response code acknowledge | A2 |
| response data current velocity (100,000 in slew) | 0001 86AD |
| status | 0009 |
| Bt = 1 | |
| Bi = 1 | |
| measured current position | 0000 CA23 |
| measured current position error | 0011 |

# User Program Commands

The SmartMotor's EEPROM can store nonvolatile PROFINET information about the network. For proper PROFINET operation, each SmartMotor must have a unique station name set with the SNAME instruction. This is can be accomplished: at the PLC over PROFINET; with SMI and a USB connection over channel 8, or RS-485 on channel 0; with a SmartMotor user program.

**NOTE:** Nonvolatile memory will be read at power-up or after the Z (reset) command has been executed.

The following sections list the commands used to operate the motor on a PROFINET network.

# SNAME("string")

***Set PROFINET Station Name***

The SNAME command is used to set a unique PROFINET station name. The setting is nonvolatile. It can use up to 54 characters; the factory default is smc6dev01. It will set the configuration change bit (Bit 1) returned by the ETH/RETH command (see below) if the Station Name has changed from the previous value in EEPROM.

Use the following PROFINET name conventions:

- Characters a-z (lowercase only, uppercase is not permitted)
- Numbers 0-9, but cannot start with a number
- No underscores or other special characters; the hyphen and period are permitted, but not as the first or last character
- No more than 127 characters total or 63 characters as a name component within the device name (e.g., a character string between two periods); may be further limited by your configuration software
- Cannot be formatted as an IP address (dotted-decimal notation)
- Cannot begin with the characters "port-nnn-", where "nnn" are three numeric characters 0-9 (e.g., port-735-)

# IPCTL(function,"string")

***Sets IP address, Mask, or Gateway***

The IPCTL command is used to set the IP address, subnet mask or gateway. The setting is nonvolatile. This command is not usually needed. Typically, the PLC will handle these settings during PROFINET network initialization.

The possible function values are:

0: set IP address

1: set Mask

2: set Gateway

The "string" is formatted as an IP address, e.g., IPCTL(0,"192.168.0.10"). By default, these values are set to 0 (i.e., "0.0.0.0").

# =ETH, RETH

***Get PROFINET error***

The =ETH and RETH commands are used to assign/report errors and certain status information for the PROFINET bus.

- Assigned to a program variable: x=ETH(y)
- As a report: RETH(y)

Refer to the following table for details.

| Assignment | Report | Description |
|---|---|---|
| =ETH(0) | RETH(0) | Gets the PROFINET status bits:<br>0 Initialization incomplete<br>Read specific error code from ETH(54);<br>Contact Moog Animatics<br>(Confirm SNAME setting if encountering a problem)<br>1 Configuration change<br>2 Reserved<br>3 Network processor failure<br>Likely due to excessive control power supply noise or ESD event<br>4 Reserved<br>5 Reserved<br>6 I/O Controller is STOP<br>7 I/O Controller is RUN<br>8 I/O Controller aborted cyclic communications<br>9 Network commanded configuration change |
| =ETH(5) | RETH(5) | LFW firmware version as 32-bit integer; e.g., 3.1.0.1 would be a value 50397185 (0x03010001). |
| =ETH(6) | RETH(6) | The current Network Lost program label number. For details, see ETHCTL (function, value) on page 38. |
| =ETH(7) | RETH(7) | Processor type:<br>-1 Failed<br>0 Unknown<br>1 netX 10<br>2 netX 50<br>3 netX 51/52<br>4 netX 100 |
| =ETH(8) | RETH(8) | Protocol type after successful initialization (Confirm SNAME setting if encountering a problem) See also RETH(19)<br>0 Not defined<br>1 PROFINET<br>2 EtherCAT<br>3 EtherNet/IP |
| =ETH(9) | RETH(9) | The current value assigned to the Network Lost action. For details, see ETHCTL(function, value) on page 38. |
| | RETH(15) | IP address; value is in dotted-decimal format; report only. |
| | RETH(16) | Subnet mask; value is in dotted-decimal format; report only. |
| | RETH(17) | Gateway; value is in dotted-decimal format; report only. |
| | RETH(18) | MAC ID string formatted; report only; e.g., 00:01:02:a9:ff:00 |

| Assignment | Report | Description |
|---|---|---|
| =ETH(19) | RETH(19) | Report the detected LFW Protocol Class. This gives a wider range of values than the known and supported protocols listed in ETH (8). Values designated according to NXF/LFW file loaded into network processor and too numerous to list here. These are the values for the supported protocols: (introduced in firmware 6.0.2.41 and 6.4.2.50 or later)<br>0 Not Defined<br>21 PROFINET<br>9 EtherCAT<br>10 Ethernet/IP<br>... ... |
| =ETH(30) | RETH(30) | Gets the present receive I/O data size in bytes |
| =ETH(31) | RETH(31) | Gets the present transmit I/O data size in bytes |
| =ETH(45) | RETH(45) | IP address as integer; e.g., for an IP address of 192.168.1.3 (C0 A8 01 03 hex), this command reports -1062731517 (it reports as a 32-bit signed value). |
| =ETH(46) | RETH(46) | IP subnet mask as integer; e.g., for an IP netmask of 255.255.0.0 (FF FF 00 00 hex), this command reports -65536 (it reports as a 32-bit signed value). |
| =ETH(47) | RETH(47) | IP gateway as integer; e.g., for an IP gateway of 192.168.1.1 (C0 A8 01 01 hex), this command reports -1062731519 (it reports as a 32-bit signed value). |
| =ETH(48) | RETH(48) | Low 3 bytes of MAC ID (device ID) as integer; e.g., for a MACID of 00:01:02:a9:ff:00, this command reports 11140864 (00 a9 ff 00 hex). |
| =ETH(49) | RETH(49) | High 3 bytes of MAC ID (device ID) as integer; e.g., for a MACID of 00:01:02:a9:ff:00, this command reports 258 (00 00 01 02 hex). |
| =ETH(50) | RETH(50) | Gets the last internal error code |
| =ETH(51) | RETH(51) | Gets the last internal error code source |
| =ETH(54) | RETH(54) | Gets the Initialization error code; for further information, read this error when RETH(0) bits 0 or 1 are indicated, or when RETH(8) returns 0.<br><br>The value -1070596029 indicates an invalid SNAME format was used. |
| =ETH(57) | RETH(57) | Gets the real-time Ethernet sync correction |
| =ETH(58) | RETH(58) | Gets the real-time Ethernet sync count |

## ETHCTL(function, value)

***Control network features***

Commands execute based on the function argument, which controls Ethernet functions. After issuing an ETHCTL command the Ethernet error codes will be checked to determine the state of Status Word 2, bit 6 (Ethernet error).

| Command | Description |
|---|---|
| ETHCTL(1,TBD)<br>...<br>ETHCTL(5,TBD) | Reserved for future use. |

| Command | Description |
|---|---|
| ETHCTL(6,<value>) | User program label number. This setting is nonvolatile.<br>Program label to jump to if the NET_LOST_LABEL option is chosen from the NET_LOST_ACTION function.<br>This function has no effect if the NET_LOST_ACTION is anything other than NET_LOST_LABEL. |
| ETHCTL(7,TBD) | Reserved for future use. |
| ETHCTL(8,TBD) | Reserved for future use. |
| ETHCTL(9,<value>) | PROFINET Network Lost Action. This setting is nonvolatile.<br>0 Ignore, no action<br>1 Send OFF command to motor<br>2 Send X command to motor (soft stop) - default setting<br>3 Send S command to motor (immediate stop)<br>4 Send GOSUB(x) command, where x is the value of the user program label.<br>5 Send GOTO(x) command, where x is the value of the user program label.<br>NOTE: Loss of network is an edge-triggered event if I/O Control goes from RUN to any other state. |
| ETHCTL(10,x) | Allows the position field of 14 byte (7 word) input module to be reconfigured for alternate data from the motor. See 240, xF0 on page 57. |
| ETHCTL(11,TBD) | Reserved for future use. |
| ETHCTL(12,<value>) | Network user bit set or clear. This is a bit in the status word of the 14 byte (7 word) input module. Also visible in response code 164 "legacy status word":<br>0 Clear Bit 12 of SmartMotor I/O Network Bit<br>1 Set Bit 12 of SmartMotor I/O Network Bit |
| ETHCTL(45,x) | Set IP address as integer; e.g., to set for an IP address of 192.168.1.3 (C0 A8 01 03 hex), x=3232235779; non-volatile. |
| ETHCTL(46,x) | Set IP subnet mask as integer; e.g., to set for an IP netmask of 255.255.0.0 (FF FF 00 00 hex), x=4294901760; non-volatile. |
| ETHCTL(47,x) | Set IP gateway as integer; e.g., to set for an IP gateway of 192.168.1.1 (C0 A8 01 01 hex), x=3232235777; non-volatile. |
| ETHCTL(50,<value>) | Resets the internal error register: RETH(50); the value argument is ignored. |
| ETHCTL(51,<value>) | Resets the internal error register: RETH(51); the value argument is ignored. |
| ETHCTL(58,<value>) | Clears the real-time Ethernet sync count |

## Program Example

The following code example sets the nonvolatile station name.

```
SNAME("mymotor1")
a=ETH(0)
IF( a&2 )
  Z           'Execute reset if station name changed
ENDIF

'Add rest of program below
```

# Output and Input Packets

This section describes the PROFINET Output and Input packet format. It also provides notes for the Command (Output) packets and Response (Input) Packets.

