PROFIBUS® IMPLEMENTATION FOR

FULLY INTEGRATED SERVO MOTORS

CLASS 5 SMARTMOTOR™ WITH COMBITRONIC™ TECHNOLOGY

Rev. E, February 2026

DESCRIBES THE CLASS 5 SMARTMOTOR™ SUPPORT FOR THE PROFIBUS® PROTOCOL

www.animatics.com

MOOG ANIMATICS

Copyright Notice

Moog Animatics Class 5 SmartMotor™ PROFIBUS Guide, Rev. E, PN: SC80100009-001.

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 "PROFIBUS 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 6

Purpose 7

PROFIBUS Overview 7

PROFIBUS Features 8

Equipment Required 9

Hardware 9

Software 9

Safety Information 10

Safety Symbols 10

Other Safety Considerations 10

Motor Sizing 10

Environmental Considerations 10

Machine Safety 11

Documentation and Training 11

Additional Equipment and Considerations 12

Safety Information Resources 12

Additional Documents 12

Related Guides 13

Other Documents 13

Additional Resources 14

PROFINET and PROFIBUS Resources 14

Connections, Wiring and Status LEDs 15

PROFIBUS Motor Connectors and Pinouts 16

D-Style Motor PROFIBUS Connector and Pinouts 16

Other D-Style Motor Connectors and Pinouts 16

Cables and Diagram 17

PROFIBUS Cables and Connectors 17

Example PROFIBUS Cable 17

Example PROFIBUS Connector 18

PROFIBUS Cable Diagram 18

Cable Length 19

PROFIBUS Status LEDs 20

PROFIBUS Configuration 21

Configure Motor with PC 22

User Program Requirements 22

Required Nonvolatile EEPROM Values 22
Reserved Motor Variables 22
Configure PLC with PC 23
Configure SmartMotor to PROFIBUS 23
PLC Sends Commands to Motor 23
Network Data Format Example 23
Status Bits for Class 5 Mode (Default) 24
Status Bits for Class 4 Emulation Mode 25
PLC Memory 25
Sequence to Set Report Data to Motor Clock 26
PROFIBUS Communication Example 27

Sample Command Sequences 30

Overview 31
Command and Response Codes 31
Handshaking of Messages 31
Disabling Limits from Preventing Motion 31
Turning the Motor Shaft 31
Disable Limits and Clear Fault Status 32
Commands 32
PLC Memory 32
Disable positive limit, command EIGN(2) 32
Disable negative limit, command EIGN(3) 32
Clear fault status, command ZS 33
Initiate Mode Torque 34
Commands 34
PLC Memory 34
Set torque value, specify the response data 34
Initiate torque mode, command MT 34
Initiate Relative Position Move 36
Commands 36
PLC Memory 36
Set acceleration value, command ADT=255 36
Set maximum velocity value, command VT=100000 36
Make a relative position move 37

Nonvolatile Data 38

Program Example 39

Output and Input Packets 40

Output and Input Packet Format 41

Command (Output) Packet Notes 43
Response (Input) Packet Notes 44

Alternate Communications Channel 45
Reserved Motor Variables 45

Command and Response Codes 46
Command Packet Codes to Motor Commands 47
Response Packet Codes to Motor Commands 56

Troubleshooting 62

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 **7**

PROFIBUS Overview **7**

PROFIBUS Features **8**

Equipment Required **9**

Hardware 9

Software 9

Safety Information **10**

Safety Symbols 10

Other Safety Considerations 10

Motor Sizing 10

Environmental Considerations 10

Machine Safety 11

Documentation and Training 11

Additional Equipment and Considerations 12

Safety Information Resources 12

Additional Documents **12**

Related Guides 13

Other Documents 13

Additional Resources **14**

PROFINET and PROFIBUS Resources **14**

Purpose

This manual explains the Moog Animatics Class 5 SmartMotor™ support for the PROFIBUS® protocol. It describes the major concepts that must be understood to integrate a SmartMotorfollower with a PLC or other PROFIBUS controller[1]. However, it does not cover all the low-level details of the PROFIBUS protocol.

NOTE: The feature set described in this version of the manual refers to motor firmware 5.32.4.x.

This manual is intended for programmers or system developers who understand the use of PROFIBUS. (PROFIBUS communication is described in IEC61158 Type 3 and IEC61784.) Therefore, this manual is not a tutorial on those specifications or the PROFIBUS 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 PROFIBUS 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 PROFIBUS protocol. The commands include those required by the specification and those added by Moog Animatics. For details, see Command and Response Codes on page 46. Also, see Nonvolatile Data on page 38.

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.

PROFIBUS Overview

PROFIBUS is an independent, open fieldbus standard that allows different manufacturers of automation products to communicate without special interface adjustments. Specifically, PROFIBUS, 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 PROFIBUS. For details, see Command Packet Codes to Motor Commands on page 47. These codes generally correspond to Class 5 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 PROFIBUS SmartMotor is a SmartMotor with the addition of the PROFIBUS connectors and interface board, which then accepts commands as a follower over a PROFIBUS network. In addition to communicating over PROFIBUS, SmartMotor commands may be sent through other communication interfaces of the SmartMotor. Depending on the SmartMotor model, it may also communicate over RS-232 or RS-485.

The Moog Animatics communications profile over PROFIBUS 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 PROFIBUS operation.

A PROFIBUS 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 PROFIBUS 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.

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

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

PROFIBUS Features

Moog Animatics PROFIBUS features include:

- Command/Response Codes for all Class 5 SmartMotor commands
- Use of onboard I/O via PROFIBUS, SmartMotor program, or RS-232 commands
- Ability to run 1000 SmartMotor subroutines through PROFIBUS
- Ability to read/write all SmartMotor variables
- SmartMotor online diagnostics through SMI software and RS-232 connection
- Up to 127 PROFIBUS nodes
- 250 microsecond interrupt-driven subroutine
- Data rates: 1.5 Mbps (default); 9.6, 19.2, 31.25, 45.45, 93.75, 187.5, 500 Kbps; 1.5, 3, 6, 12 Mbps

NOTE: PROFIBUS baud rates are achievable only with proper cable length and termination connectors. The minimum cable length when operating >=1 MBaud is 1 meter (~3 feet). If the cable is too short, reflected impedance can cause loss of communication data packets and spurious node errors. For more cabling details, refer to Cables and Diagram on page 17.

Equipment Required

The section describes the required PROFIBUS hardware and software.

Hardware

The following hardware is required:

- Moog Animatics PROFIBUS SmartMotor™
- Moog Animatics power supply or user-supplied equivalent
- Moog Animatics RS-232 or RS-485 communications cable that is compatible with the SmartMotor
- User-supplied PC with the Microsoft Windows operating system; for the Class 5 SmartMotor, an RS-232 port is required
- For the Class 5 SmartMotor, Moog Animatics RS-232 to RS-485 converter
- User-supplied PLC with PROFIBUS controller or other PROFIBUS controller
- Moog Animatics PROFIBUS cable, or equivalent, and connectors with correct terminating resistors

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 PROFIBUS 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 PROFIBUS GSD configuration file name will have the form "SM5_070C.GSD". Make sure you obtain the latest version of the file.

