SMART Motor™

PROFINET®

GUIDE

Class 6 SmartMotor™ Technology

MOOG ANIMATICS

Copyright Notice

Moog Animatics Class 6 SmartMotor™ PROFINET Guide, Rev. A, 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: techwriter@moog.animatics.com. Thank you in advance for your contribution.

Contact Us:

Moog Inc., Animatics
1421 McCarthy Boulevard
Milpitas, CA 95035
USA

Tel: 1 (408) 965-3320
Fax: 1 (408) 965-3319
Support: 1 (888) 356-0357

www.animatics.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 SmartMotor slave with a PLC or other PROFINET master. 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 6.0.1.x.

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 51. Also, see User Program Commands on page 43.

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.

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 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 52. 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 slave 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 master

and to connect to the slave 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 10.

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 master or other PROFINET master
- Moog Animatics PROFINET cable, or equivalent, to connect the PLC to the SmartMotor's industrial Ethernet port (for details, see PROFINET Motor Connectors and Pinouts on page 18)

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 at:

  http://www.animatics.com/support/download-center.html

- Moog Animatics PROFINET GSDML file, which is available on the Moog Animatics website at:

  http://www.animatics.com/support/download-center.html

  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 the following safety symbols:

WARNING: This symbol indicates a potentially non-lethal mechanical hazard, where failure to follow the instructions could result in serious injury to the operator or major damage to the equipment.

CAUTION: This symbol indicates a potential minor hazard, where failure to follow 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, the following 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 locale where the machine is being installed and operated. For more details, see Machine Safety on page 12.

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 *Moog Animatics Product Catalog*, 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 locale 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 following 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 locale 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 file "109_Controls, Warnings and Cautions.pdf" located at:

http://www.animatics.com/support/moog-animatics-catalog.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://www.ul.com/global/eng/pages/solutions/standards/accessstandards/catalogofstandards/

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/enterprise/policies/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 the following list:

- *Moog Animatics SmartMotor™ User's Guide*

  http://www.animatics.com/support/download-center.html

- *Moog Animatics SmartMotor™ Command Reference Guide*

  http://www.animatics.com/support/download-center.html

- *SmartMotor™ Product Certificate of Conformance*

  http://www.animatics.com/download/Animatics_SmartMotor_Servida_Class_5_Declaration_of_Conformity_CE_Rev_1.pdf

- *SmartMotor™ UL Certification*

  http://www.animatics.com/download/MA_UL_online_listing.pdf

- *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)

  http://www.animatics.com/support/download-center.html

- *Moog Animatics Product Catalog*

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

Additional Resources

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

- General company information:

  http://www.animatics.com

- Product information:

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

- Product support (Downloads, How To videos, Forums, Knowledge Base, and FAQs):

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

- Sales and distributor information:

  http://www.animatics.com/sales-offices.html

- Application ideas (including videos and sample programs):

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

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

- PROFIBUS and PROFINET International (PI) website:

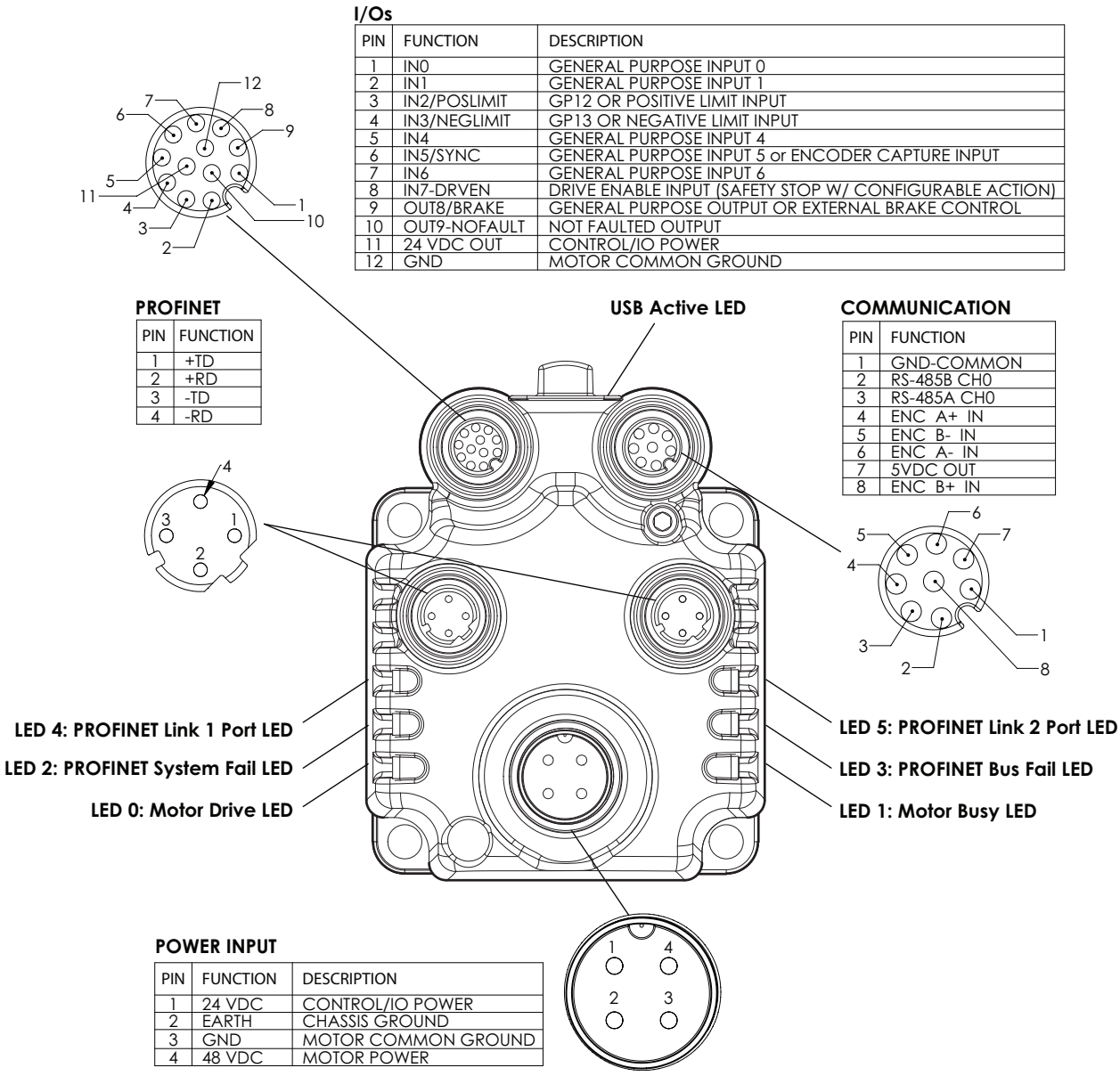
  http://www.profibus.com/

PROFINET Motor Pinouts, Connections and Status LEDs

The following sections describe the motor pinouts, system connections and the status LEDs.