# Output and Input Packet Format

Two options exist for the input/output packet size:

- 3 words (6 bytes) out, 7 words (14 bytes) in
- 12 words (24 bytes) out, 28 words (56 bytes) in
  This option in Class 6M is only available with firmware 6.0.2.41 or later, and requires XML (GSDML) date -20190118 or later. For Class 6D, firmware: 6.4.2.50 with XML (GSDML) date -20220114 or later is required. The I/O controller (PLC) will have an interface for loading the XML file and choosing these options for the input and output packets. Some tools may provide a drag-and-drop interface for selecting the available modules from the XML file and placing them into the 2 available slots in the motor.

**NOTE:** Only the two combinations of specific in/out sizes are allowed. For example, 3 words out cannot be used with 28 words in.

### Output (3 Words) Data Format (I/O Controller Command)

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 0 | 0 | Command Code | | N/A |
| | 1 | Response Code | | N/A |
| 1 | 2 | Command Data Value (32 bits), big-endian format | | Yes |
| | 3 | | | |
| 2 | 4 | | | |
| | 5 | | | |

### Input (7 words) Data Format (Motor response)

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 0 | 0 | Command Code Acknowledge | | N/A |
| | 1 | Response Code Acknowledge | | N/A |
| 1 | 2 | Response Data Value (32 bits), big-endian format | | Yes |
| | 3 | | | |
| 2 | 4 | | | |
| | 5 | | | |
| 3 | 6 | Status Word (16 bits), big-endian format | | Yes |
| | 7 | | | |

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 4 | 8 | Measured Position (32 bits), big-endian format<br>NOTE: This field can be configured to report al[0] or af[0]. | | Yes |
| | 9 | | | |
| 5 | 10 | | | |
| | 11 | | | |
| 6 | 12 | Position Error (16 bits), big-endian format | | Yes |
| | 13 | | | |

**Command Code:** Indicates a command to be issued to the SmartMotor. Also, see Command Data Value.

**Response Code:** Indicates additional data to be included in the Response Data Value of the Input Data.

**Command Data Value:** Indicates the 32-bit value to be used in conjunction with the Command Code.

**Command Code Acknowledge:** Returned in the Input Data to indicate that a Command Code was processed.

**Response Code Acknowledge:** Returned in the Input Data to indicate that a Response Code was processed and that the current Response Data Value corresponds to that Response Code.

**Response Data Value:** 32-bit value returned in the Input Data in response to a Response Code.

**Status Word:** SmartMotor's current status word (16 bit).

**Measured Position:** SmartMotor's current measured position value (32-bit); result of RPA command.

**Position Error:** SmartMotor's current commanded trajectory position less the current measured position.

**Extended format:** 12 words (24 bytes) out, 28 words (56 bytes) in. This format was created for the purpose of easier access to reading data cyclically from the motor. Up to 8 response data items can be read on every cycle, though the first reponse item is required for certain special items, see notes.

There is still only 1 command code item because most commands require a sequence of events to operate correctly, and multiple fields could cause conflict.

The command and response codes have been extended to 16-bit values in this extended format. This allows for a wider set of codes and direct access to variables. See Extended 16-bit command codes on page 58, and Extended 16-bit response codes on page 66.

Several pre-set fields have been removed from the extended format: position, status word, position error. These are still available in the list of response codes (codes 141, 164, 143 respectively), and can be selected by setting those response request codes in the output data. Position error is a full 32-bit when accessed by this method.

**NOTE:** The PROFINET SmartMotor supports the byte/word swap parameter as of firmware 6.0.2.41 or 6.4.2.50 or later

**Output (12 Words) Data Format (I/O Controller Command)**

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 0 | 0 | Reserved (write as 0x00) | 4 | N/A |
| | 1 | Reserved (write as 0x00) | 4 | N/A |

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 1 | 2 | Command Data Value (32 bits) | | Yes |
| | 3 | | | |
| 2 | 4 | | | |
| | 5 | | | |
| 3 | 6 | Command code request (16 bits) | | Yes |
| | 7 | | | |
| 4 | 8 | Response code 0 request (16 bits)<br>See note 1: some codes must use of this section | 1 | Yes |
| | 9 | | | |
| 5 | 10 | Response code 1 request (16 bits) | 2 | Yes |
| | 11 | | | |
| 6 | 12 | Response code 2 request (16 bits) | 2 | Yes |
| | 13 | | | |
| 7 | 14 | Response code 3 request (16 bits) | 2 | Yes |
| | 15 | | | |
| 8 | 16 | Response code 4 request (16 bits) | 2 | Yes |
| | 17 | | | |
| 9 | 18 | Response code 5 request (16 bits) | 2 | Yes |
| | 19 | | | |
| 10 | 20 | Response code 6 request (16 bits) | 2 | Yes |
| | 21 | | | |
| 11 | 22 | Response code 7 request (16 bits) | 2 | Yes |
| | 23 | | | |

**Input (28 words) Data Format (Motor response)**

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 0 | 0 | Reserved (reports as 0xFF) | 3 | N/A |
| | 1 | Reserved (reports as 0xFF) | 3 | N/A |
| 1 | 2 | Reserved (ignore, reports as 0) | | N/A |
| | 3 | Reserved (ignore, reports as 0) | | N/A |
| 2 | 4 | Reserved (ignore, reports as 0) | | N/A |
| | 5 | Reserved (ignore, reports as 0) | | N/A |
| 3 | 6 | Command code ack (16 bits) | | Yes |
| | 7 | | | |

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 4 | 8 | Response code 0 ack (16 bits)<br>See note 1: some codes must use of this section | 1 | Yes |
| | 9 | | | |
| 5 | 10 | Response code 1 ack (16 bits) | 2 | Yes |
| | 11 | | | |
| 6 | 12 | Response code 2 ack (16 bits) | 2 | Yes |
| | 13 | | | |
| 7 | 14 | Response code 3 ack (16 bits) | 2 | Yes |
| | 15 | | | |
| 8 | 16 | Response code 4 ack (16 bits) | 2 | Yes |
| | 17 | | | |
| 9 | 18 | Response code 5 ack (16 bits) | 2 | Yes |
| | 19 | | | |
| 10 | 20 | Response code 6 ack (16 bits) | 2 | Yes |
| | 21 | | | |
| 11 | 22 | Response code 7 ack (16 bits) | 2 | Yes |
| | 23 | | | |
| 12 | 24 | Response data 0 (32 bits)<br>See note 1: some codes must use of this section | | Yes |
| | 25 | | | |
| 13 | 26 | | | |
| | 27 | | | |
| 14 | 28 | Response data 1 (32 bits) | | Yes |
| | 29 | | | |
| 15 | 30 | | | |
| | 31 | | | |
| 16 | 32 | Response data 2 (32 bits) | | Yes |
| | 33 | | | |
| 17 | 34 | | | |
| | 35 | | | |
| 18 | 36 | Response data 3 (32 bits) | | Yes |
| | 37 | | | |
| 19 | 38 | | | |
| | 39 | | | |
| 20 | 40 | Response data 4 (32 bits) | | Yes |
| | 41 | | | |
| 21 | 42 | | | |
| | 43 | | | |

| Offset | | Description | Notes | Affected by byte / word swap parameter |
|---|---|---|---|---|
| Word | Byte | | | |
| 22 | 44 | Response data 5 (32 bits) | | Yes |
| | 45 | | | |
| 23 | 46 | | | |
| | 47 | | | |
| 24 | 48 | Response data 6 (32 bits) | | Yes |
| | 49 | | | |
| 25 | 50 | | | |
| | 51 | | | |
| 26 | 52 | Response data 7 (32 bits) | | Yes |
| | 53 | | | |
| 27 | 54 | | | |
| | 55 | | | |

**NOTE:** 1) This response request code location is allowed to include attributes (request codes) 214-225 (special access to variables and EEPROM). It is recommended to reserve this slot for this type of access. For constantly read information, like access to variables (using response codes above 255), position, etc., use response code locations 1-7 instead.

**NOTE:** 2) This response request code location is not allowed to use attributes (request codes) 214-225, an error code (255) will result. These request code locations 1-7 are recommended for information that must be read every cycle, like a variable, position, velocity, current, etc.

**NOTE:** 3) These fields report 255 as protection so that the data will be interpreted as an error if PLC program / controller reads this and attempts to interpret as 14-byte input format. It is also helpful for the PLC program / controller to read this when the 56 byte input mode is intended because it will provide confirmation by reporting 255. A PLC program could use this as a verification check.

**NOTE:** 4) You must write 0 to these fields. If any other value is written, then the remainder of the output packet will be ignored. This is for protection in case the PLC/Controller is attempting to write in the 6 byte output format but has the 24-byte format configured.

## Command (Output) Packet Notes

The following are notes regarding the Command (Output) Packets:

- A command is issued to the SmartMotor exactly one time after the Command Code or Command Data Value changes in the output data. To issue a command:
    a. Set the Command Code to 0.
    b. Wait for Command Code Acknowledge = 0.
    c. Set the Command Data Value to the desired value.
    d. Set the Command Code to the desired command.
    e. Wait for Command Code Acknowledge = Command Code.
- For <value>, insert the Command Data Value.
- For the variables <a to zzz>:
    - <a to z> u8VarIndexSet (0-25)
    - <aa to zz> u8VarIndexSet (26-51)
    - <aaa to zzz> u8VarIndexSet (52-77)
- For <index>, insert the array index stored in u8ArrIndexSetActual.
- For <length>, insert the length stored in u8VarLenSet or u8ArrLenSet.
- Curly brackets {} indicate binary data rather than ASCII characters.
- The PROFINET interface does not interfere with the SmartMotor's EPTR command for access to EEPROM. Therefore, the user program may use the EPTR command at the same time.