  PROFIBUS GSD file and firmware combinations:

  - SM5_070C.GSD requires firmware 5.32.4.9 or newer.
  - Firmware 5.32.4.9 or newer may use DEAF070C.GSD; however, SM5_070C.GSD is strongly recommended.
  - Firmware 5.32.4.7 or previous must use DEAF070C.GSD; contact Moog Animatics for that 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 5 SmartMotor™ Installation and Startup Guide*

  http://www.animatics.com/cl-5-install-startup-guide

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

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

Connections, Wiring and Status LEDs

This chapter provides information on the SmartMotor system connections, a multidrop cable diagram, and 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.

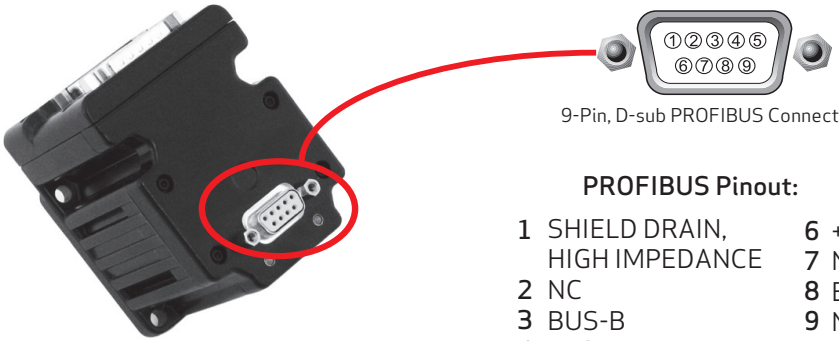
PROFIBUS Motor Connectors and Pinouts **16**
D-Style Motor PROFIBUS Connector and Pinouts 16
Other D-Style Motor Connectors and Pinouts 16
Cables and Diagram **17**
PROFIBUS Cables and Connectors 17
Example PROFIBUS Cable 17
Example PROFIBUS Connector 18
PROFIBUS Cable Diagram 18
Cable Length 19
PROFIBUS Status LEDs **20**

PROFIBUS Motor Connectors and Pinouts

The following figure provides an overview of the PROFIBUS connectors and pinouts available on the Class 5 SmartMotors.

D-Style Motor PROFIBUS Connector and Pinouts

The following figure shows the location of the PROFIBUS connector on the back of the D-style SmartMotor.

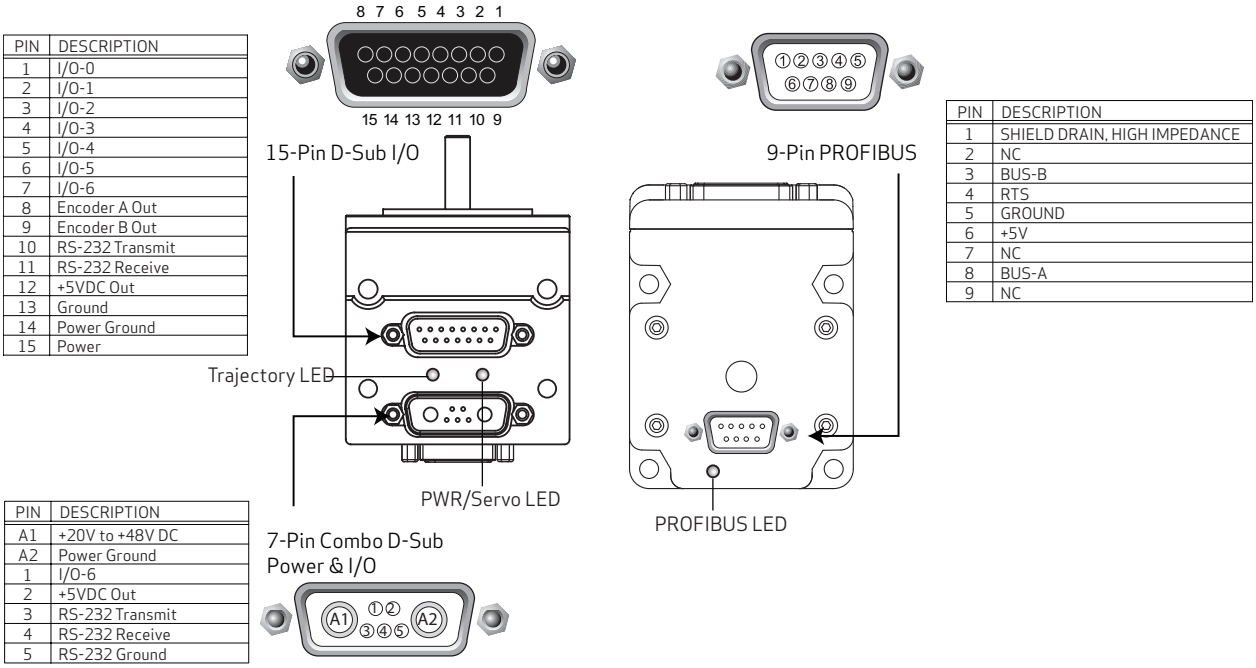


PROFIBUS Pinout:

1	SHIELD DRAIN, HIGH IMPEDANCE	6	+5V
2	NC	7	NC
3	BUS-B	8	BUS-A
4	RTS	9	NC
5	GROUND		

Other D-Style Motor Connectors and Pinouts

The following figure shows the locations of all connectors and status LEDs that are included on the D-style SmartMotor with the PROFIBUS option.



PIN	DESCRIPTION
1	I/O-0
2	I/O-1
3	I/O-2
4	I/O-3
5	I/O-4
6	I/O-5
7	I/O-6
8	Encoder A Out
9	Encoder B Out
10	RS-232 Transmit
11	RS-232 Receive
12	+5VDC Out
13	Ground
14	Power Ground
15	Power

PIN	DESCRIPTION
1	SHIELD DRAIN, HIGH IMPEDANCE
2	NC
3	BUS-B
4	RTS
5	GROUND
6	+5V
7	NC
8	BUS-A
9	NC

PIN	DESCRIPTION
A1	+20V to +48V DC
A2	Power Ground
1	I/O-6
2	+5VDC Out
3	RS-232 Transmit
4	RS-232 Receive
5	RS-232 Ground

Cables and Diagram

This section provides some general information on connecting Moog Animatics PROFIBUS SmartMotors.

NOTE: For full details and requirements on PROFIBUS cables and termination procedures, consult the PROFIBUS standards.