PROFINET Motor Connectors and Pinouts

The following figure provides an overview of the PROFINET connectors and pinouts available on the Class 6 SmartMotors.



Cables and Diagram

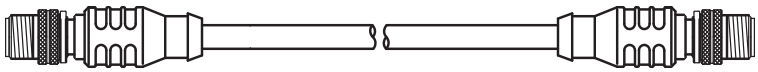
This section provides information on Moog Animatics industrial Ethernet cables and a PROFINET system cable diagram.

Moog Animatics Industrial Ethernet Cables

The following cables are available from Moog Animatics.

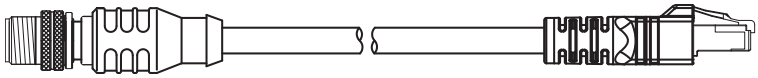
M-style to M-style Ethernet Cable

This cable has M12 male threaded connectors at both ends. It is available in 1, 3, 5 and 10 meter lengths. For the standard cable, use part number CBLIP-ETH-MM-xM, where "x" denotes the cable length. A right-angle version is also available; use part number CBLIP-ETH-MM-xMRA.



M-style to RJ45 Ethernet Cable

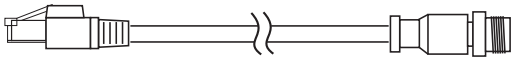
This cable has an M12 male threaded connector at one end, and an RJ45 male connector at the opposite end. It is available in 1, 3, 5 and 10 meter lengths. For the standard cable, use part number CBLIP-ETH-MR-xM, where "x" denotes the cable length. A right-angle version is also available; use part number CBLIP-ETH-MR-xMRA.



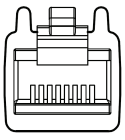
Ethernet Cable Schematic

The following figure provides details for creating a custom industrial Ethernet shielded cable.

NOTE: The motor end of the cable requires an industrial Ethernet connector.

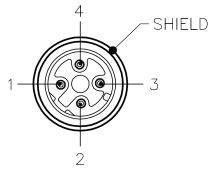


RJ45 Connector
(EtherCAT master end of cable)



PIN	DESCRIPTION
1	+TX
2	-TX
3	+RX
4	No Connection
5	No Connection
6	-RX
7	No Connection
8	No Connection
Shield tied to RJ45 connector	

Industrial Ethernet Connector
(Motor end of cable)



PIN	DESCRIPTION
1	+TX
2	+RX
3	-TX
4	-RX
Shield tied to motor housing	

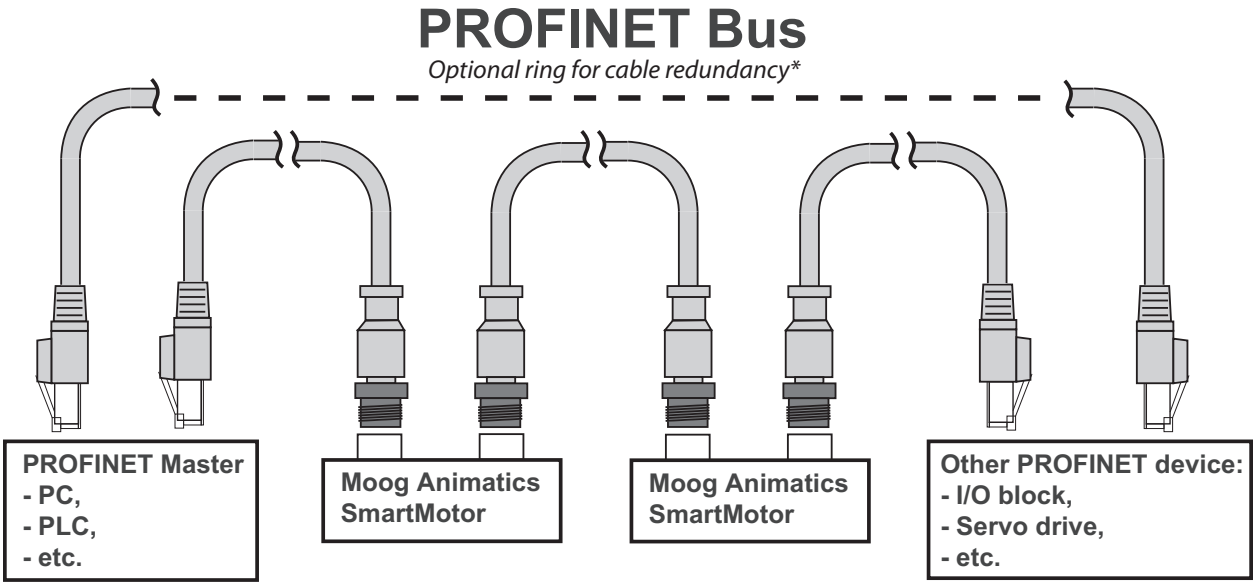
PROFINET Cable Diagram

PROFINET can support line, tree or star device-connection topology. The supported network topology and maximum number of devices depends on the selected PROFINET mode and network class. For example, higher performing modes, like PROFINET IRT, require specialized equipment. For PROFINET network design and installation details, see the information available at:

http://www.profinet.com

CAUTION: To minimize the possibility of electromagnetic interference (EMI), all connections should use *shielded* Ethernet Category 5 (Cat 5), or better, cables.

The following diagram shows an example PROFINET network with the SmartMotors daisy chained to the master device. An optional "ring" configuration can be created if it is supported by the selected PROFINET mode and network devices.



**Ring configuration requires specific PROFINET modes and supporting devices*

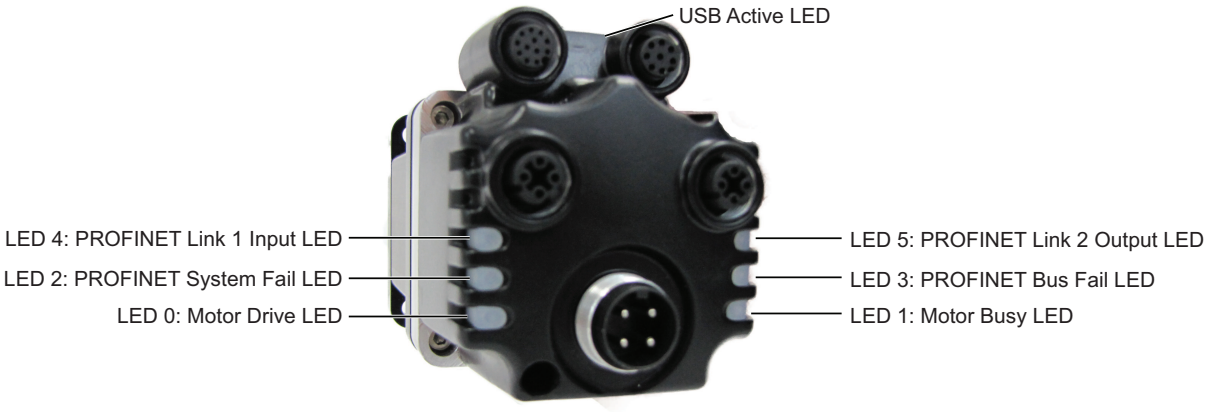
NOTE: Unlike other fieldbus protocols, PROFINET does not require terminators at each end of the network bus.