## Response (Input) Packet Notes

The following are notes regarding the Command (Output) Packets:

- The requests associated with any Response Codes other than 214-225 are issued to the SmartMotor continuously (or according to the polling rate if set). When the Response Code in the output data transitions to a value in the range of 214-225, the associated request will be issued to the SmartMotor exactly one time after transition to one of those values. To issue a request for data:
    a. Set the Response Code to 0.
    b. Wait for Response Code Acknowledge = 0.
    c. Set the Response Code to the desired value.
    d. Wait for Response Code Acknowledge = Response Code read data from Response Data Value.
    e. Repeat as desired if not Response Codes 214-225.
- For <value>, insert the Response Data Value.
- For the variables <a to zzz>:
    - <a to z> u8VarIndexGet (0-25)
    - <aa to zz> u8VarIndexGet (26-51)
    - <aaa to zzz> u8VarIndexGet (52-77)
- For <index>, insert the array index stored in u8ArrIndexGetActual.
- For <length>, insert the length stored in u8VarLenGet or u8ArrIndexGet.
- Curly brackets {} indicate binary data rather than ASCII characters.
- The Response Data Value for a GET_MODE (SmartMotor RMODE) command will contain the integer code returned by the SmartMotor, which may be unexpected by users familiar with the RMODE command in older Moog Animatics products. For details on the RMODE command, see the *Moog Animatics SmartMotor™ Command Reference Guide*.
- The PROFINET interface does not use the SmartMotor's EPTR command during initialization to read startup parameters from the SmartMotor. Therefore, the user program may use EPTR command at the same time. Also, the SmartMotor variable zzz is not used by the PROFINET interface, which may be unexpected by users familiar with older Moog Animatics products.

# Alternate Communications Channel

In addition to communicating over PROFINET, commands in the SmartMotor™ programming language may be sent through an existing communications channel of the SmartMotor. For details, see the *Moog Animatics SmartMotor™ User's Guide*.

## Reserved Motor Variables

The PROFINET interface does not:

- Require the reservation of any user variables. Some older Moog Animatics products required the reservation of yyy and zzz. However, this is not the case in the PROFINET interface–these variables are freely available for the user.
- Require the reservation of any serial channels. Therefore, all other ports and associated channels are freely available to the user for the application.
- Interfere with the EPTR variable of the EEPROM command set. When PROFINET accesses the EEPROM, it is done through a private version of EPTR. Therefore, the user no longer has to monitor variable zzz for shared access. The user may access the EEPROM at any time.

  **NOTE:** EEPROM reads may still cause a user command to wait until the EEPROM is available, but there is no user interaction required.

# Command and Response Codes

This section lists the PROFINET packet command and response codes and their corresponding SmartMotor commands.

# Command Packet Codes to Motor Commands

This section provides a reference table of PROFINET command packet codes and corresponding SmartMotor commands.

Variables beginning with u8, u16 or u32 are internal to the motor's PROFINET module.

For the variables:

- <a to z> use values (0 to 25)
- <aa to zz> use values (26 to 51)
- <aaa to zzz> use values (52 to 77)