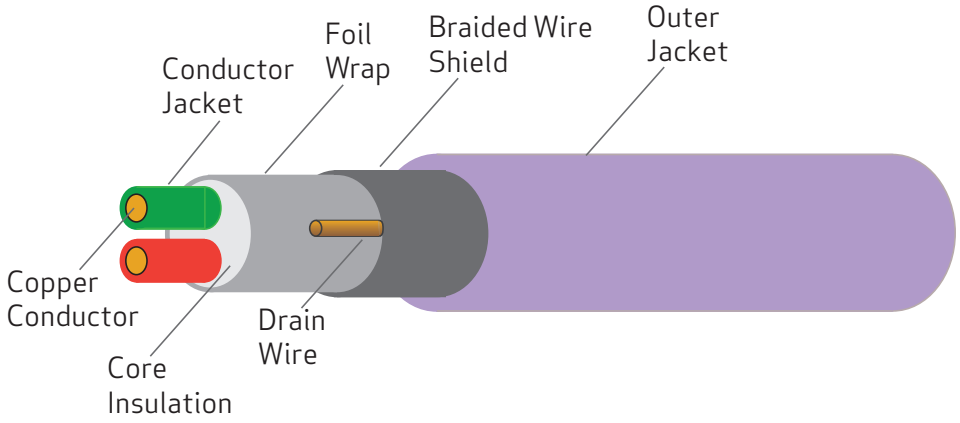
PROFIBUS Cables and Connectors

Moog Animatics does not currently stock or supply PROFIBUS cables or connectors. However, these items are readily available from industrial electronics suppliers.

Example PROFIBUS Cable

The following figure shows a cross-section of typical PROFIBUS cable. The conductor wires attach to the A and B terminals on the PROFIBUS connector.

CAUTION: Make certain that you always attach the same wire color to the same connector terminal (e.g., green wire to A terminal and red wire to B terminal).

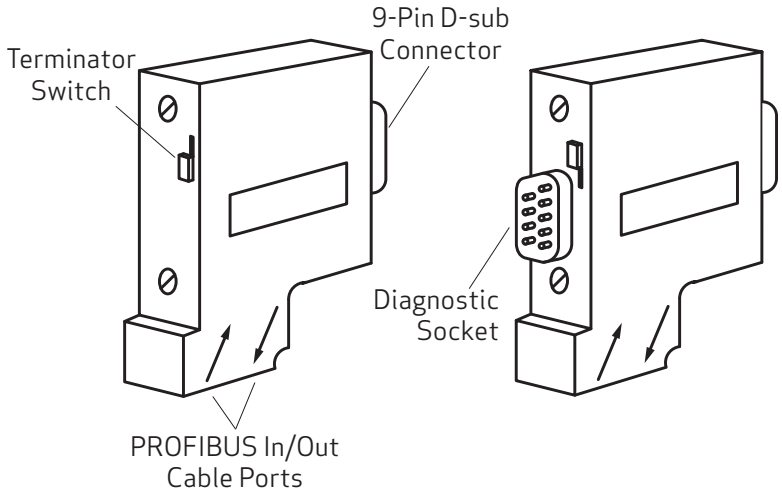


Cross-Section of Shielded PROFIBUS Cable

CAUTION: To minimize the possibility of electromagnetic interference (EMI), all connections should use *shielded* cables.

Example PROFIBUS Connector

The following figure shows an example of a PROFIBUS connector. These connectors allow you to have a cable input and output on one connector for daisy-chaining to other PROFIBUS devices. They also incorporate a termination switch. That switch must be activated (ON) at both ends of the bus. The switch must be deactivated (OFF) for the other devices along the bus.



Example of PROFIBUS Connectors

NOTE: At least one PROFIBUS connector with a diagnostic/programmer socket is required in each segment. These are for diagnostic and monitoring purposes only. They are not used to stack PROFIBUS connectors.

PROFIBUS Cable Diagram

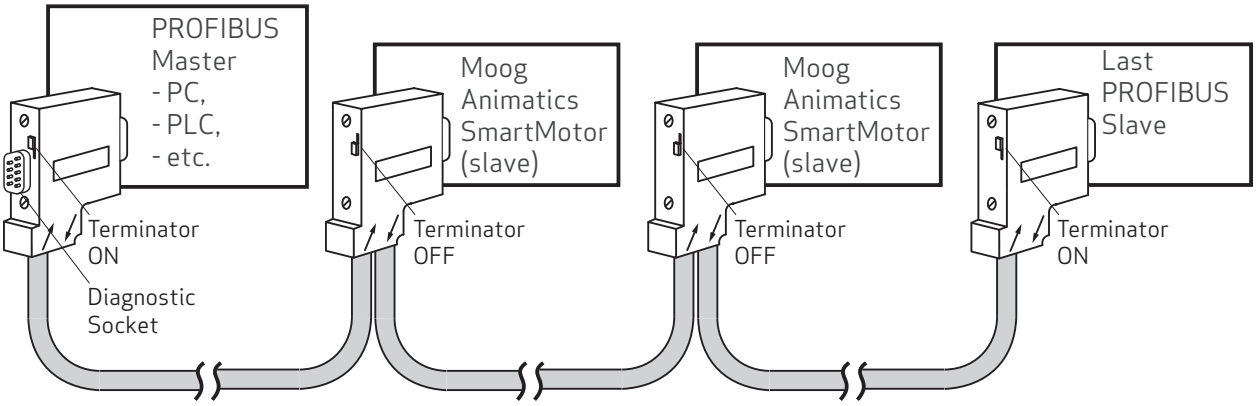
The following figure shows an in-line (daisy chain) network topology. For PROFIBUS network design and installation details, see the information available at:

http://www.profibus.com

The following diagram shows an example PROFIBUS network with the SmartMotors daisy chained to the master device.

CAUTION: PROFIBUS *requires* terminators at each end of the network bus.

PROFIBUS Cable Diagram



NOTE: The "incoming" cable connection must be used on the first and last connector of the system. Otherwise, the terminator switch will disconnect the device if it is wired to the "outgoing" connection.

NOTE: At least one PROFIBUS connector with a diagnostic/programmer socket is required in each segment. These are for diagnostic and monitoring purposes only. They are not used to stack PROFIBUS connectors.