Maximum Cable Length

For transmission speeds of 100 Megabits/second on shielded Ethernet Cat 5 cable, EtherCAT and PROFINET allow cable lengths up to 100 meters between network nodes.

PROFINET Status LEDs

This following figure and tables describe the functionality of the PROFINET Status LEDs on the SmartMotor.



Flickering = On/Off in 0.1 sec; Blinking = On/Off in 0.5 sec; Flashing = separated by 1 sec for PROFINET LEDs and 2 sec for Fault Codes

USB Active LED

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

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

LED 2: PROFINET System Fail LED

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

LED 4: PROFINET Link 1 Port LED

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

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

LED 3: PROFINET Bus Fail LED

Off	No Error
Solid red	PROFINET Bus failed

LED 5: PROFINET Link 2 Port LED

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

LED Status on Power-up:

- With no program and the travel limit inputs are low:
  LED 0 solid red; motor is 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 Fault Codes:

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

*Busy LED pauses for 2 seconds before flashing the code

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 28.

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("MY_MOTOR01")
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 28.

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 10.
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 PROFINET Status LEDs on page 21.

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 43
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 slave motor:	**Input from slave 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 master to the slave motor. See Command Packet Codes to Motor Commands on page 52 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 master as a response from the slave 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.

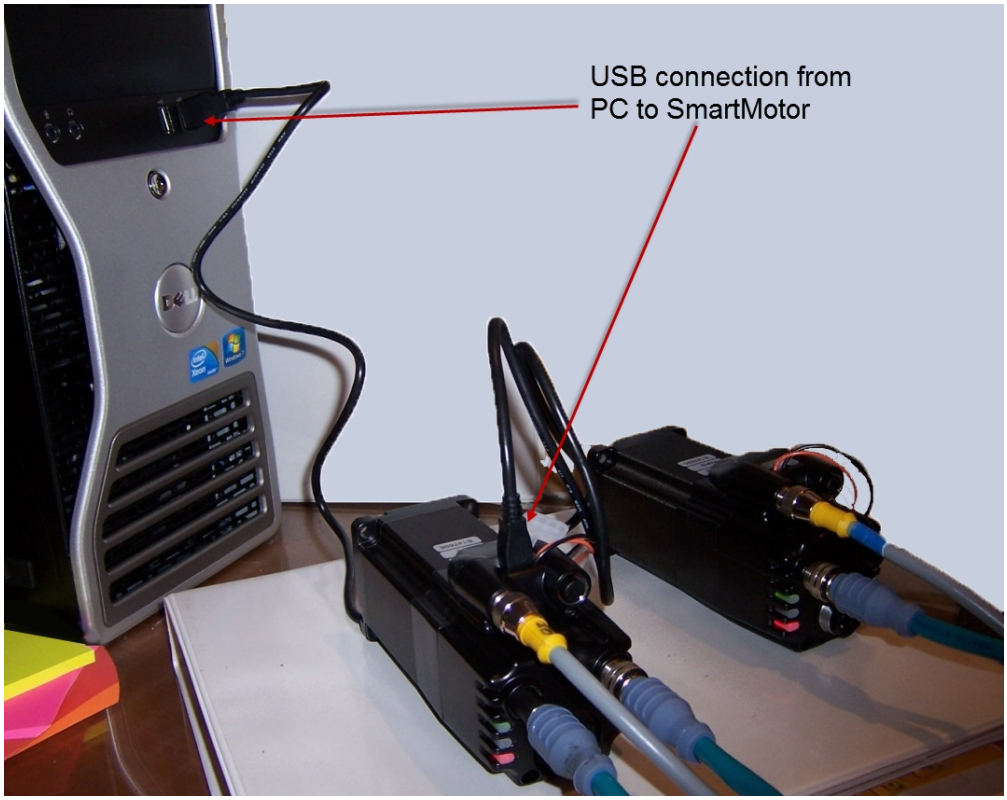
007A 0000 0000 007A **0001 B01A** 0680 0000 0000 0000

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.

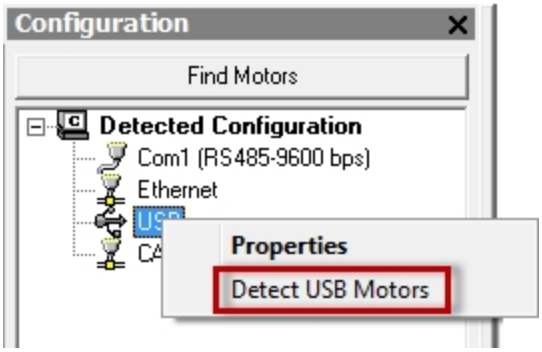
To create a PROFINET connection to the SmartMotor:

1. Install the SMI software. For more details, see the *Moog Animatics SmartMotor™ User's 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.