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 0, x00 | N/A | N/A | NULL / NOP - No command requested<br>Does not perform any action, this command code provided as a way to not initiate a new command, or as a value to alternate to prior to setting a new command. | |
| 1, x01 | 0, x00 | C/D* | Engage brake | BRKENG |
| 1, x01 | 1, x01 | C/D* | Use only internal brake; disable external brake | EOBK(-1) |
| 1, x01 | 2, x02 | C/D* | Direct brake to output number 8 | EOBK(8) |
| 1, x01 | 3, x03 | N/A | (Reserved) | |
| 1, x01 | 4, x04 | C/D* | Release brake | BRKRLS |
| 1, x01 | 5, x05 | C/D* | Brake while servo inactive | BRKSRV |
| 1, x01 | 6, x06 | C/D* | Brake while trajectory inactive | BRKTRJ |
| 1, x01 | 7, x07 | N/A | (Reserved) | |
| 1, x01 | 8, x08 | C/D* | Select internal encoder for servo | ENC0 |
| 1, x01 | 9, x09 | C/D* | Select external encoder for servo | ENC1 |
| 1, x01 | 10, x0A | C/D* | End user program | END |
| 1, x01 | 11, x0B | C/D* | Transfer buffered PID tuning to live values | F |
| 1, x01 | 12, x0C | C/D* | Start motion (GO) | G |
| 1, x01 | 13, x0D | N/A | (Obsolete) Use KG=0 | KGOFF |
| 1, x01 | 14, x0E | N/A | (Obsolete) Use KG=<value>, command 131 | KGON |
| 1, x01 | 15-18, x80F-x12 | N/A | (Obsolete) | |
| 1, x01 | 19, x13 | N/A | (Reserved) not implemented | |
| 1, x01 | 20-22, x14-x16 | N/A | (Obsolete) | |
| 1, x01 | 23, x17 | N/A | (Reserved) not implemented | |
| 1, x01 | 24, x18 | C/D* | Set mode follow and zero out | MF0 |
| 1, x01 | 25-27, x19-x1B | N/A | (Obsolete) | |
| 1, x01 | 28, x1C | C/D* | Initiate mode follow quadrature | MFR |
| 1, x01 | 29, x1D | C/D* | Enable position mode | MP |
| 1, x01 | 30, x1E | N/A | (Obsolete) | |
| 1, x01 | 31, x1F | C/D* | Configure step and direction, and zero out | MS0 |
| 1, x01 | 32, x20 | C/D* | Initiate mode step ratio calculation | MSR |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 1, x01 | 33, x21 | C/D* | Enable torque mode | MT |
| 1, x01 | 34, x22 | C/D* | Immediately engage MTB brake | MTB |
| 1, x01 | 35, x23 | C/D* | Enable velocity mode | MV |
| 1, x01 | 36, x24 | C/D* | Stop servoing the motor | OFF |
| 1, x01 | 37, x25 | C/D* | Divide PID sample rate by 1 | PID1 |
| 1, x01 | 38-40, x26-x28 | N/A | (Reserved) | |
| 1, x01 | 41, x29 | C/D* | Execute stored program | RUN |
| 1, x01 | 42, x2A | C/D* | End program if RUN has not been commanded yet (since power up) | RUN? |
| 1, x01 | 43, x2B | C/D* | Abruptly stop move in progress | S |
| 1, x01 | 44, x2C | C/D* | Make I/O 0 an input | EIGN(0) |
| 1, x01 | 45, x2D | N/A | (Obsolete) Use CMD_OUT(x) | |
| 1, x01 | 46, x2E | C/D* | Make I/O 1 an input | EIGN(1) |
| 1, x01 | 47, x2F | N/A | (Obsolete) Use CMD_OUT(x) | |
| 1, x01 | 48, x30 | C/D* | Make I/O 2 an input; disable right-limit function | EIGN(2) |
| 1, x01 | 49, x31 | N/A | (Obsolete) Use CMD_OUT(x) | |
| 1, x01 | 50, x32 | C/D* | Set I/O 2 to be a right-limit input | EILP |
| 1, x01 | 51, x33 | C/D* | Make I/O 3 an input; disable left-limit function | EIGN(3) |
| 1, x01 | 52, x34 | N/A | (Obsolete) Use CMD_OUT(x) | |
| 1, x01 | 53, x35 | C/D* | Set I/O 3 to be a left-limit input | EILN |
| 1, x01 | 54, x36 | C/D* | Slow motor motion to stop | X |
| 1, x01 | 55, x37 | C/D* | Total system reset | Z |
| 1, x01 | 56, x38 | C/D* | Reset overcurrent error bit | Za |
| 1, x01 | 57, x39 | C/D* | Reset serial data parity violation latch bit, i.e., clears the parity error bits in RCHN(0) and RCHN(1) | |
| 1, x01 | 58, x3A | C/D* | Reset communications buffer overflow latch bit, i.e., clears the overflow error bits in RCHN(0) and RCHN (1) | |
| 1, x01 | 59, x3B | C/D* | Not available in Class 6 PROFINET | |
| 1, x01 | 60, x3C | C/D* | Reset position error fault | Ze |
| 1, x01 | 61, x3D | C/D* | Reset serial communication framing error latch bit, i.e., clears the framing error bits in RCHN(0) and RCHN(1) | |
| 1, x01 | 62, x3E | C/D* | Reset overtemperature fault; requires temperature to fall 5 degrees below limit | Zh |
| 1, x01 | 63, x3F | C/D* | Reset historical left-limit latch bit | Zl |
| 1, x01 | 64, x40 | C/D* | Reset historical right-limit latch bit | Zr |
| 1, x01 | 65, x41 | C/D* | Reset command scan error latch bit | Zs |
| 1, x01 | 66, x42 | C/D* | Not available in Class 6 PROFINET | |
| 1, x01 | 67, x43 | C/D* | Reset encoder wraparound event latch bit | Zw |
| 1, x01 | 68, x44 | C/D* | Reset system latches to power-up state | ZS |
| 1, x01 | 69, x45 | C/D* | Disable software limits | SLD |
| 1, x01 | 70, x46 | C/D* | Enable software limits | SLE |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 1, x01 | 71, x47 | C/D* | Make I/O 6 an input; disable GO synchronization function | EIGN(6) |
| 1, x01 | 72, x48 | C/D* | Set input 6 as the GO synchronization function | EISM(6) |
| 1, x01 | 73, x49 | C/D* | Make I/O 4 an input | EIGN(4) |
| 1, x01 | 74, x4A | C/D* | Make I/O 5 an input; (Not supported in STO models) | EIGN(5) |
| 1, x01 | 75, x4B | C/D* | Arm index capture from internal encoder, rising edge | Ai(0) |
| 1, x01 | 76, x4C | C/D* | Arm index capture from internal encoder, falling edge | Aj(0) |
| 1, x01 | 77, x4D | C/D* | Arm index capture from internal encoder, rising then falling edge | Aij(0) |
| 1, x01 | 78, x4E | C/D* | Arm index capture from internal encoder, falling then rising edge | Aji(0) |
| 1, x01 | 79, x4F | C/D* | Arm index capture from external encoder, rising edge | Ai(1) |
| 1, x01 | 80, x50 | C/D* | Arm index capture from external encoder, falling edge | Aj(1) |
| 1, x01 | 81, x51 | C/D* | Arm index capture from internal encoder, rising then falling edge | Aij(1) |
| 1, x01 | 82, x52 | C/D* | Arm index capture from internal encoder, falling then rising edge | Aji(1) |
| 1, x01 | 83, x53 | N/A | (Reserved) MDT not supported in Class 6. | |
| 1, x01 | 84, x54 | C/D* | Request enhanced trapezoidal commutation mode; entered as soon as angle is satisfied | MDE |
| 1, x01 | 85, x55 | C/D* | Request sine commutation mode (voltage mode); entered as soon as angle is satisfied | MDS |
| 1, x01 | 86, x56 | C/D* | Request current-controlled sine mode; entered as soon as angle is satisfied. | MDC |
| 1, x01 | 87, x57 | C/D* | Turn on Trajectory Overshoot Braking (TOB) feature for trapezoidal mode | MDB |
| 1, x01 | 88, x58 | C/D* | Deactivate the Drive Enable input 7. Input 7 will no longer be required to enable the motor. | EIDE(-1) |
| 1, x01 | 89, x59 | C/D* | Use input 7 as the Drive Enable input signal. | EIDE(7) |
| 1, x01 | 90, x5A | C/D* | Disable output 9 as the fault output. Allow output 9 to be user controlled. | EOFT(-1) |
| 1, x01 | 91, x5B | C/D* | Use output 9 as the fault output signal. | EOFT(9) |
| 1, x01 | 92, x5C | C/D* | Make I/O 8 an input; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | EIGN(8) |
| 1, x01 | 93, x5D | C/D* | Make I/O 9 an input; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | EIGN(9) |
| 1, x01 | 94+, x5E+ | N/A | (Reserved) | |
| 2, x02 | <value> | C/D* | Set absolute position and start motor | PT=<value> G |
| 3, x03 | <value> | C/D* | Set relative position and start motor | PRT=<value> G |
| 4, x04 | <value> | C/D* | Set velocity and start motor | VT=<value> G |
| 5, x05 | <value> | C/D* | Call a subroutine | GOSUB(<value>) |
| 6, x06 | <value> | C/D* | Branch program execution to a label | GOTO(<value>) |
| 7-89, x07-x59 | N/A | N/A | (Reserved) | |
| 90, x5A | <value> | C/D* | Clear mask on user bits, word 0, status word 12 | UR(W,0,<value>) |
| 91, x5B | <value> | C/D* | Clear mask on user bits, word 1, status word 13 | UR(W,1,<value>) |
| 92, x5C | <value> | C/D* | Set mask on user bits, word 0, status word 12 | US(W,0,<value>) |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 93, x5D | <value> | C/D* | Set mask on user bits, word 1, status word 13 | US(W,1,<value>) |
| 94, x5E | <value> | C/D* | Clear specific user bit 0-31 | UR(<value>) |
| 95, x5F | <value> | C/D* | Set specific user bit 0-31 | US(<value>) |
| 96, x60 | <value> | C/D* | Set output 8 to 0 or 1, ON is 1 sourcing. Requires EOBK command to enable, see developer's guide. | OUT(8)=<value> |
| 97, x61 | N/A | N/A | (Reserved) | |
| 98, x62 | <value> | C/D* | Set output 9 to 0 or 1, ON is 1 sourcing. Requires EOFT command to enable, see developer's guide. Requires 6.0.2.41 or 6.4.2.50 or later. | OUT(9)=<value> |
| 99, x63 | N/A | N/A | (Reserved) | |
| 100, x64 | <value> | C/D* | Set acceleration | ADT=<value> |
| 101, x65 | <value> | C/D* | Set RS-232/RS-485 address | ADDR=<value> |
| 102, x66 | <value> | C/D* | Set PWM drive signal limit | AMPS=<value> |
| 103-123, x67-x7B | N/A | N/A | (Reserved) | |
| 124, x7C | <value> | C/D* | Set relative distance (position) | PRT=<value> |
| 125, x7D | <value> | C/D* | Set allowable position error | EL=<value> |
| 126, x7E | <value> | C/D* | Set special use timer. | |
| 127, x7F | N/A | N/A | (Obsolete) | |
| 128, x80 | N/A | N/A | (Reserved) | |
| 129, x81 | <value> | C/D* | PID acceleration feed forward | KA=<value> |
| 130, x82 | <value> | C/D* | PID derivative compensation | KD=<value> |
| 131, x83 | <value> | C/D* | PID gravity compensation; for limits, see the *Moog Animatics SmartMotor™ User's Guide* | KG=<value> |
| 132, x84 | <value> | C/D* | PID integral compensation | KI=<value> |
| 133, x85 | <value> | C/D* | PID integral limit | KL=<value> |
| 134, x86 | <value> | C/D* | PID proportional compensation | KP=<value> |
| 135, x87 | <value> | C/D* | PID derivative term sample rate | KS=<value> |
| 136, x88 | <value> | C/D* | PID velocity feed forward | KV=<value> |
| 137, x89 | <value> | C/D* | Mode follow with ratio divisor | MFDIV=<value> |
| 138, x8A | <value> | C/D* | Mode follow with ratio multiplier | MFMUL=<value> |
| 139, x8B | <value> | C/D* | Set origin | O=<value> |
| 140, x8C | <value> | C/D* | Shift origin | OSH(<value>) |
| 141, x8D | N/A | N/A | (Reserved) | |
| 142, x8E | <value> | C/D* | Set absolute position target | PT=<value> |
| 143-144, x8F-x90 | N/A | N/A | (Reserved) | |
| 145, x91 | <value> | C/D* | Set RS-232/RS-485 address | SADDR<value> |
| 146-147, x92-x93 | N/A | N/A | (Reserved) | |
| 148, x94 | <value> | C/D* | Assign torque value in torque mode | T=<value> |
| 149, x95 | N/A | N/A | (Reserved) | |
| 150, x96 | <value> | C/D* | Set maximum allowable temperature (high limit) | TH=<value> |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 151-162, x97-xA2 | N/A | N/A | (Reserved) | |
| 163, xA3 | <value> | C/D* | Set velocity target | VT=<value> |
| 164, xA4 | N/A | N/A | (Reserved) | |
| 165, xA5 | <value> | C/D* | Set value of negative software limit | SLN=<value> |
| 166, xA6 | <value> | C/D* | Set value of positive software limit | SLP=<value> |
| 167, xA7 | <value> | C/D* | Set digital output 4 to 0 or 1 (supported in Class 6D, S and MT2 models only, Requires firmware 6.x.2.73 or greater | OUT(4)=<value> |
| 168, xA8 | <value> | C/D* | Set digital output 5 to 0 or 1 (supported in Class 6D, S and MT2 models only. Not supported in STO models. Requires firmware 6.x.2.73 or greater | OUT(5)=<value> |
| 169, xA9 | N/A | N/A | (Reserved) | |
| 170, xAA | <value> | C/D* | Clear status word 0; bit indicated by value | Z(0,<value>) |
| 171, xAB | <value> | C/D* | Clear status word 1; bit indicated by value | Z(1,<value>) |
| 172, xAC | <value> | C/D* | Clear status word 2; bit indicated by value | Z(2,<value>) |
| 173, xAD | <value> | C/D* | Clear status word 3; bit indicated by value | Z(3,<value>) |
| 174, xAE | <value> | C/D* | Clear status word 4; bit indicated by value | Z(4,<value>) |
| 175, xAF | <value> | C/D* | Clear status word 5; bit indicated by value | Z(5,<value>) |
| 176, xB0 | <value> | C/D* | Clear status word 6; bit indicated by value | Z(6,<value>) |
| 177-199, xB1-xC7 | N/A | N/A | (Reserved) | |
| 200, C8 | <value> | C/D* | u8VarIndexSet = <value><br>u8VarIndexSetActual = <value><br>where <value> represents which variable is referred to in the next variable write operation:<br>'a' is 0, 'b' is 1, ..., 'zzz' is 77. (Range is 0-77) | |
| 201, xC9 | | N/A | (Reserved) | |
| 202, xCA | <value> | C/D* | u8VarLenSet = <value><br>Where <value> represents quantity of variables to write in the next variable write operation. Range is 0-78. | |
| 203, xCB | <value> | C/D* | u8ArrIndexSet = <value><br>u8ArrIndexSetActual = <value><br>where <value> is the array index to begin the next array write operation at.<br>The ranges of <value> are as follows depending on the type of array:<br>ab[]: 0-203<br>aw[]: 0-101<br>al[]: 0-50 | |
| 204, xCC | | N/A | (Reserved) | |
| 205, xCD | <value> | C/D* | u8ArrLenSet = <value><br>where <value> represents the quantity of array variables to write in the next array write operation.<br>The ranges of <value> are as follows depending on the type of array:<br>ab[]: 0-204<br>aw[]: 0-102<br>al[]: 0-51 | |
| 206, xCE | <value> | C/D* | u8AutoIncSet = <value><br>Enable increment of variable or array index on the next write operation. Where <value> is: 0=NO, 1=YES | |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 207, xCF | <value> | C/D* | u8VarIndexGet = <value><br>u8VarIndexGetActual = <value><br>where <value> represents which variable is referred to in the next variable read operation:<br>'a' is 0, 'b' is 1, ..., 'zzz' is 77. (Range is 0-77) | |
| 208, xD0 | | N/A | (Reserved) | |
| 209, xD1 | <value> | C/D* | u8VarLenGet = <value><br>Where <value> represents quantity of variables to read in the next variable read operation. Range is 0-78. | |
| 210, xD2 | <value> | C/D* | u8ArrIndexGet = <value><br>u8ArrIndexGetActual = <value><br>where <value> is the array index to begin the next array read operation at.<br>The ranges of <value> are as follows depending on the type of array:<br>ab[]: 0-203<br>aw[]: 0-101<br>al[]: 0-50 | |
| 211, xD3 | | N/A | (Reserved) | |
| 212, xD4 | <value> | C/D* | u8ArrLenGet = <value><br>where <value> represents the quantity of array variables to read in the next array read operation.<br>The ranges of <value> are as follows depending on the type of array:<br>ab[]: 0-204<br>aw[]: 0-102<br>al[]: 0-51 | |
| 213, xD5 | <value> | C/D* | u8AutoIncGet = <value><br>Enable increment of variable or array index on the next read operation. Where value is: 0=NO, 1=YES | |
| 214, xD6 | <value> | C/D* | Set variable <a to zzzz><br>='a'+u8VarIndexSetActual;<br>if (u8AutoIncSet) then:<br>u8VarIndexSetActual += 1 | <a to zzz>=<value> |
| 215, xD7 | <value> | C/D* | Set byte array variable<br><index>=u8ArrIndexSetActual;<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += 1 | ab[<index>]=<value> |
| 216, xD8 | <value> | C/D* | Set word array variable<br><index>=u8ArrIndexSetActual;<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += 1 | aw[<index>]=<value> |
| 217, xD9 | <value> | C/D* | Set long array variable<br><index>=u8ArrIndexSetActual;<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += 1 | al[<index>]=<value> |
| 218, xDA | <value> | C/D* | Store byte to EEPROM<br>u32EptrActual += 1<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | VST(<value byte>,1)<br>(But does not affect EPTR or variables.) |
| 219, xDB | <value> | C/D* | Store word to EEPROM<br>u32EptrActual += 2<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | VST(<value word16>,1)<br>(But does not affect EPTR or variables.) |
| 220, xDC | <value> | C/D* | Store long to EEPROM<br>u32EptrActual += 4<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | VST(<value long>,1)<br>(But does not affect EPTR or variables.) |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 221, xDD | <value> | C/D* | Set variable and store to EEPROM<br><a to z>='a'+u8VarIndexSetActual<br>u32EptrActual += 4;<br>if (u8AutoIncSet) then:<br>u8VarIndexSetActual += 1<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | <a to z>=<value><br>VST(<a to z>,1)<br>(does not affect EPTR) |
| 222, xDE | N/A | C/D* | Store variables to EEPROM<br><a to z>='a'+u8VarIndexSetActual<br><length>=u8VarLenSet<br>u32EptrActual += (<length>*4);<br>if (u8AutoIncSet) then:<br>u8VarIndexSetActual += <length><br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | VST(<a to z>, <length>)<br>(does not affect EPTR) |
| 223, xDF | N/A | C/D* | Store byte array variables to EEPROM<br><index>=u8ArrIndexSetActual<br><length>=u8ArrLenSet<br>u32EptrActual += (<length>*1);<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += <length><br>(This u32EptrActual is not the same as the program EPTR= command.) | VST(ab[<index>], <length>)<br>(does not affect EPTR) |
| 224, xE0 | N/A | C/D* | Store word array variables to EEPROM<br><index>=u8ArrIndexSetActual<br><length>=u8ArrLenSet<br>u32EptrActual += (<length>*2);<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += <length><br>(This u32EptrActual is not the same as the program EPTR= command.) | VST(aw[<index>], <length>)<br>(does not affect EPTR) |
| 225, xE1 | N/A | C/D* | Store long array variables to EEPROM<br><index>=u8ArrIndexSetActual<br><length>=u8ArrLenSet<br>u32EptrActual += (<length>*4);<br>if (u8AutoIncSet) then:<br>u8ArrIndexSetActual += <length><br>(This u32EptrActual is not the same as the program EPTR= command.) | VST(al[<index>], <length>)<br>(does not affect EPTR) |
| 226, xE2 | <value> | C/D* | Set the EEPROM address<br>u32EptrSet=<value><br>u32EptrActual=<value><br>(doesn't affect EPTR) | |
| 227-239, xE3-xEF | N/A | N/A | (Reserved) | |
| 240, xF0 | <value> | C/D* | Configure the PROFINET input packet to use an alternate data source for the 'Measured position' field (words 4,5)<br><value>:<br>0 - report actual position in encoder counts (this is the power-up default value)<br>1 - report al[0] (big-endian format)<br>2 - report af[0] (IEEE-754 32-bit single precision, big-endian format) | ETHCTL(10,x) |
| 241-254, xF1-xFE | N/A | N/A | (Reserved) | |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | *decimal, hex* | | | |
| 255, xFF | N/A | N/A | (Error)<br>Not a command. This is what the command ack will return if the command code could not be performed successfully | |