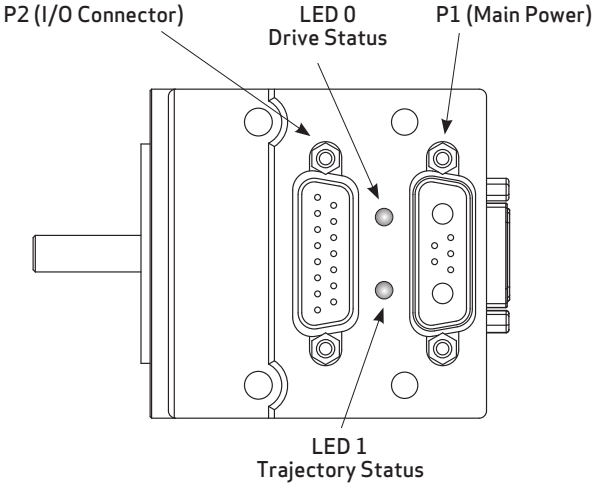
Cable Length

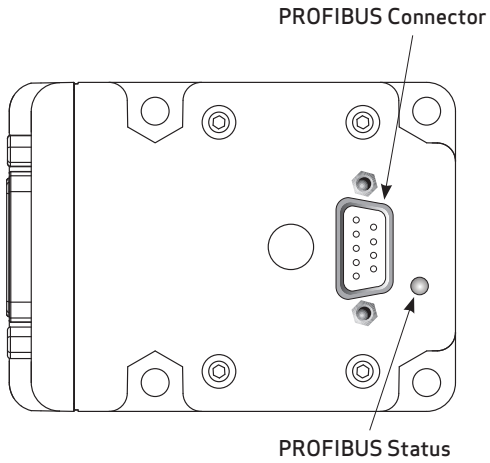
The PROFIBUS transmission speed determines the cable length. As noted previously, the minimum cable length when operating >=1 MBaud (or 1 Mbps) is 1 meter (~3 feet). If the cable is too short, reflected impedance can cause loss of communication data packets and spurious node errors.

At a speed of 1.5 Mbps, the maximum cable length is 200 meters; for speeds of 3, 6, and 12 Mbps, the maximum cable length is 100 meters. If longer cable lengths are needed, repeaters can be incorporated into the system.

PROFIBUS Status LEDs

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





LED 0: Drive Status Indicator

Off	No power
Solid green	Drive on
Blinking green	Drive off, no faults
Triple red flash	Watchdog fault
Solid red	Major fault
Alt. red/green	In boot load, needs firmware

LED 1: Trajectory Status Indicator

Off	Not busy
Solid green	Drive on, trajectory in progress

PROFIBUS Status Indicator

Off	No connection or improper configuration; verify that host is communicating to the MACID set for the motor
Green	Device is connected and exchanging data with host; host continues to send traffic to motor to keep watchdog active

LED Status on Power-up:

- With no program the travel limit inputs are low:
  LED 0 will be solid red indicating the motor is in a fault state due to travel limit fault.
  LED 1 will be off
- With no program and the travel limits are high:
  LED 0 will be solid red for 500 milliseconds and then begin flashing green.
  LED 1 will be off
- With a program that only disables travel limits and nothing else:
  LED 0 will be solid red for 500 milliseconds and then begin flashing green.
  LED 1 will be off

PROFIBUS Configuration

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

Configure Motor with PC **22**
User Program Requirements 22
Required Nonvolatile EEPROM Values 22
Reserved Motor Variables 22
Configure PLC with PC **23**
Configure SmartMotor to PROFIBUS **23**
PLC Sends Commands to Motor 23
Network Data Format Example 23
Status Bits for Class 5 Mode (Default) 24
Status Bits for Class 4 Emulation Mode 25
PLC Memory 25
Sequence to Set Report Data to Motor Clock 26
PROFIBUS Communication Example **27**

Configure Motor with PC

Use the following procedure to configure the SmartMotor for communication with the PC. Refer to the figures in PROFIBUS Communication Example on page 27.

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 5 PROFIBUS 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.

Use the following terminal commands to set the values:

- Set the PROFIBUS node ID to 3:

  CADDR=3

- Set the polling rate as fast as possible. In this command, the first number sets the poll rate, the second number is the number of microseconds to wait between polling loops.

  CANCTL(8,0)

- Set the action to take on network lost:

  CANCTL(9,0)

- Set the program label to jump to on network lost (if selected):

  CANCTL(6,0)

For the previous settings:

- CADDR= is mandatory, and the rest are optional. The defaults for the commands shown here are from the EEPROM; therefore, no assumptions should be made.
- CANCTL(8,0) by default EEPROM values would provide the fastest possible polling rate.
- CANCTL(9,0) will take no action on network lost
- CANCTL(6,0) depends on the user's program – only relevant when the "network lost" option is set to "GOSUB or GOTO a program label"

Reserved Motor Variables

The Class 5 PROFIBUS motor does not need to reserve any user variables as the Class 4 motor did. The variables yyy and zzz are not affected unless they are deliberately accessed by the user. EPTR may be

used by the user and is not modified by PROFIBUS activity as the Class 4 was. VST and VLD commands are also independent on PROFIBUS.

Configure PLC with PC

Use the following procedure to configure the PLC for communication with the PC. Refer to the figures in PROFIBUS Communication Example on page 27.

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 GSD file, set it up as a PROFIBUS node, from the catalog, and define the node number of the motor. For more details on the GSD file, see Software on page 9.
2. Set up the PLC memory that is the three words (six bytes) PROFIBUS output to the SmartMotor and the seven words (fourteen bytes) input from the SmartMotor.

Configure SmartMotor to PROFIBUS

CAUTION: PROFIBUS *requires* termination and bias resistors at each end of the bus. If the PROFIBUS connectors have terminator switches, make sure the terminator switches at both ends of the PROFIBUS bus are ON, and all other terminator switches are OFF. For details, see Cables and Diagram on page 17.

Use the following procedure to configure the SmartMotor to PROFIBUS. Refer to the PROFIBUS Status LEDs on page 20.

1. Plug the PROFIBUS connector into the motor.
2. Power up or power cycle the motor. For 2 seconds, the PROFIBUS LEDs on the motor in the proximity of the PROFIBUS connector will have meaningless states.
3. If the PLC is running, within a few seconds the PROFIBUS connection status LED on the motor will turn GREEN, indicating the PROFIBUS Controller has established a connection with the follower motor.

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 PROFIBUS and communicate with the motor.

The following are sequences of commands sent, which show all the intermediary PROFIBUS 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).

Status Bits for Class 5 Mode (Default)

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	User's status bit defined by CANCTL(12,x), see Nonvolatile Data on page 38
13	Command error (includes math and array errors)
14	Peak overcurrent occurred
15	Drive ready

Status Bits for Class 4 Emulation Mode

NOTE: CANCTL(11,1) enables Class 4 emulation mode; CANCTL(11,0) disables Class 4 emulation mode.