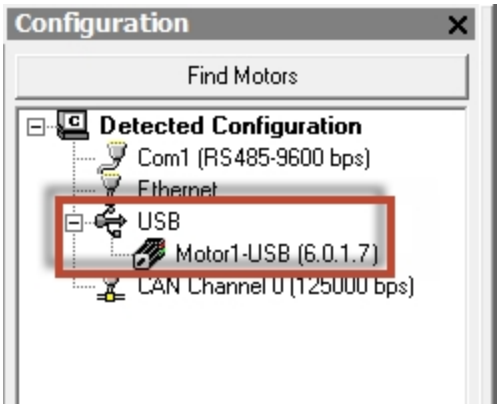


USB Connection from PC to SmartMotor

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

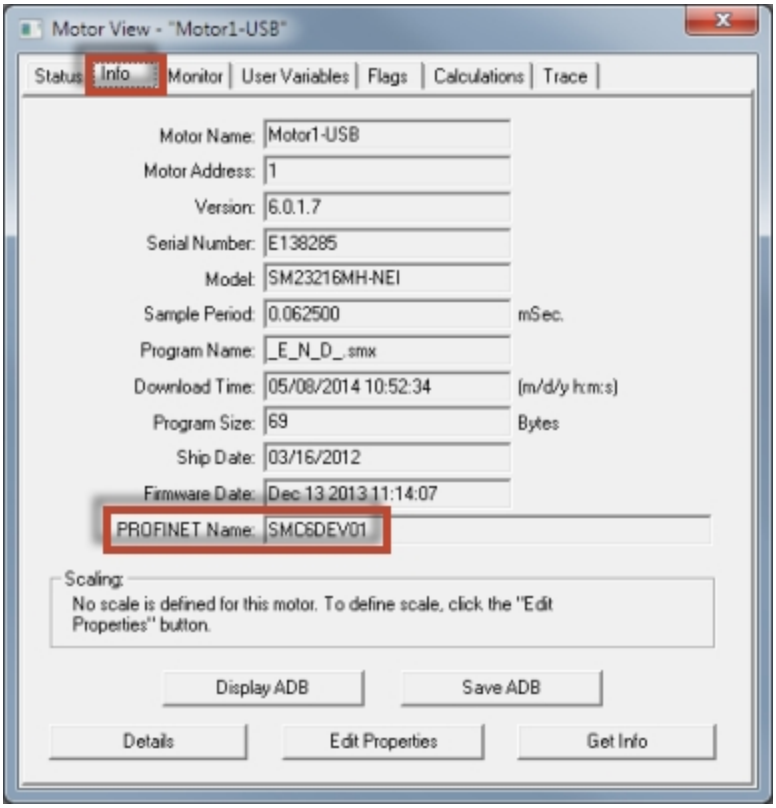


When detection has completed, Motor 1 will be shown under the USB network.



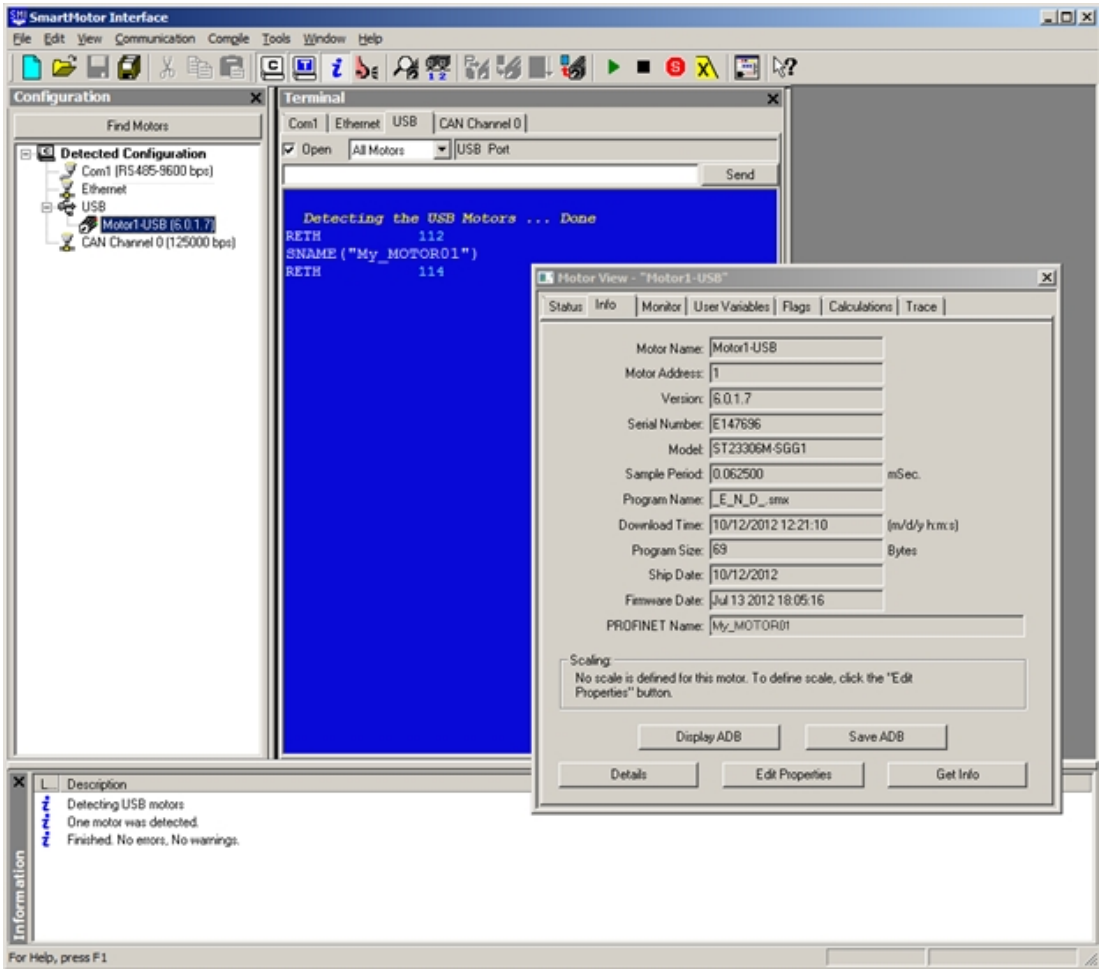
5. Double click Motor1 to open the Motor View tool. Click Poll to update the Status.

6. Select the Info tab. Note the default PROFINET name.



7. Set the station name
    a. Execute RETH to get the current Ethernet interface status bit (112 decimal = 70 hex).
    b. Type SNAME("My_MOTOR01") into the Terminal window.
    c. Execute RETH to get the updated Ethernet interface status bit (114 decimal = 72 hex).

    After the station name has changed, the status for the report from RETH should indicate a PROFINET status configuration change on Bit 1 (zero based). Refer to User Program Commands on page 43.



Entering and Verifying the Station Name

8. Cycle motor power to use the new configuration and station name.
9. Configure your PLC (figure 2) 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 33.

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 52 The symbolic command and response codes are listed, along with their values and the related SmartMotor™ command. See Output and Input Packets on page 45 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 45 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 35.

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 37
- Initiate Relative Position Move on page 39

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 slave motor:

3 words out

0000 0000 0000

Input from slave 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 master to the slave motor.

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

Set command code 0x01 in the output buffer.

0100 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 master as a response from the slave 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.

0000 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 master to the slave motor.

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

Set command code 0x01in the output buffer.

0100 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 master as a response from the slave 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:

0000 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 master to the slave motor.

0000 0000 00**44**	0000 0000 0000 **0086** 0000 0000 0000

Set command code 0x01in the output buffer.

0100 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 master as a response from the slave 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.

0000 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 RVA (polled motor response)
0x01	0xA2	0x21	MT RVA (polled motor response)
0x01		0x0C	G (begin motion)

PLC Memory

Output to slave motor:	**Input from slave 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.

00A2 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.

01A2 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.

00A2 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.

01A2 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

00A2 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 slave motor:
3 words out
0000 0000 0000

Input from slave 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.

00A2 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.

A3A2 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**.

00A2 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**.

01A2 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.

03A2 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	0001 86AD
velocity (100,000 in slew)	
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 table lists the commands used to operate the motor on a PROFINET network.