C/D* Indicates that a change of command code or a change of the command data will cause this command to occur. Values that are changed locally in a SmartMotor program will not trigger this update. In other words, the change of data must be a change relative to the previous network output data cycle from the PLC for the command to occur in the motor.

The above codes can be used in the original 3 word out, 7 word in data exchange, or in the extended 12 word out, 28 word in data exchange. The high-order byte should be set to 0 when using these in the extended packet, which has 16-bit fields for the command and reponse codes. For example: command code 124 sets PRT=. As an 8-bit hex value, 124 is x7C; as a 16-bit value, that is x007C. The endian-ness is determined by the byte-swap configuration parameter.

## Extended 16-bit command codes

Below are additional 16-bit codes. Therefore, they require the extended data format with its 16-bit fields for command and response codes.

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 256, x0100 | <value> | C/D* | Set variable a | a=<value> |
| 257, x0101 | <value> | C/D* | Set variable b | b=<value> |
| 258, x0102 | <value> | C/D* | Set variable c | c=<value> |
| 259, x0103 | <value> | C/D* | Set variable d | d=<value> |
| 260, x0104 | <value> | C/D* | Set variable e | e=<value> |
| 261, x0105 | <value> | C/D* | Set variable f | f=<value> |
| 262, x0106 | <value> | C/D* | Set variable g | g=<value> |
| 263, x0107 | <value> | C/D* | Set variable h | h=<value> |
| 264, x0108 | <value> | C/D* | Set variable i | i=<value> |
| 265, x0109 | <value> | C/D* | Set variable j | j=<value> |
| 266, x010A | <value> | C/D* | Set variable k | k=<value> |
| 267, x010B | <value> | C/D* | Set variable l | l=<value> |
| 268, x010C | <value> | C/D* | Set variable m | m=<value> |
| 269, x010D | <value> | C/D* | Set variable n | n=<value> |
| 270, x010E | <value> | C/D* | Set variable o | o=<value> |
| 271, x010F | <value> | C/D* | Set variable p | p=<value> |
| 272, x0110 | <value> | C/D* | Set variable q | q=<value> |
| 273, x0111 | <value> | C/D* | Set variable r | r=<value> |
| 274, x0112 | <value> | C/D* | Set variable s | s=<value> |
| 275, x0113 | <value> | C/D* | Set variable t | t=<value> |
| 276, x0114 | <value> | C/D* | Set variable u | u=<value> |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 277, x0115 | <value> | C/D* | Set variable v | v=<value> |
| 278, x0116 | <value> | C/D* | Set variable w | w=<value> |
| 279, x0117 | <value> | C/D* | Set variable x | x=<value> |
| 280, x0118 | <value> | C/D* | Set variable y | y=<value> |
| 281, x0119 | <value> | C/D* | Set variable z | z=<value> |
| 282, x011A | <value> | C/D* | Set variable aa | aa=<value> |
| 283, x011B | <value> | C/D* | Set variable bb | bb=<value> |
| 284, x011C | <value> | C/D* | Set variable cc | cc=<value> |
| 285, x011D | <value> | C/D* | Set variable dd | dd=<value> |
| 286, x011E | <value> | C/D* | Set variable ee | ee=<value> |
| 287, x011F | <value> | C/D* | Set variable ff | ff=<value> |
| 288, x0120 | <value> | C/D* | Set variable gg | gg=<value> |
| 289, x0121 | <value> | C/D* | Set variable hh | hh=<value> |
| 290, x0122 | <value> | C/D* | Set variable ii | ii=<value> |
| 291, x0123 | <value> | C/D* | Set variable jj | jj=<value> |
| 292, x0124 | <value> | C/D* | Set variable kk | kk=<value> |
| 293, x0125 | <value> | C/D* | Set variable ll | ll=<value> |
| 294, x0126 | <value> | C/D* | Set variable mm | mm=<value> |
| 295, x0127 | <value> | C/D* | Set variable nn | nn=<value> |
| 296, x0128 | <value> | C/D* | Set variable oo | oo=<value> |
| 297, x0129 | <value> | C/D* | Set variable pp | pp=<value> |
| 298, x012A | <value> | C/D* | Set variable qq | qq=<value> |
| 299, x012B | <value> | C/D* | Set variable rr | rr=<value> |
| 300, x012C | <value> | C/D* | Set variable ss | ss=<value> |
| 301, x012D | <value> | C/D* | Set variable tt | tt=<value> |
| 302, x012E | <value> | C/D* | Set variable uu | uu=<value> |
| 303, x012F | <value> | C/D* | Set variable vv | vv=<value> |
| 304, x0130 | <value> | C/D* | Set variable ww | ww=<value> |
| 305, x0131 | <value> | C/D* | Set variable xx | xx=<value> |
| 306, x0132 | <value> | C/D* | Set variable yy | yy=<value> |
| 307, x0133 | <value> | C/D* | Set variable zz | zz=<value> |
| 308, x0134 | <value> | C/D* | Set variable aaa | aaa=<value> |
| 309, x0135 | <value> | C/D* | Set variable bbb | bbb=<value> |
| 310, x0136 | <value> | C/D* | Set variable ccc | ccc=<value> |
| 311, x0137 | <value> | C/D* | Set variable ddd | ddd=<value> |
| 312, x0138 | <value> | C/D* | Set variable eee | eee=<value> |
| 313, x0139 | <value> | C/D* | Set variable fff | fff=<value> |
| 314, x013A | <value> | C/D* | Set variable ggg | ggg=<value> |
| 315, x013B | <value> | C/D* | Set variable hhh | hhh=<value> |
| 316, x013C | <value> | C/D* | Set variable iii | iii=<value> |