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	Math overflow
12	Array index error
13	Syntax error
14	Peak overcurrent occurred
15	Program checksum error

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 47 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

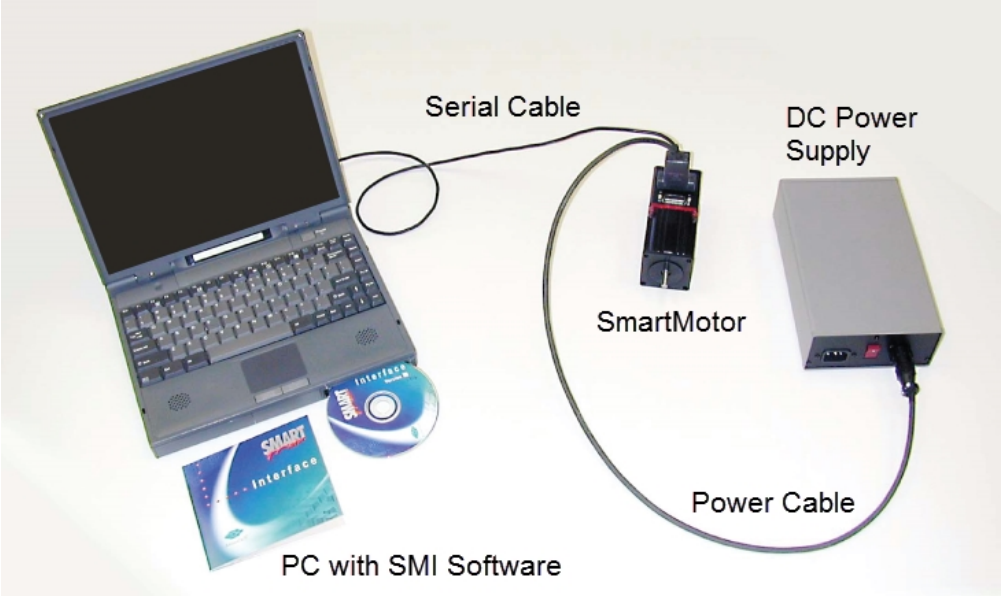
PROFIBUS Communication Example

The example illustrates communication over PROFIBUS by sending commands from a PLC over PROFIBUS to cause the motor to continually report its changing clock value to the PLC. The value displayed by the PLC registers containing the PROFIBUS data received from the motor changes as the updated clock value is received from the motor.

NOTE: Your motor type may vary from the motor pictured below.

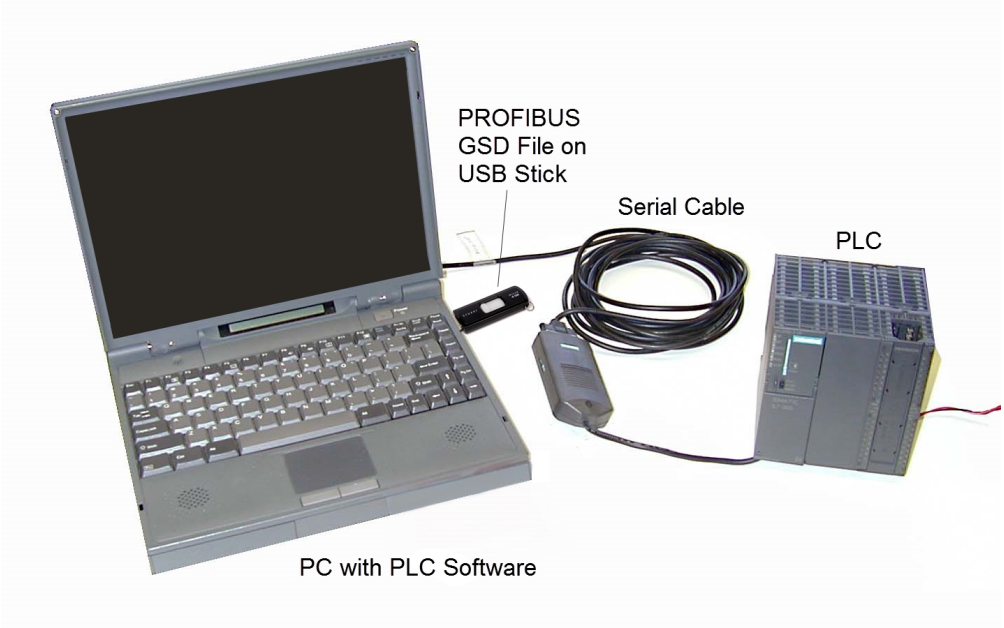
The create a PROFIBUS connection to the SmartMotor:

1. Install the SMI software. For more details, see the *Moog Animatics SmartMotor™ User's Guide*.
2. Refer to the following figure. Configure the motor through its serial port using a PC that is running the SMI software to:
    a. Set PROFIBUS node number in motor non-volatile storage.



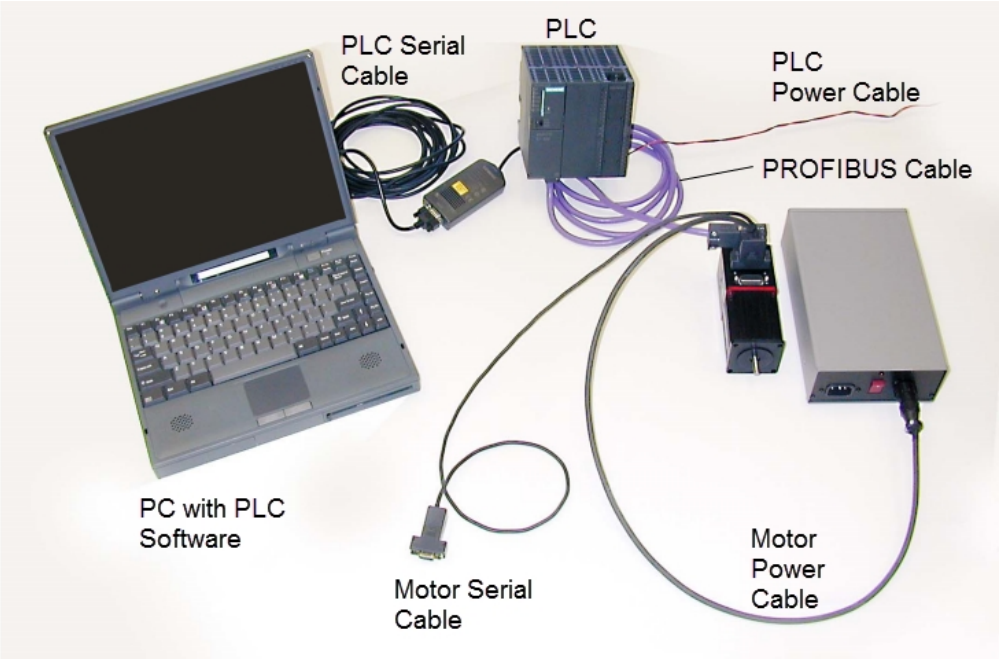
Configuring the SmartMotor

3. Refer to the following figure. Configure your PLC through its serial port using a PC that is running your PLC configuration software to:
    a. Load the motor's PROFIBUS GSD file
    b. Assign and display the PLC registers associated with the motor's PROFIBUS input and output data



Configuring the PLC

4. Refer to the following figure. Connect a PROFIBUS cable to the PLC and to the SmartMotor.



PROFIBUS Communication between PC, PLC and SmartMotor

5. Power cycle the motor to initialize the motor with the configured values.
6. Using a PC that is running the PLC software and the PLC online (see the previous figure):
    a. Enter the PROFIBUS motor command to report motor clock in the PLC PROFIBUS data registers.
    b. Watch the clock value being updated in the PLC PROFIBUS input registers.