Command	Description/ Parameter	Values	Non-Volatile Setting
SNAME("string")	Unique PROFINET Station Name	Can use up to 54 characters; factory default is SMC6DEV01. 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. **NOTE:** There are restrictive naming rules that must be followed. For details, see the SNAME command description in the *Moog Animatics SmartMotor™ Command Reference Guide*.	YES
IPCTL(action,"string")	action= 0: set IP address 1: set Mask 2: set Gateway	Not usually needed. Typically the PLC will handle these settings during PROFINET network initialization. Value is formatted as an IP address entered as a string, e.g., IPCTL(0,"192.168.0.10"). By default, these values are set to 0 (i.e., "0.0.0.0")	YES
RETH(0), or x=ETH(0) RETH is the same as RETH(0) x=ETH is the same as x=ETH(0)	PROFINET status	Bit 0 = Initialization failure Bit 1 = Configuration change Bit 2 = Nonvolatile data error Bit 3 = Network processor failure Bit 4 = Port 1 has LINK Bit 5 = Port 2 has LINK Bit 6 = I/O Controller is STOP Bit 7 = I/O Controller is RUN Bit 8 = I/O Controller aborted cyclic communications Bit 9 = Network commanded configuration change	N/A
ETHCTL(1,TBD) ... ETHCTL(5,TBD)	Reserved	Future use	
ETHCTL(6,<value>)	User program label number	Program label to jump to if the NET_LOST_LABEL option is chosen from the NET_LOST_ACTION function. This function has no effect if the NET_LOST_ACTION is anything other than NET_LOST_LABEL.	YES
ETHCTL(7,TBD)	Reserved		
ETHCTL(8,TBD)	Reserved		

Command	Description/ Parameter	Values	Non-Volatile Setting
ETHCTL(9,<value>)	PROFINET Network Lost Action NOTE: Loss of network is an edge-triggered event if I/O Control goes from RUN to any other state.	0 – Ignore, no action (default setting) 1 – Send OFF command to motor 2 – Send X command to motor (soft stop) 3 – Send S command to motor (immediate stop) 4 – Send GOSUB(x) command, where x is the value of the user program label. 5 – Send GOTO(x) command, where x is the value of the user program label.	YES
ETHCTL(10,TBD)	Reserved		
ETHCTL(11,TBD)	Reserved		
ETHCTL(12,<value>)	Network user bit set or clear	0 – Clear Bit 12 of SmartMotor I/O Network Bit 1 – Set Bit 12 of SmartMotor I/O Network Bit	