| Command Code | Command Data Value | Event for update | Command Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 317, x013D | <value> | C/D* | Set variable jjj | jjj=<value> |
| 318, x013E | <value> | C/D* | Set variable kkk | kkk=<value> |
| 319, x013F | <value> | C/D* | Set variable lll | lll=<value> |
| 320, x0140 | <value> | C/D* | Set variable mmm | mmm=<value> |
| 321, x0141 | <value> | C/D* | Set variable nnn | nnn=<value> |
| 322, x0142 | <value> | C/D* | Set variable ooo | ooo=<value> |
| 323, x0143 | <value> | C/D* | Set variable ppp | ppp=<value> |
| 324, x0144 | <value> | C/D* | Set variable qqq | qqq=<value> |
| 325, x0145 | <value> | C/D* | Set variable rrr | rrr=<value> |
| 326, x0146 | <value> | C/D* | Set variable sss | sss=<value> |
| 327, x0147 | <value> | C/D* | Set variable ttt | ttt=<value> |
| 328, x0148 | <value> | C/D* | Set variable uuu | uuu=<value> |
| 329, x0149 | <value> | C/D* | Set variable vvv | vvv=<value> |
| 330, x014A | <value> | C/D* | Set variable www | www=<value> |
| 331, x014B | <value> | C/D* | Set variable xxx | xxx=<value> |
| 332, x014C | <value> | C/D* | Set variable yyy | yyy=<value> |
| 333, x014D | <value> | C/D* | Set variable zzz | zzz=<value> |
| 334 - 511, x014E - x01FF | N/A | N/A | (Reserved) | |
| 512, x0200 | <value> | C/D* | Set float 0 (32-bit IEEE) | af[0]=<value> |
| 513, x0201 | <value> | C/D* | Set float 1 (32-bit IEEE) | af[1]=<value> |
| 514, x0202 | <value> | C/D* | Set float 2 (32-bit IEEE) | af[2]=<value> |
| 515, x0203 | <value> | C/D* | Set float 3 (32-bit IEEE) | af[3]=<value> |
| 516, x0204 | <value> | C/D* | Set float 4 (32-bit IEEE) | af[4]=<value> |
| 517, x0205 | <value> | C/D* | Set float 5 (32-bit IEEE) | af[5]=<value> |
| 518, x0206 | <value> | C/D* | Set float 6 (32-bit IEEE) | af[6]=<value> |
| 519, x0207 | <value> | C/D* | Set float 7 (32-bit IEEE) | af[7]=<value> |
| 520 - 767, x0208 - x02FF | N/A | N/A | (Reserved) | |
| 768, x0300 | <value> | C/D* | Set long array element 0 | al[0]=<value> |
| 769, x0301 | <value> | C/D* | Set long array element 1 | al[1]=<value> |
| ... | | | | ... |
| 818, x0332 | <value> | C/D* | Set long array element 50 | al[50]=<value> |
| 819 - 65535, x0333 - xFFFF | N/A | N/A | (Reserved) | |

C/D* Indicates that a change of command code or a change of the command data will cause this variable to be written. Variables that are changed locally in a SmartMotor program will not trigger this update. In other words, to set a new value for the variable, the change of data must be a change relative to the previous network output data cycle (not the current state of the variable in the motor) from the PLC.

# Response Packet Codes to Motor Commands

This section provides a reference table of PROFINET response packet codes and corresponding SmartMotor commands.

Variables beginning with u8, u16 or u32 are internal to the motor's PROFINET module.

For the variables:

- <a to z> use values (0 to 25)
- <aa to zz> use values (26 to 51)
- <aaa to zzz> use values (52 to 77)