For examples of sending command sequences and communication handshaking, refer to Sample Command Sequences on page 30.

Sample Command Sequences

This chapter contains sample PROFIBUS command sequences.

Overview **31**
Command and Response Codes 31
Handshaking of Messages 31
Disabling Limits from Preventing Motion 31
Turning the Motor Shaft 31
Disable Limits and Clear Fault Status **32**
Commands 32
PLC Memory 32
Disable positive limit, command EIGN(2) 32
Disable negative limit, command EIGN(3) 32
Clear fault status, command ZS 33
Initiate Mode Torque **34**
Commands 34
PLC Memory 34
Set torque value, specify the response data 34
Initiate torque mode, command MT 34
Initiate Relative Position Move **36**
Commands 36
PLC Memory 36
Set acceleration value, command ADT=255 36
Set maximum velocity value, command VT=100000 36
Make a relative position move 37

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 47 The symbolic command and response codes are listed, along with their values and the related SmartMotor™ command. See Output and Input Packets on page 40 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 40 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 32.

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 34
- Initiate Relative Position Move on page 36

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.

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 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.

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 controller to the follower 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 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:

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 controller to the follower 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 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.

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 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.

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 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.

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 velocity (100,000 in slew)	0001 86AD
status	0009
Bt = 1	
Bi = 1	
measured current position	0000 CA23
measured current position error	0011

Nonvolatile Data

The motor stores some information about the PROFIBUS network in EEPROM, and it must be initialized with the data shown in the following table. This is accomplished through serial channel 0 using the CANCTL (function, value) and CADDR= commands. After a connection to the PROFIBUS Controller had been established, it would be possible to modify these values through PROFIBUS. These commands are described in the PROFIBUS Packet Commands section of this manual. For details, see Output and Input Packets on page 40.

The Class 5 PROFIBUS interface is not a gateway expansion in the same way the Class 4 PROFIBUS expansion was. Therefore, these values are more integrated into the Class 5 motor with unique commands.

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

Command	Description/ Parameter	Values
CADDR=	PROFIBUS node (MACID)	Typically 3 to 127; 63 is the factory default.
CANCTL(1,0)	Reset PROFIBUS	Resets the PROFIBUS hardware and protocol stack.
CANCTL(6, <value>)	NET_LOST_LABEL	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. Sets u16NetLostLabelDefault and u16NetLostLabel.
CANCTL(7,<value>)	Axis identifier u32MotorId (not MACID)	A 32-bit value settable and readable through PROFIBUS response code 239. Not required for normal operation and may be ignored.
CANCTL(8, <value>)	POLL_RATE	Polling rate of the PROFIBUS service in microseconds. 0 is as fast as possible. Note that full microsecond resolution is not available, but the service time is typically under 1 millisecond. This setting allows for slowing down PROFIBUS service to allow more CPU time for user programs.
CANCTL(9,<value>)	NET_LOST_ACTION	Action to take if PROFIBUS network is lost. (Controller is no longer communicating to follower; timeout of master occurs in the SmartMotor.) <value>: 0 - IGNORE 1 - Send command OFF to motor 2 - Send command X to motor (soft stop) 3 - Send command S to motor (immediate stop) 4 - Send command GOSUB(x), where x is the value of NET_ LOST_LABEL 5 - Send command GOTO(x), where x is the value of NET_ LOST_LABEL Sets u8NetLostActionDefault and u8NetLostAction
CANCTL(10,<value>)	SET_PA_FIELD	Configure the PROFIBUS input packet to use an alternate data source for the "Measured position" field (words 4,5). <value>: 0 - Report actual position in encoder counts (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)

Command	Description/ Parameter	Values
CANCTL(11,<value>)	Class 4 emulation	Configure the motor to perform certain actions similar to Class 4. Value is reset to 0 (Class 5 mode) when motor is reset. This value is not stored in the EE. 0 - Behave like a Class 5 (power-on default) 1 - Perform certain actions like Class 4: Status word contains math error instead of communication error and array error, these are cleared by ZS. Status word index capture bit resets on index read, index re-armed by the read. Status word checksum error instead of drive ready. CMD_Zd works through PROFIBUS to clear math overflow. CMD_Zu works through PROFIBUS to clear array error.
CANCTL(12,<value>)	Control user status bit	When in Class 5 mode (Class 4 emulation disabled), it is possible to use bit 12 in the PROFIBUS status word as a user-controlled bit. The user can choose what purpose this is used for. <value>: 0- Clear bit 12 in the PROFIBUS status word 1- Set bit 12 in the PROFIBUS status word

Program Example

The following code example sets the nonvolatile station name.

```
a=<address>
IF CADDR!=a
  CADDR=a
  Z     'Reboot motor with new address
ENDIF
```

Output and Input Packets

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

Output and Input Packet Format **41**

Command (Output) Packet Notes **43**

Response (Input) Packet Notes **44**

Output and Input Packet Format

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			
4	8	Measured Position (32 bits), big-endian format 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.

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 Polling Rate (u32PollRate) is the minimum time (in microseconds) to wait between sending status requests, position requests, and requests associated with the Response Codes and commands. The microsecond scaling of this value is to support the higher-speed nature of the Class 5 PROFIBUS interface. It does not imply that there is a response time with a resolution of single microseconds. Setting this value to 0 polls the PROFIBUS interface as fast as possible, which is approximately 300-500 microseconds.
- The SmartMotor variable yyy is no longer used by the PROFIBUS module in the Class 5 motor, as compared to the Class 4 PROFIBUS interface. The Class 5 memory map also differs such that ab[200], aw[100] and al[50] do not share space with variable yyy. That means those array locations are also not affected by the PROFIBUS interface.
- The PROFIBUS 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.
- If an invalid value of the MACID (not 0-125) is found at startup, then the PROFIBUS Default Address of 126 will be used.

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.
- The Polling Rate (u32PollRate) is the minimum time (in microseconds) to wait between sending status requests, position requests, and requests associated with the Response Codes and commands. The microsecond scaling of this value supports the higher-speed nature of the Class 5 PROFIBUS interface. It does not imply that there is a response time with a resolution of single microseconds. When this value is set to 0, it polls the PROFIBUS interface as fast as possible, which is approximately 300-500 microseconds.
- 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 SmartMotor variable yyy is not used by the PROFIBUS module. Therefore, the user program may use variable yyy. In the memory map, ab[200], aw[100] and al[50] do not share space with variable yyy. The means that those array locations are also not affected by the PROFIBUS interface. This functionality may be unexpected by users familiar with older Moog Animatics products.
- The PROFIBUS 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 PROFIBUS interface, which may be unexpected by users familiar with older Moog Animatics products.
- If an invalid value of the MACID (not 0-125) is found at startup, then the PROFIBUS default address of 126 will be used.

Alternate Communications Channel

In addition to communicating over PROFIBUS, 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 PROFIBUS 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 PROFIBUS 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 PROFIBUS 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 PROFIBUS packet command and response codes and their corresponding SmartMotor commands.