Program Example

The following code example sets the nonvolatile station name.

```
SNAME("MY_MOTOR01")
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

Output Data Format (I/O Controller Command)

Word	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Command Code							
	1	Response Code							
1	2	Command Data Value (32 bits), big-endian format							
	3								
2	4								
	5								

Input Data Format (Motor response)

Word	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Command Code Acknowledge							
	1	Response Code Acknowledge							
1	2	Response Data Value (32 bits), big-endian format							
	3								
2	4								
	5								
3	6	Status Word (16 bits), big-endian format							
	7								
4	8	Measured Position (32 bits), big-endian format NOTE: This field can be configured to report al[0] or af[0].							
	9								
5	10								
	11								
6	12	Position Error (16 bits), big-endian format							
	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.

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

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

Command Code	Command Data Value	Note	Command Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex	*decimal, hex*				
1, x01	69, x45		Disable software limits	SLD	
1, x01	70, x46		Enable software limits	SLE	
1, x01	71, x47		Make I/O 6 an input; disable GO synchronization function	EIGN(6)	
1, x01	72, x48		Enable GO synchronization function	EISM(6)	
1, x01	73-74, x49-x4A	Reserved			
1, x01	75, x4B		Arm index capture from internal encoder, rising edge	Ai(0)	
1, x01	76, x4C		Arm index capture from internal encoder, falling edge	Aj(0)	
1, x01	77, x4D		Arm index capture from internal encoder, rising then falling edge	Aij(0)	
1, x01	78, x4E		Arm index capture from internal encoder, falling then rising edge	Aji(0)	
1, x01	79, x4F		Arm index capture from external encoder, rising edge	Ai(1)	
1, x01	80, x50		Arm index capture from external encoder, falling edge	Aj(1)	
1, x01	81, x51		Arm index capture from internal encoder, rising then falling edge	Aij(1)	
1, x01	82, x52		Arm index capture from internal encoder, falling then rising edge	Aji(1)	
1, x01	83, x53		Immediately force trapezoidal commutation mode	MDT	
1, x01	84, x54		Request enhanced trapezoidal commutation mode; entered as soon as angle is satisfied	MDE	
1, x01	85, x55		Request sine commutation mode (voltage mode); entered as soon as angle is satisfied	MDS	
1, x01	86, x56		Request current-controlled sine mode; entered as soon as angle is satisfied.	MDC	
1, x01	87, x57		Turn on Trajectory Overshoot Braking (TOB) feature for trapezoidal mode	MDB	
1, x01	88+, x58+	Reserved			
2, x02	<value>	DO_MOVE_POS_ABS	Set absolute position and start motor	PT=<value> G	
3, x03	<value>	DO_MOVE_POS_REL	Set relative position and start motor	PRT=<value> G	
4, x04	<value>	DO_MOVE_VEL	Set velocity and start motor	VT=<value> G	
5, x05	<value>		Call a subroutine	GOSUB(<value>)	
6, x06	<value>		Branch program execution to a label	GOTO(<value>)	
7-89, x07-x59		Reserved			
90, x5A	<value>		Clear mask on user bits, word 0, status word 12	UR(W,0,<value>)	
91, x5B	<value>		Clear mask on user bits, word 1, status word 13	UR(W,1,<value>)	

Command Code	Command Data Value	Note	Command Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex	*decimal, hex*				
92, x5C	<value>		Set mask on user bits, word 0, status word 12	US(W,0,<value>)	
93, x5D	<value>		Set mask on user bits, word 1, status word 13	US(W,1,<value>)	
94, x5E	<value>		Clear specific user bit 0-31	UR(<value>)	
95, x5F	<value>		Set specific user bit 0-31	US(<value>)	
96, x60	<value>		Set output 8 to 0 or 1, ON is 1 sourcing	OUT(8)=<value>	
97-99, x61-x63	<value>	Reserved			
100, x64	<value>		Set acceleration	ADT=<value>	
101, x65	<value>		Set RS-232/RS-485 address	ADDR=<value>	
102, x66	<value>		Set PWM drive signal limit	AMPS=<value>	
103-123, x67-x7B		Reserved			
124, x7C	<value>		Set relative distance (position)	PRT=<value>	
125, x7D	<value>		Set allowable position error	EL=<value>	
126, x7E		Reserved			
127, x7F		Obsolete			
128, x80		Reserved			
129, x81	<value>		PID acceleration feed forward	KA=<value>	
130, x82	<value>		PID derivative compensation	KD=<value>	
131, x83	<value>		PID gravity compensation; for limits, see the *Moog Animatics SmartMotor™ User's Guide*	KG=<value>	
132, x84	<value>		PID integral compensation	KI=<value>	
133, x85	<value>		PID integral limit	KL=<value>	
134, x86	<value>		PID proportional compensation	KP=<value>	
135, x87	<value>		PID derivative term sample rate	KS=<value>	
136, x88	<value>		PID velocity feed forward	KV=<value>	
137, x89	<value>		Mode follow with ratio divisor	MFDIV=<value>	
138, x8A	<value>		Mode follow with ratio multiplier	MFMUL=<value>	
139, x8B	<value>		Set origin	O=<value>	
140, x8C	<value>		Shift origin	OSH(<value>)	
141, x8D		Reserved			
142, x8E	<value>		Set absolute position target	PT=<value>	
143-144, x8F-x90		Reserved			
145, x91	<value>		Set RS-232/RS-485 address	SADDR<value>	
146-147, x92-x93		Reserved			
148, x94	<value>		Assign torque value in torque mode	T=<value>	
149, x95		Reserved			
150, x96	<value>		Set maximum allowable temperature (high limit)	TH=<value>	
151-162, x97-xA2		Reserved			
163, xA3	<value>		Set velocity target	VT=<value>	
164, xA4		Reserved			

Command Code	Command Data Value	Note	Command Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex	*decimal, hex*				
165, xA5	<value>		Set value of negative software limit	SLN=<value>	
166, xA6	<value>		Set value of positive software limit	SLP=<value>	
167-169, xA7-xA9		Reserved			
170, xAA	<value>		Clear status word 0; bit indicated by value	Z(0,<value>)	
171, xAB	<value>		Clear status word 1; bit indicated by value	Z(1,<value>)	
172, xAC	<value>		Clear status word 2; bit indicated by value	Z(2,<value>)	
173, xAD	<value>		Clear status word 3; bit indicated by value	Z(3,<value>)	
174, xAE	<value>		Clear status word 4; bit indicated by value	Z(4,<value>)	
175, xAF	<value>		Clear status word 5; bit indicated by value	Z(5,<value>)	
176, xB0	<value>		Clear status word 6; bit indicated by value	Z(6,<value>)	
177-199, xB1-xC7		Reserved			
200, C8	<a-zzz> 0-77	SET_VAR_ INDEX_SET	u8VarIndexSet = <value> u8VarIndexSetActual = <value>		
201, xC9		Reserved			
202, xCA	<value> 0-78	SET_VAR_ LEN_SET	u8VarLenSet = <value>		
203, xCB	<value> 0-203 ab[] 0-101 aw[] 0-50 al[]	SET_ARRAY_ INDEX_SET	u8ArrIndexSet = <value> u8ArrIndexSetActual = <value>		
204, xCC		Reserved			
205, xCD	<value> 0-204 ab[] 0-102 aw[] 0-51 al[]	SET_ARR_ LEN_SET	u8ArrLenSet = <value>		
206, xCE	<value> 0=NO, 1=YES	SET_AUTO_ INC_SET	u8AutoIncSet = <value>		
207, xCF	<a-zzz> 0-77	SET_VAR_ INDEX_GET	u8VarIndexGet = <value> u8VarIndexGetActual = <value>		
208, xD0		Reserved			
209, xD1	<value> 0-78	SET_VAR_ LEN_GET	u8VarLenGet = <value>		