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 0, x00 | N/A | 0 | NULL / NOP - No response requested | |
| 1-95, x01-x5f | N/A | 0 | (Reserved) | |
| 96, x60 | cyclic | <value>= | digital I/O number 4 | RIN(4) |
| 97, x61 | N/A | 0 | (Reserved) | |
| 98, x62 | cyclic | <value>= | digital I/O number 5 | RIN(5) |
| 99, x63 | N/A | 0 | (Reserved) | |
| 100, x64 | cyclic | <value>= | acceleration target | RAT |
| 101, x65 | cyclic | <value>= | SmartMotor serial address | RADDR |
| 102, x66 | cyclic | <value>= | assigned PWM limit | RAMPS |
| 103, x67 | cyclic | <value>= | overcurrent status | RBa |
| 104, x68 | N/A | 0 | (Obsolete) | |
| 105, x69 | cyclic | <value>= | serial communications error bit | |
| 106, x6A | N/A | 0 | (Obsolete) | |
| 107, x6B | cyclic | <value>= | position error status | RBe |
| 108, x6C | N/A | 0 | (Obsolete) | |
| 109, x6D | cyclic | <value>= | overheat status | RBh |
| 110, x6E | cyclic | <value>= | index status | RBi |
| 111, x6F | cyclic | <value>= | program checksum error | RBk |
| 112, x70 | cyclic | <value>= | historical left limit status | RBl |
| 113, x71 | cyclic | <value>= | negative limit status | RBm |
| 114, x72 | cyclic | <value>= | motor off status | RBo |
| 115, x73 | cyclic | <value>= | positive limit status | RBp |
| 116, x74 | cyclic | <value>= | historical right limit status | RBr |
| 117, x75 | cyclic | <value>= | program scan status | RBs |
| 118, x76 | cyclic | <value>= | trajectory status | RBt |
| 119, x77 | N/A | 0 | (Obsolete) | |
| 120, x78 | cyclic | <value>= | wrapped encoder position | RBw |
| 121, x79 | cyclic | <value>= | hardware index input level | RBx |
| 122, x7A | cyclic | <value>= | millisecond clock | RCLK |
| 123, x7B | cyclic | <value>= | secondary counter | RCTR(1) |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 124, x7C | cyclic | <value>= | buffered move distance value | RPRC |
| 125, x7D | cyclic | <value>= | buffered maximum position error | REL |
| 126, x7E | cyclic | <value>= | special use timer | |
| 127, x7F | cyclic | 0 | (Obsolete) | |
| 128, x80 | cyclic | <value>= | index position captured from recent Ai(0) command - object 1, data value 75 | RI(0) |
| 129, x81 | cyclic | <value>= | buffered acceleration feed forward coefficient | RKA |
| 130, x82 | cyclic | <value>= | buffered derivative coefficient | RKD |
| 131, x83 | cyclic | <value>= | buffered gravity coefficient | RKG |
| 132, x84 | cyclic | <value>= | buffered integral coefficient | RKI |
| 133, x85 | cyclic | <value>= | buffered integral limit | RKL |
| 134, x86 | cyclic | <value>= | buffered proportional coefficient | RKP |
| 135, x87 | cyclic | <value>= | buffered sampling interval | RKS |
| 136, x88 | cyclic | <value>= | buffered velocity feed forward coefficient | RKV |
| 137, x89 | cyclic | <value>= | follow mode divisor | RMFDIV |
| 138, x8A | cyclic | <value>= | follow mode multiplier | RMFMUL |
| 139, x8B | N/A | 0 | (Reserved) | |
| 140, x8C | cyclic | <value>= | current mode of operation | RMODE |
| 141, x8D | cyclic | <value>= | present position | RPA |
| 142, x8E | cyclic | <value>= | buffered position setpoint | RPT |
| 143, x8F | cyclic | <value>= | present position error | REA |
| 144, x90 | N/A | 0 | (Reserved) | |
| 145, x91 | cyclic | <value>= | motor RMS current<br>Requires firmware 6.0.2.41 or 6.4.2.50 or later | RUIA |
| 146, x92 | cyclic | <value>= | servo bus voltage<br>Requires firmware 6.0.2.41 or 6.4.2.50 or later | RUJA |
| 147, x93 | cyclic | <value>= | Control Voltage; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | RINA(A,11) |
| 148, x94 | cyclic | <value>= | current requested torque | RT |
| 149, x95 | cyclic | <value>= | temperature | RTEMP |
| 150, x96 | cyclic | <value>= | temperature shutdown limit | RTH |
| 151, x97 | cyclic | <value>= | current algorithm THD time | RTHD |
| 152, x98 | cyclic | <value>= | digital I/O number 0 | RIN(0) |
| 153, x99 | cyclic | <value>= | analog input number 0 | RINA(A,0) |
| 154, x9A | cyclic | <value>= | digital I/O number 1 | RIN(1) |
| 155, x9B | cyclic | <value>= | analog input number 1 | RINA(A,1) |
| 156, x9C | cyclic | <value>= | digital I/O number 2 | RIN(2) |
| 157, x9D | N/A | 0 | (Reserved) | |
| 158, x9E | cyclic | <value>= | digital I/O number 3 | RIN(3) |
| 159, x9F | N/A | 0 | (Reserved) | |
| 160, xA0 | cyclic | <value>= | digital I/O number 6 | RIN(6) |
| 161, xA1 | N/A | 0 | (Reserved) | |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 162, xA2 | cyclic | <value>= | velocity | RVA |
| 163, xA3 | cyclic | <value>= | buffered maximum velocity | RVT |
| 164, xA4 | cyclic | <value>= | legacy status word | (n/a) |
| 165, xA5 | cyclic | <value>= | value of negative software limit | RSLN |
| 166, xA6 | cyclic | <value>= | value of positive software limit | RSLP |
| 167, xA7 | N/A | 0 | (Reserved) | |
| 168, xA8 | cyclic | <value>= | Inputs 0-7, 16-bit value, right justified | RIN(W,0) |
| 169, xA9 | N/A | 0 | (Reserved) | |
| 170, xAA | cyclic | <value>= | status word 0 | RW(0) |
| 171, xAB | cyclic | <value>= | status word 1 | RW(1) |
| 172, xAC | cyclic | <value>= | status word 2 | RW(2) |
| 173, xAD | cyclic | <value>= | status word 3 | RW(3) |
| 174, xAE | cyclic | <value>= | status word 4 | RW(4) |
| 175, xAF | cyclic | <value>= | status word 5 | RW(5) |
| 176, xB0 | cyclic | <value>= | status word 6 | RW(6) |
| 177, xB1 | cyclic | <value>= | status word 7 | RW(7) |
| 178, xB2 | cyclic | <value>= | status word 8 | RW(8) |
| 179, xB3 | cyclic | <value>= | status word 9 | RW(9) |
| 180-181, xB4-xB5 | N/A | 0 | (Reserved) | |
| 182, xB6 | cyclic | <value>= | user bits 0-15 (status word 12) | RW(12) |
| 183, xB7 | cyclic | <value>= | user bits 16-31 (status word 13) | RW(13) |
| 184-185, xB8-xB9 | N/A | 0 | (Reserved) | |
| 186, xBA | cyclic | <value>= | I/O 0-7 (status word 16) | RW(16) |
| 187-194, xBB-xC2 | N/A | 0 | (Reserved) | |
| 195, xC3 | cyclic | <value>= | digital I/O number 7: Requires firmware 6.x.2.73 or greater | RIN(7) |
| 196, xC4 | cyclic | <value>= | digital I/O number 8; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | RIN(8) |
| 197, xC5 | cyclic | <value>= | digital I/O number 9; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | RIN(9) |
| 198, xC6 | cyclic | <value>= | digital I/O number 14; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | RIN(14) |
| 199, xC7 | cyclic | <value>= | digital I/O number 15; (Supported in Class 6 D and S models only) Requires firmware 6.x.2.73 or greater | RIN(15) |
| 200, xC8 | cyclic | <value>= | u8VarIndexSet | |
| 201, xC9 | cyclic | <value>= | u8VarIndexSetActual | |
| 202, xCA | cyclic | <value>= | u8VarLenSet | |
| 203, xCB | cyclic | <value>= | u8ArrIndexSet | |
| 204, xCC | cyclic | <value>= | u8ArrIndexSetActual | |
| 205, xCD | cyclic | <value>= | u8ArrLenSet | |
| 206, xCE | cyclic | <value>= | u8AutoIncSet | |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 207, xCF | cyclic | <value>= | u8VarIndexGet | |
| 208, xD0 | cyclic | <value>= | u8VarIndexGetActual | |
| 209, xD1 | cyclic | <value>= | u8VarLenGet | |
| 210, xD2 | cyclic | <value>= | u8ArrIndexGet | |
| 211, xD3 | cyclic | <value>= | u8ArrIndexGetActual | |
| 212, xD4 | cyclic | <value>= | u8ArrLenGet | |
| 213, xD5 | cyclic | <value>= | u8AutoIncGet | |
| 214, xD6 Response 0 only | RC* | <value> | Get 1 variable: a to zzz<br><var a-zzz> is: 'a'+u8VarIndexGetActual<br><value> = value of <var a-zzz><br>if (u8AutoIncGet) then:<br>u8VarIndexGetActual += 1 | R<var a to zzz> |
| 215, xD7 Response 0 only | RC* | <value> | Get 1 byte array variable<br><index>=u8ArrIndexGetActual<br><value>=ab[<index>]<br>if (u8AutoIncGet) then:<br>u8ArrIndexGetActual += 1 | Rab[<index>] |
| 216, xD8 Response 0 only | RC* | <value> | Get 1 word array variable<br><index>=u8ArrIndexGetActual<br><value>=aw[<index>]<br>if (u8AutoIncGet) then:<br>u8ArrIndexGetActual += 1 | Raw[<index>] |
| 217, xD9 Response 0 only | RC* | <value> | Get 1 long array variable<br><index>=u8ArrIndexGetActual<br><value>=al[<index>]<br>if (u8AutoIncGet) then:<br>u8ArrIndexGetActual += 1 | Ral[<index>] |
| 218, xDA Response 0 only | RC* | <value> | Get **byte** from EEPROM<br><value>=EE byte at u32EptrActual<br>u32EptrActual += 1<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | (VLD, but does not affect EPTR or variables.) |
| 219, xDB Response 0 only | RC* | <value> | Get **word** from EEPROM<br><value>=EE 2 bytes at u32EptrActual<br>u32EptrActual += 2<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | (VLD, but does not affect EPTR or variables.) |
| 220, xDC Response 0 only | RC* | <value> | Get **long** from EEPROM<br><value>=EE 4 bytes at u32EptrActual<br>u32EptrActual += 4<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | (VLD, but does not affect EPTR or variables.) |
| 221, xDD Response 0 only | RC* | <value> | Get 4 bytes from EEPROM, store in 1 var a to zzz, and return that value.<br><var a-zzz> is: 'a'+u8VarIndexGetActual<br><value> = value of <var a-zzz> after VLD.<br>u32EptrActual += 4<br>if (u8AutoIncGet) then:<br>u8VarIndexGetActual += 1<br>NOTE: This u32EptrActual is not the same as the program EPTR= command. | VLD(<var a to zzz>,1)<br>R<var a to zzz><br><br>(does not affect EPTR) |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 222, xDE Response 0 only | RC* | 0 | Load data from EEPROM into multiple variables <var a-zzz> is: 'a'+u8VarIndexGetActual <length>=u8VarLenGet u32EptrActual += (<length>*4); if (u8AutoIncGet) then: u8VarIndexGetActual += <length> NOTE: This u32EptrActual is not the same as the program EPTR= command. | VLD(<var a to zzz>,<length>)<br><br>(does not affect EPTR) |
| 223, xDF Response 0 only | RC* | 0 | Load data from EEPROM into multiple byte array variables <index>=u8ArrIndexGetActual <length>=u8ArrLenGet u32EptrActual += (<length>*1); if (u8AutoIncGet) then: u8ArrIndexGetActual += <length> NOTE: This u32EptrActual is not the same as the program EPTR= command. | VLD(ab[<index>], <length>)<br><br>(does not affect EPTR) |
| 224, xE0 Response 0 only | RC* | 0 | Load data from EEPROM into multiple word array variables <index>=u8ArrIndexGetActual <length>=u8ArrLenGet u32EptrActual += (<length>*2); if (u8AutoIncGet) then: u8ArrIndexGetActual += <length> NOTE: This u32EptrActual is not the same as the program EPTR= command. | VLD(aw[<index>], <length>)<br><br>(does not affect EPTR) |
| 225, xE1 Response 0 only | RC* | 0 | Load data from EEPROM into multiple long array variables <index>=u8ArrIndexGetActual <length>=u8ArrLenGet u32EptrActual += (<length>*4); if (u8AutoIncGet) then: u8ArrIndexGetActual += <length> NOTE: This u32EptrActual is not the same as the program EPTR= command. | VLD(al[<index>], <length>)<br><br>(does not affect EPTR) |
| 226, xE2 | cyclic | <value>= | u32EptrSet (last set EEPROM address) NOTE: This u32EptrActual is not the same as the program EPTR= command. | |
| 227, xE3 | cyclic | <value>= | u32EptrActual (actual EEPROM address currently in PROFINET interface) NOTE: This u32EptrActual is not the same as the program EPTR= command. | |
| 228, xE4 | cyclic | <value>= | u16NetLostLabel (initialized to the value of u16NetLostLabelDefault during power-up); see ETHCTL(...) | |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 229, xE5 | cyclic | <value>= | u8NetLostAction (initialized to the value of u8NetLostActionDefault during power-up)<br><br>See ETHCTL(...) | On loss of communication with PROFINET host, command is based on <value>:<br><br>0=IGNORE (No Command), 1=OFF (Motor Off),<br>2=X (Soft Stop),<br>3=S (Immediate Stop),<br>4=GOSUB,<br>5=GOTO |
| 230-234, xE6-xEA | N/A | 0 | (Reserved) | |
| 235, xEB | cyclic | <value>= | encoder resolution | RRES |
| 236, xEC | cyclic | <value>= | firmware version | RFW |
| 237, xED | N/A | 0 | (Obsolete) | |
| 238, xEE | cyclic | <value>= | sample period as RSP command reports it (microseconds * 100), so 8kHz reports as 12500. | RSP |
| 239-254, xF0-xFE | N/A | 0 | (Reserved) | |
| 255, xFF | N/A | 0 | (Reserved) | |

RC* indicates that a change of response request code is required to begin or repeat this event. For repeated events, that means that the code should be changed to 0, then back to the desired code per event. If using the extended packet which allows for multiple requests, then only the first request area (0) is allowed to make this request. The reason is that the global state of the motor is affected and multiple requests concurrently would be a problem.

The above codes can be used in the original 3 word out, 7 word in data exchange, or in the extended 12 word out, 28 word in data exchange. The high-order byte should be set to 0 when using these in the extended packet, which has 16-bit fields for the command and reponse codes. For example: reponse code 122 returns CLK (clock). As an 8-bit hex value, 122 is x7A; as a 16-bit value, that is x007A. The endianness is determined by the byte-swap configuration parameter.

## Extended 16-bit response codes

Below are additional 16-bit codes. Therefore, they require the extended data format with its 16-bit fields for command and response codes.