Command Packet Codes to Motor Commands **47**

Response Packet Codes to Motor Commands **56**

Command Packet Codes to Motor Commands

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

Certain commands can have different meanings based on the SmartMotor style. For example, RBa will have the meaning associated with the style of the motor.

Variables beginning with u8, u16 or u32 are internal to the motor's PROFIBUS 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 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 6	EOBK(6)
1, x01	3, x03	C/D*	Direct brake to output number 2	EOBK(2)
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

Command Code	Command Data Value	Event for update	Command Description	Smart Motor Command(s)
decimal, hex	*decimal, hex*			
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
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, x26	C/D*	Divide PID sample rate by 2, default	PID2
1, x01	39, x27	C/D*	Divide PID sample rate by 4	PID4
1, x01	40, x28	C/D*	Divide PID sample rate by 8	PID8
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; if a brake redirect, this is ignored	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; if a brake redirect, this is ignored	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; if a brake redirect, this is ignored; 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; if a brake redirect, this is ignored; 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*	Available in Class 4 emulation mode only to reset math error	
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

Command Code	Command Data Value	Event for update	Command Description	Smart Motor Command(s)
decimal, hex	*decimal, hex*			
1, x01	66, x42	C/D*	Available in Class 4 emulation mode only to reset array index error	
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
1, x01	71, x47	C/D*	Make I/O 6 an input; if a brake redirect, this is ignored; 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; if a brake redirect, this is ignored;	EIGN(4)
1, x01	74, x4A	C/D*	Make I/O 5 an input; if a brake redirect, this is ignored;	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	C/D*	Immediately force trapezoidal commutation mode	MDT
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	N/A	(Reserved) MDC not available on this platform.	
1, x01	87, x57	C/D*	Turn on Trajectory Overshoot Braking (TOB) feature for trapezoidal mode	MDB
1, x01	88+, x58+	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>)
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 digital output 4 to 0 or 1	OUT(4)=<value>

Command Code	Command Data Value	Event for update	Command Description	Smart Motor Command(s)
decimal, hex	*decimal, hex*			
97, x61	N/A	N/A	(Reserved)	
98, x62	<value>	C/D*	Set digital output 5 to 0 or 1.	OUT(5)=<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>
151, x97	N/A	N/A	(Reserved)	
152, x98	<value>	C/D*	Set I/O A output	OUT(0)=<value>
153, x99	N/A	N/A	(Reserved)	
154, x9A	<value>	C/D*	Set I/O B output	OUT(1)=<value>
155, x9B	N/A	N/A	(Reserved)	
156, x9C	<value>	C/D*	Set I/O C output	OUT(2)=<value>

Command Code	Command Data Value	Event for update	Command Description	Smart Motor Command(s)
decimal, hex	*decimal, hex*			
157, x9D	N/A	N/A	(Reserved)	
158, x9E	<value>	C/D*	Set I/O D output	OUT(3)=<value>
159, x9F	N/A	N/A	(Reserved)	
160, xA0	<value>	C/D*	Set I/O G output	OUT(6)=<value>
161-162, xA1-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>
167-169, xA7-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> u8VarIndexSetActual = <value> where <value> represents which variable is referred to in the next variable write operation: '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> 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> u8ArrIndexSetActual = <value> where <value> is the array index to begin the next array write operation at. The ranges of <value> are as follows depending on the type of array: ab[]: 0-203 aw[]: 0-101 al[]: 0-50	
204, xCC		N/A	(Reserved)	
205, xCD	<value>	C/D*	u8ArrLenSet = <value> where <value> represents the quantity of array variables to write in the next array write operation. The ranges of <value> are as follows depending on the type of array: ab[]: 0-204 aw[]: 0-102 al[]: 0-51	
206, xCE	<value>	C/D*	u8AutoIncSet = <value> 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> u8VarIndexGetActual = <value> where <value> represents which variable is referred to in the next variable read operation: '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> 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> u8ArrIndexGetActual = <value> where <value> is the array index to begin the next array read operation at. The ranges of <value> are as follows depending on the type of array: ab[]: 0-203 aw[]: 0-101 al[]: 0-50	
211, xD3		N/A	(Reserved)	
212, xD4	<value>	C/D*	u8ArrLenGet = <value> where <value> represents the quantity of array variables to read in the next array read operation. The ranges of <value> are as follows depending on the type of array: ab[]: 0-204 aw[]: 0-102 al[]: 0-51	
213, xD5	<value>	C/D*	u8AutoIncGet = <value> 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> ='a'+u8VarIndexSetActual; if (u8AutoIncSet) then: u8VarIndexSetActual += 1	<a to zzz>=<value>
215, xD7	<value>	C/D*	Set byte array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then: u8ArrIndexSetActual += 1	ab[<index>]=<value>
216, xD8	<value>	C/D*	Set word array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then: u8ArrIndexSetActual += 1	aw[<index>]=<value>
217, xD9	<value>	C/D*	Set long array variable <index>=u8ArrIndexSetActual; if (u8AutoIncSet) then: u8ArrIndexSetActual += 1	al[<index>]=<value>
218, xDA	<value>	C/D*	Store byte to EEPROM u32EptrActual += 1 NOTE: This u32EptrActual is not the same as the program EPTR= command.	VST(<value byte>,1) (But does not affect EPTR or variables.)
219, xDB	<value>	C/D*	Store word to EEPROM u32EptrActual += 2 NOTE: This u32EptrActual is not the same as the program EPTR= command.	VST(<value word16>,1) (But does not affect EPTR or variables.)
220, xDC	<value>	C/D*	Store long to EEPROM u32EptrActual += 4 NOTE: This u32EptrActual is not the same as the program EPTR= command.	VST(<value long>,1) (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
			<a to z>='a'+u8VarIndexSetActual
			u32EptrActual += 4;
			if (u8AutoIncSet) then:
			u8VarIndexSetActual += 1
			NOTE: This u32EptrActual is not the same as the program EPTR= command.
				<a to z>=<value>
				VST(<a to z>,1)
				(does not affect EPTR)

222, xDE	N/A	C/D*	Store variables to EEPROM
			<a to z>='a'+u8VarIndexSetActual
			<length>=u8VarLenSet
			u32EptrActual += (<length>*4);
			if (u8AutoIncSet) then:
			u8VarIndexSetActual += <length>
			NOTE: This u32EptrActual is not the same as the program EPTR= command.
				VST(<a to z>, <length>)
				(does not affect EPTR)

223, xDF	N/A	C/D*	Store byte array variables to EEPROM
			<index>=u8ArrIndexSetActual
			<length>=u8ArrLenSet
			u32EptrActual += (<length>*1);
			if (u8AutoIncSet) then:
			u8ArrIndexSetActual += <length>
			(This u32EptrActual is not the same as the program EPTR= command.)
				VST(ab[<index>], <length>)
				(does not affect EPTR)