210, xD2	<value> 0-203 ab[] 0-101 aw[] 0-50 al[]	SET_ARRAY_ INDEX_GET	u8ArrIndexGet = <value> u8ArrIndexGetActual = <value>		
211, xD3		Reserved			
212, xD4	<value> 0-204 ab[] 0-102 aw[] 0-51 al[]	SET_ARR_ LEN_GET	u8ArrLenGet = <value>		
213, xD5	<value> 0=NO, 1=YES	SET_AUTO_ INC_GET	u8AutoIncGet = <value>		

Command Code	Command Data Value	Note	Command Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex	*decimal, hex*				
214, xD6	<value>	SET_VAR	Set variable <a to zzzz> ='a'+u8VarIndexSetActual; if (u8AutoIncSet) then u8VarIndexSetActual += 1	<a to zzz>= <value>	
215, xD7	<value>	SET_ARRAY_ BYTE	Set byte array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then u8ArrIndexSetActual += 1	ab[<index>]= <value>	
216, xD8	<value>	SET_ARRAY_ WORD	Set word array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then u8ArrIndexSetActual += 1	aw[<index>]= <value>	
217, xD9	<value>	SET_ARRAY_ LONG	Set long array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then u8ArrIndexSetActual += 1	al[<index>]= <value>	
218, xDA	<value>	SET_NVOL_ BYTE	Store byte to EEPROM u32EptrActual += 1	VST(<value byte>,1)	
219, xDB	<value>	SET_NVOL_ WORD	Store word to EEPROM u32EptrActual += 2	VST(<value word16>,1)	
220, xDC	<value>	SET_NVOL_ LONG	Store long to EEPROM u32EptrActual += 4	VST(<value long>,1)	
221, xDD	<value>	SET_NVOL_ VAR	Set variable and store to EEPROM <a to z>='a'+u8VarIndexSetActual u32EptrActual += 4; if (u8AutoIncSet) then u8VarIndexSetActual += 1	<a to z>= <value> VST(<a to z>,1)	
222, xDE		STORE_ NVOL_VARS	Store variables to EEPROM <a to z>='a'+u8VarIndexSetActual <length>=u8VarLenSet u32EptrActual += (<length>*4); if (u8AutoIncSet) then u8VarIndexSetActual += <length>	VST(<a to z>, <length>)	
223, xDF		STORE_ NVOL_ ARRAY_BYTE	Store byte array variables to EEPROM <index>=u8ArrIndexSetActual <length>=u8ArrLenSet u32EptrActual += (<length>*1); if (u8AutoIncSet) then u8ArrIndexSetActual += <length>	VST(ab[<index>], <length>)	
224, xE0		STORE_ NVOL_ ARRAY_ WORD	Store word array variables to EEPROM <index>=u8ArrIndexSetActual <length>=u8ArrLenSet u32EptrActual += (<length>*2); if (u8AutoIncSet) then u8ArrIndexSetActual += <length>	VST(aw [<index>], <length>)	
225, xE1		STORE_ NVOL_ ARRAY_ LONG	Store long array variables to EEPROM <index>=u8ArrIndexSetActual <length>=u8ArrLenSet u32EptrActual += (<length>*4); if (u8AutoIncSet) then u8ArrIndexSetActual += <length>	VST(al[<index>], <length>)	
226, xE2	<value>		Set the EEPROM address u32EptrSet=<value> u32EptrActual=<value> (doesn't affect EPTR)		
227-239, xE3-xEF		Reserved			

Command Code	Command Data Value	Note	Command Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex	*decimal, hex*				
240, xF0	<value>	SET_PA_ FIELD	Configure the PROFINET input packet to use an alternate data source for the 'Measured position' field (words 4,5) <value>: 0 - report actual position in encoder counts (this is the power-up default value) 1 - report al[0] (big-endian format) 2 - report af[0] (IEEE-754 32-bit single precision, big-endian format)	CANCTL(10,x)	
241-254, xF1-xFE		Reserved			
255, xFF		ERROR	Returned if the command code could not be performed successfully		

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	Response Data Value	Note	Response Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex					
0, x00	0	NULL	No command		
1-95,					
x01-x5f		Reserved			
96, x60	<value>		Report digital I/O number 4	RIN(4)	
97, x61	<value>		Report analog input number 4	RINA(A,4)	
98, x62	<value>		Report digital I/O number 5	RIN(5)	
99, x63	<value>		Report analog input number 5	RINA(A,5)	
100, x64	<value>		Report acceleration target	RAT	<value>
101, x65	<value>		Get SmartMotor address	RADDR	<value>
102, x66	<value>		Report assigned PWM limit	RAMPS	<value>
103, x67	<value>		Report overcurrent status	RBa	<value>
104, x68	<value>	Obsolete			
105, x69	<value>	Obsolete			
106, x6A	<value>	Obsolete			
107, x6B	<value>		Report position error status	RBe	<value>
108, x6C	<value>	Obsolete			
109, x6D	<value>		Report overheat status	RBh	<value>
110, x6E	<value>		Report index status	RBi	<value>
111, x6F	<value>		Report program checksum error	RBk	<value>
112, x70	<value>		Report historical left limit status	RBl	<value>
113, x71	<value>		Report negative limit status	RBm	<value>
114, x72	<value>		Report motor off status	RBo	<value>
115, x73	<value>		Report positive limit status	RBp	<value>
116, x74	<value>		Report historical right limit status	RBr	<value>
117, x75	<value>		Report program scan status	RBs	<value>
118, x76	<value>		Report trajectory status	RBt	<value>
119, x77	<value>	Obsolete			
120, x78	<value>		Report wrapped encoder position	RBw	<value>
121, x79	<value>		Report hardware index input level	RBx	<value>
122, x7A	<value>		Report millisecond clock	RCLK	<value>
123, x7B	<value>		Report secondary counter	RCTR(1)	<value>
124, x7C	<value>		Report buffered move distance value	RPRC	<value>
125, x7D	<value>		Report buffered maximum position error	REL	<value>

Response Code	Response Data Value	Note	Response Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex					
126, x7E		Reserved			
127, x7F	<value>	Obsolete			
128, x80	<value>		Report index position captured from recent Ai(0) command - object 1, data value 75.	RI(0)	<value>
129, x81	<value>		Report buffered acceleration feed forward coefficient	RKA	<value>
130, x82	<value>		Report buffered derivative coefficient	RKD	<value>
131, x83	<value>		Report buffered gravity coefficient	RKG	<value>
132, x84	<value>		Report buffered integral coefficient	RKI	<value>
133, x85	<value>		Report buffered integral limit	RKL	<value>
134, x86	<value>		Report buffered proportional coefficient	RKP	<value>
135, x87	<value>		Report buffered sampling interval	RKS	<value>
136, x88	<value>		Report buffered velocity feed forward coefficient	RKV	<value>
137, x89	<value>		Report follow mode divisor	RMFDIV	<value>
138, x8A	<value>		Report follow mode multiplier	RMFMUL	<value>
139, x8B		Reserved			
140, x8C	<value>		Report current mode of operation	RMODE	<value>
141, x8D	<value>		Report present position	RPA	<value>
142, x8E	<value>		Report buffered position setpoint	RPT	<value>
143, x8F	<value>		Report present position error	REA	<value>
144-147, x90-x93		Reserved			
148, x94	<value>		Report current requested torque	RT	<value>
149, x95	<value>		Report temperature	RTEMP	<value>
150, x96	<value>		Report temperature shutdown limit	RTH	<value>
151, x97	<value>		Report current algorithm THD time	RTHD	<value>
152, x98	0=Bit 0 Off 1=Bit 0 On		Report digital I/O number 0	RIN(0)	<value>
153, x99	<value>		Report analog input number 0	RINA(A,0)	<value>
154, x9A	0=Bit 1 Off 1=Bit 1 On		Report digital I/O number 1	RIN(1)	<value>
155, x9B	<value>		Report analog input number 1	RINA(A,1)	<value>
156, x9C	0=Bit 2 Off 1=Bit 2 On		Report digital I/O number 2	RIN(2)	<value>
157, x9D	<value>		Report analog input number 2	RINA(A,2)	<value>