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 256, x0100 | cyclic | <value>= | variable a | Ra |
| 257, x0101 | cyclic | <value>= | variable b | Rb |
| 258, x0102 | cyclic | <value>= | variable c | Rc |
| 259, x0103 | cyclic | <value>= | variable d | Rd |
| 260, x0104 | cyclic | <value>= | variable e | Re |
| 261, x0105 | cyclic | <value>= | variable f | Rf |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 262, x0106 | cyclic | <value>= | variable g | Rg |
| 263, x0107 | cyclic | <value>= | variable h | Rh |
| 264, x0108 | cyclic | <value>= | variable i | Ri |
| 265, x0109 | cyclic | <value>= | variable j | Rj |
| 266, x010A | cyclic | <value>= | variable k | Rk |
| 267, x010B | cyclic | <value>= | variable l | Rl |
| 268, x010C | cyclic | <value>= | variable m | Rm |
| 269, x010D | cyclic | <value>= | variable n | Rn |
| 270, x010E | cyclic | <value>= | variable o | Ro |
| 271, x010F | cyclic | <value>= | variable p | Rp |
| 272, x0110 | cyclic | <value>= | variable q | Rq |
| 273, x0111 | cyclic | <value>= | variable r | Rr |
| 274, x0112 | cyclic | <value>= | variable s | Rs |
| 275, x0113 | cyclic | <value>= | variable t | Rt |
| 276, x0114 | cyclic | <value>= | variable u | Ru |
| 277, x0115 | cyclic | <value>= | variable v | Rv |
| 278, x0116 | cyclic | <value>= | variable w | Rw |
| 279, x0117 | cyclic | <value>= | variable x | Rx |
| 280, x0118 | cyclic | <value>= | variable y | Ry |
| 281, x0119 | cyclic | <value>= | variable z | Rz |
| 282, x011A | cyclic | <value>= | variable aa | Raa |
| 283, x011B | cyclic | <value>= | variable bb | Rbb |
| 284, x011C | cyclic | <value>= | variable cc | Rcc |
| 285, x011D | cyclic | <value>= | variable dd | Rdd |
| 286, x011E | cyclic | <value>= | variable ee | Ree |
| 287, x011F | cyclic | <value>= | variable ff | Rff |
| 288, x0120 | cyclic | <value>= | variable gg | Rgg |
| 289, x0121 | cyclic | <value>= | variable hh | Rhh |
| 290, x0122 | cyclic | <value>= | variable ii | Rii |
| 291, x0123 | cyclic | <value>= | variable jj | Rjj |
| 292, x0124 | cyclic | <value>= | variable kk | Rkk |
| 293, x0125 | cyclic | <value>= | variable ll | Rll |
| 294, x0126 | cyclic | <value>= | variable mm | Rmm |
| 295, x0127 | cyclic | <value>= | variable nn | Rnn |
| 296, x0128 | cyclic | <value>= | variable oo | Roo |
| 297, x0129 | cyclic | <value>= | variable pp | Rpp |
| 298, x012A | cyclic | <value>= | variable qq | Rqq |
| 299, x012B | cyclic | <value>= | variable rr | Rrr |
| 300, x012C | cyclic | <value>= | variable ss | Rss |
| 301, x012D | cyclic | <value>= | variable tt | Rtt |
| 302, x012E | cyclic | <value>= | variable uu | Ruu |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 303, x012F | cyclic | <value>= | variable vv | Rvv |
| 304, x0130 | cyclic | <value>= | variable ww | Rww |
| 305, x0131 | cyclic | <value>= | variable xx | Rxx |
| 306, x0132 | cyclic | <value>= | variable yy | Ryy |
| 307, x0133 | cyclic | <value>= | variable zz | Rzz |
| 308, x0134 | cyclic | <value>= | variable aaa | Raaa |
| 309, x0135 | cyclic | <value>= | variable bbb | Rbbb |
| 310, x0136 | cyclic | <value>= | variable ccc | Rccc |
| 311, x0137 | cyclic | <value>= | variable ddd | Rddd |
| 312, x0138 | cyclic | <value>= | variable eee | Reee |
| 313, x0139 | cyclic | <value>= | variable fff | Rfff |
| 314, x013A | cyclic | <value>= | variable ggg | Rggg |
| 315, x013B | cyclic | <value>= | variable hhh | Rhhh |
| 316, x013C | cyclic | <value>= | variable iii | Riii |
| 317, x013D | cyclic | <value>= | variable jjj | Rjjj |
| 318, x013E | cyclic | <value>= | variable kkk | Rkkk |
| 319, x013F | cyclic | <value>= | variable lll | Rlll |
| 320, x0140 | cyclic | <value>= | variable mmm | Rmmm |
| 321, x0141 | cyclic | <value>= | variable nnn | Rnnn |
| 322, x0142 | cyclic | <value>= | variable ooo | Rooo |
| 323, x0143 | cyclic | <value>= | variable ppp | Rppp |
| 324, x0144 | cyclic | <value>= | variable qqq | Rqqq |
| 325, x0145 | cyclic | <value>= | variable rrr | Rrrr |
| 326, x0146 | cyclic | <value>= | variable sss | Rsss |
| 327, x0147 | cyclic | <value>= | variable ttt | Rttt |
| 328, x0148 | cyclic | <value>= | variable uuu | Ruuu |
| 329, x0149 | cyclic | <value>= | variable vvv | Rvvv |
| 330, x014A | cyclic | <value>= | variable www | Rwww |
| 331, x014B | cyclic | <value>= | variable xxx | Rxxx |
| 332, x014C | cyclic | <value>= | variable yyy | Ryyy |
| 333, x014D | cyclic | <value>= | variable zzz | Rzzz |
| 334 - 511, x014E - x01FF | N/A | 0 | (Reserved) | |
| 512, x0200 | cyclic | <value>= | float 0 (32-bit IEEE) | Raf[0] |
| 513, x0201 | cyclic | <value>= | float 1 (32-bit IEEE) | Raf[1] |
| 514, x0202 | cyclic | <value>= | float 2 (32-bit IEEE) | Raf[2] |
| 515, x0203 | cyclic | <value>= | float 3 (32-bit IEEE) | Raf[3] |
| 516, x0204 | cyclic | <value>= | float 4 (32-bit IEEE) | Raf[4] |
| 517, x0205 | cyclic | <value>= | float 5 (32-bit IEEE) | Raf[5] |
| 518, x0206 | cyclic | <value>= | float 6 (32-bit IEEE) | Raf[6] |
| 519, x0207 | cyclic | <value>= | float 7 (32-bit IEEE) | Raf[7] |

| Response Code | Event for update | Response Data Value | Response Description | Smart Motor Command(s) |
|---|---|---|---|---|
| *decimal, hex* | | | | |
| 520 - 767, x0208 - x02FF | N/A | 0 | (Reserved) | |
| 768, x0300 | cyclic | <value>= | long array element 0 | Ral[0] |
| 769, x0301 | cyclic | <value>= | long array element 1 | Ral[1] |
| ... | cyclic | <value>= | long array element ... | Ral[...] |
| 818, x0332 | cyclic | <value>= | long array element 50 | Ral[50] |
| 819 - 65535, x0333 - xFFFF | cyclic | 0 | (Reserved) | |

# Troubleshooting

The following table provides troubleshooting information for solving SmartMotor problems that may be encountered when using PROFINET. For additional support resources, see the Moog Animatics Support page at:

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

| Issue | Cause | Solution |
|---|---|---|
| **PROFINET Communication Issues** | | |
| **NOTE:** Station Name, IP Address, Subnet Mask, and Gateway must be correct at the PROFINET I/O controller. | | |
| No PROFINET connection. | Motor not powered. | Check Drive Status LED. If LED is not lit, check wiring. |
| | Disconnected or miswired connector, or broken wiring between follower and controller. | Check that connectors are correctly wired and connected to motor. For details, see Motor Connectors and Pinouts on page 1. |
| | Motor nonvolatile settings. | Check that motor PROFINET Station name is set, and that all motors have been programmed with a unique station name. |
| | Wrong type of cable. | Check that cable is a PROFINET cable. For details, see Cables and Diagram on page 1. |
| | Wrong GSDML file. | Verify that the correct GSDML file was used to configure the controller and connect the follower motor as part of the PROFINET network. |
| Command code Ack and/or Response code Ack is returning as 255 | Unknown command or response code | Check if command/response code is supported in this version of firmware |
| | Mismatch of the original vs. extended packet size | Verify that the controller/PLC has the correct input/output data size set. Not all possible combinations are supported. Both input and output size must be the original (14 bytes in and 6 bytes out), or both must be the extended size (56 bytes in and 24 bytes out.) |
| | Byte order of command / response code or data is wrong. | Check that the byte-order parameter is set as intended. |
| | Request in the incorrect response-request area. | Certain responses (response codes 214-225) are only allowed in the first response request section (0) of the extended packet and not in responses 1-7. |
| | Value out of range | Check the data value, or related commands such as EE address or variable index for appropriate range of values. |

| Issue | Cause | Solution |
|---|---|---|
| **Other Communication and Control Issues** | | |
| 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 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. |
| After power reset, motor stops communicating over USB or serial port, requires re-detection. | Motor does not have its address set in the user program. NOTE: Serial addresses are lost when motor power is off or reset. | Use the SADDR or ADDR= command within the program to set the motor address. |
| 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. |
| 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 the default firmware version closest to the motor firmware version. In the SMI software, view the motor firmware version by right-clicking the motor and selecting Properties. |
| | Unsupported commands used in program. | Check the unrecognized commands against those listed in the section "Commands Not Currently Supported" in the *Class 6Moog Animatics SmartMotor™ User's Guide* |

NOTES

# 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 Animatics Class 6 SmartMotor™ PROFINET Guide, Rev. H
SC80100007-001