158, x9E	0=Bit 3 Off 1=Bit 3 On		Report digital I/O number 3	RIN(3)	<value>
159, x9F	<value>		Report analog input number 3	RINA(A,3)	<value>
160, xA0	0=Bit 6 Off 1=Bit 6 On		Report digital I/O number 6	RIN(6)	<value>

Response Code	Response Data Value	Note	Response Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex					
161, xA1	<value>		Report analog input number 6	RINA(A,6)	<value>
162, xA2	<value>		Report velocity	RVA	<value>
163, xA3	<value>		Report buffered maximum velocity	RVT	<value>
164, xA4	<value>		Report legacy status word	(n/a)	<value>
165, xA5	<value>		value of negative software limit	RSLN	<value>
166, xA6	<value>		value of positive software limit	RSLP	<value>
167, xA7	<value>	Reserved			
168, xA8	<value>		Inputs 0-7, 16-bit value, right justified	RIN(W,0)	<value>
169, xA9		Reserved			
170, xAA	<value>		Report status word 0	RW(0)	<value>
171, xAB	<value>		Report status word 1	RW(1)	<value>
172, xAC	<value>		Report status word 2	RW(2)	<value>
173, xAD	<value>		Report status word 3	RW(3)	<value>
174, xAE	<value>		Report status word 4	RW(4)	<value>
175, xAF	<value>		Report status word 5	RW(5)	<value>
176, xB0	<value>		Report status word 6	RW(6)	<value>
177, xB1	<value>		Report status word 7	RW(7)	<value>
178, xB2	<value>		Report status word 8	RW(8)	<value>
179, xB3	<value>		Report status word 9	RW(9)	<value>
180-181, xB4-xB5	<value>	Reserved			
182, xB6	<value>		Report user bits 0-15 (status word 12)	RW(12)	
183, xB7	<value>		Report user bits 16-31 (status word 13)	RW(13)	
184-185, xB8-xB9	<value>	Reserved			
186, xBA	<value>		Report I/O 0-7 (status word 16)	RW(16)	
187-199, xBB-xC7		Reserved			
200, xC8	<value>	GET_VAR_ INDEX_SET	<value>=u8VarIndexSet		
201, xC9	<value>	GET_VAR_ INDEX_SET_ ACTUAL	<value>=u8VarIndexSetActual		
202, xCA	<value>	GET_VAR_ LEN_SET	<value>=u8VarLenSet		
203, xCB	<value>	GET_ARRAY_ INDEX_SET	<value>=u8ArrIndexSet		
204, xCC	<value>	GET_ARRAY_ INDEX_SET_ ACTUAL	<value>=u8ArrIndexSetActual		
205, xCD	<value>	GET_ARR_ LEN_SET	<value>=u8ArrLenSet		
206, xCE	<value>	GET_AUTO_ INC_SET	<value>=u8AutoIncSet		
207, xCF	<value>	GET_VAR_ INDEX_GET	<value>=u8VarIndexGet		
208, xD0	<value>	GET_VAR_ INDEX_GET_ ACTUAL	<value>=u8VarIndexGetActual		

Response Code	Response Data Value	Note	Response Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex					
209, xD1	<value>	GET_VAR_ LEN_GET	<value>=u8VarLenGet		
210, xD2	<value>	GET_ARRAY_ INDEX_GET	<value>=u8ArrIndexGet		
211, xD3	<value>	GET_ARRAY_ INDEX_GET_ ACTUAL	<value>=u8ArrIndexGetActual		
212, xD4	<value>	GET_ARR_ LEN_GET	<value>=u8ArrLenGet		
213, xD5	<value>	GET_AUTO_ INC_GET	<value>=u8AutoIncGet		
214, xD6	<value>	GET_VAR	Get variable <a to zzz>='a'+u8VarIndexGetActual; if (u8AutoIncGet) then u8VarIndexGetActual += 1	R<a to zzz> (Issued only one time)	<value>
215, xD7	<value>	GET_ARRAY_ BYTE	Get byte array variable <index>=u8ArrIndexGetActual; if (u8AutoIncGet) then u8ArrIndexGetActual += 1	Rab[<index>] (Issued only one time)	<value>
216, xD8	<value>	GET_ARRAY_ WORD	Get word array variable <index>=u8ArrIndexGetActual; if (u8AutoIncGet) then u8ArrIndexGetActual += 1	Raw[<index>] (Issued only one time)	<value>
217, xD9	<value>	GET_ARRAY_ LONG	Get long array variable <index>=u8ArrIndexGetActual; if (u8AutoIncGet) then u8ArrIndexGetActual += 1	Ral[<index>] (Issued only one time)	<value>
218, xDA	<value>	GET_NVOL_ BYTE	Get byte from EEPROM u32EptrActual += 1	VLD(<value>,1) (Issued only one time)	<value>
219, xDB	<value>	GET_NVOL_ WORD	Get word from EEPROM u32EptrActual += 2	VLD(<value>,1) (Issued only one time)	<value>
220, xDC	<value>	GET_NVOL_ LONG	Get long from EEPROM u32EptrActual += 4	VLD(<value>,1) (Issued only one time)	<value>
221, xDD	<value>	GET_NVOL_ VAR	Get variable from EEPROM <a to zzz>='a'+u8VarIndexGetActual; u32EptrActual += 4 if (u8AutoIncGet) then u8VarIndexGetActual += 1	VLD(<a to zzz>,1) R<a to zzz> (Issued only one time)	<value>

222, xDE		LOAD_NVOL_ VARS	Load variables from EEPROM
			<a to zzz>='a'+u8VarIndexGetActual
			<length>=u8VarLenGet
			u32EptrActual += (<length>*4);
			if (u8AutoIncGet) then
			u8VarIndexGetActual += <length>
				VLD(<a to zzz>,<length>)
				(Issued only one time)	

223, xDF		LOAD_NVOL_ ARRAY_BYTE	Load byte array variables from EEPROM
			<index>=u8ArrIndexGetActual
			<length>=u8ArrLenGet
			u32EptrActual += (<length>*1);
			if (u8AutoIncGet) then
			u8ArrIndexGetActual += <length>
				VLD(ab[<index>],<length>)
				(Issued only one time)	

Response Code	Response Data Value	Note	Response Description	Smart Motor Command(s)	Smart Motor Response
decimal, hex					
224, xE0		LOAD_NVOL_ ARRAY_WORD	Load word array variables from EEPROM <index>=u8ArrIndexGetActual <length>=u8ArrLenGet u32EptrActual += (<length>*2); if (u8AutoIncGet) then u8ArrIndexGetActual += <length>	VLD(aw [<index>], <length>) (Issued only one time)	
225, xE1		LOAD_NVOL_ ARRAY_LONG	Load long array variables from EEPROM <index>=u8ArrIndexGetActual <length>=u8ArrLenGet u32EptrActual += (<length>*4); if (u8AutoIncGet) then u8ArrIndexGetActual += <length>	VLD(al [<index>], <length>) (Issued only one time)	
226, xE2	<value>		Get last set EEPROM address <value>=u32EptrSet		
227, xE3	<value>		Get actual EEPROM address setting in PROFINET interface <value>=u32EptrActual		
228, xE4	<value>	GET_NET_ LOST_LABEL	<value>=u16NetLostLabel (initialized to the value of u16NetLostLabelDefault during power-up)		
229, xE5	<value>	GET_NET_ LOST_ACTION	<value>=u8NetLostAction (initialized to the value of u8NetLostActionDefault during power-up)	Upon loss of communication with PROFINET host, command is based on <value>: 0=IGNORE (No Command), 1=OFF (Motor Off), 2=X (Soft Stop), 3=S (Immediate Stop), 4=GOSUB, 5=GOTO	
230-234, xE6-xEA		Reserved			
235, xEB	<value>	GET_ENC_ RESOLUTION	<value>=s32EncResolution (read at startup from SmartMotor EEPROM location 32000)	RRES	
236, xEC	<value>	GET_ FIRMWARE_ VERSION	<value>=u32FirmwareVersion (read at startup with RSP)	RFW	
237, xED		Reserved			
238, xEE	<value>	GET_SAMPLE_ RATE	<value>=s32SampleRate (read at startup with RSP)		
239-254, xF0-xFE		Reserved			
255, xFF	<error code>	ERROR	Returned if the response code could not be performed successfully (<error code> values to be determined)		

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 slave and master.	Check that connectors are correctly wired and connected to motor. For details, see PROFINET Motor Connectors and Pinouts on page 18.
	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 19.
	Wrong GSDML file.	Verify that the correct GSDML file was used to configure the master and connect the slave motor as part of the PROFINET network.
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.

Issue	Cause	Solution
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.
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 6 Moog Animatics SmartMotor™ User's Guide*

PN: SC80100007-001
Rev. A