224, xE0	N/A	C/D*	Store word array variables to EEPROM
			<index>=u8ArrIndexSetActual
			<length>=u8ArrLenSet
			u32EptrActual += (<length>*2);
			if (u8AutoIncSet) then:
			u8ArrIndexSetActual += <length>
			(This u32EptrActual is not the same as the program EPTR= command.)
				VST(aw[<index>], <length>)
				(does not affect EPTR)

225, xE1	N/A	C/D*	Store long array variables to EEPROM
			<index>=u8ArrIndexSetActual
			<length>=u8ArrLenSet
			u32EptrActual += (<length>*4);
			if (u8AutoIncSet) then:
			u8ArrIndexSetActual += <length>
			(This u32EptrActual is not the same as the program EPTR= command.)
				VST(al[<index>], <length>)
				(does not affect EPTR)

226, xE2	<value>	C/D*	Set the EEPROM address u32EptrSet=<value> u32EptrActual=<value> (doesn't affect EPTR)	
227, xE3	N/A	N/A	(Reserved)	
228, xE4	<value>	C/D*	Set GOTO/GOSUB number for net lost action u16NetLostLabel=<value> (initialized to the value of u16NetLostLabelDefault during power-up) Range is: 0 - 999	See CANCTL(...)

Command Code	Command Data Value	Event for update	Command Description	Smart Motor Command(s)
decimal, hex	*decimal, hex*			
229, xE5	<value>	C/D*	Set PROFIBUS network lost action u8NetLostAction=<value> (initialized to the value of u8NetLostActionDefault during power-up)	See CANCTL(...) Upon loss of communication with PROFIBUS 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, xE6	<value>	C/D*	Set rate (in us) at which the SmartMotor is polled by PROFIBUS u32PollRate=<value> (saved in nonvolatile storage in the SmartMotor, loaded at power-up.) Range is: 0 to 2047907 (approx. 2 seconds)	CANCTL(8,<value>)
231, xE7	<value>	C/D*	Set GOTO/GOSUB number for net lost action u16NetLostLabelDefault=<value> (saved in nonvolatile storage in the SmartMotor) Range is: 0 - 999	See CANCTL(...)
232, xE8	<value>	C/D*	Set PROFIBUS network lost action u8NetLostActionDefault=<value> (saved in nonvolatile storage in the SmartMotor)	See CANCTL(...) Upon loss of communication with PROFIBUS 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
233, xE9	<value>	C/D*	Set PROFIBUS address on network	CADDR=<value>
234-239, xEA-xEF	N/A	N/A	(Reserved)	
240, xF0	<value>	C/D*	Configure the PROFIBUS 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)
241-253, xF1-xFD	N/A	N/A	(Reserved)	
254, xFE	N/A	C/D*	Restores all nonvolatile settings to the following defaults: u16NetLostLabelDefault=3 u8NetLostAction=IGNORE CADDR=0 u32PollRate=0	
255, xFF	N/A	N/A	(Error) 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.

Response Packet Codes to Motor Commands

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

Certain commands can have different meanings based on the SmartMotor style. For example, RBa will have the meaning associated with the style of the motor.

Variables beginning with u8, u16 or u32 are internal to the motor's PROFIBUS 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	cyclic	<value>=	analog input number 4	RINA(A,4)
98, x62	cyclic	<value>=	digital I/O number 5	RIN(5)
99, x63	cyclic	<value>=	analog input number 5	RINA(A,5)
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 (math error if Class 4 emulation mode is enabled.)	
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

Response Code	Event for update	Response Data Value	Response Description	Smart Motor Command(s)
decimal, hex				
121, x79	cyclic	<value>=	hardware index input level	RBx
122, x7A	cyclic	<value>=	millisecond clock	RCLK
123, x7B	cyclic	<value>=	secondary counter	RCTR(1)
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; when Class 4 emulation mode, this re-arms the capture	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 Requires firmware 5.32.4.51 or later	RUIA
146, x92	cyclic	<value>=	servo bus voltage Requires firmware 5.32.4.51 or later	RUJA
147, x93	N/A	0	(Reserved)	
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	cyclic	<value>=	analog input number 2	RINA(A,2)
158, x9E	cyclic	<value>=	digital I/O number 3	RIN(3)

Response Code	Event for update	Response Data Value	Response Description	Smart Motor Command(s)
decimal, hex				
159, x9F	cyclic	<value>=	analog input number 3	RINA(A,3)
160, xA0	cyclic	<value>=	digital I/O number 6	RIN(6)
161, xA1	cyclic	<value>=	analog input number 6	RINA(A,6)
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>=	I/O 0-6, 7-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)
179-181, xB3-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)
187-199, xBB-xC7	N/A	0	(Reserved)	
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	
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	

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

Response Code	Event for update	Response Data Value	Response Description	Smart Motor Command(s)
decimal, hex				

222, xDE	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>)
				
				(does not affect EPTR)

223, xDF	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>)
				
				(does not affect EPTR)

224, xE0	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>)
				
				(does not affect EPTR)

225, xE1	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>)
				
				(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 PROFIBUS 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 CANCTL(...)	

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) See CANCTL(...)	On loss of communication with PROFIBUS 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, xE6	cyclic	<value>=	u32PollRate Rate at which SmartMotor is polled by PROFIBUS (saved in nonvolatile storage in the SmartMotor) Set by CANCTL(8,<value>)	
231, xE7	cyclic	<value>=	u16NetLostLabelDefault (saved in nonvolatile storage in the SmartMotor) See CANCTL(...)	
232, xE8	cyclic	<value>=	u8NetLostActionDefault (saved in nonvolatile storage in the SmartMotor) See CANCTL(...)	On loss of communication with PROFIBUS 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
233, xE9	cyclic	<value>=	PROFIBUS address on network	RCADDR
234, 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, xEF	cyclic	<value>=	u32MotorId (read at startup from SmartMotor EEPROM) Set by CANCTL(7,<value>)	
240-254, 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.

Troubleshooting

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

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

Issue	Cause	Solution
PROFIBUS Communication Issues		
NOTE: Auto-detect of master's PROFIBUS baud rate may take 5 to 10 seconds after SmartMotor power up. PROFIBUS master repeatedly sends the "are you there?" command packet. Follower motor does not transmit on PROFIBUS unless commanded to answer by the master.		
No PROFIBUS connection. The PROFIBUS connection status LED is off after power up and while connected to the master.	Motor not powered. Drive Status LED is off and/or PROFIBUS status LED is off.	Check Drive Status LED. If LED is not lit, check wiring. If only the PROFIBUS status LED is off, connection to PROFIBUS Host, or watchdog expired. Verify that host is communicating to the MACID set for this motor.
	Disconnected or miswired connector, or broken wiring between follower and master.	Check that connectors are correctly wired and connected to motor. For details, see PROFIBUS Motor Connectors and Pinouts on page 16. Check that oscilloscope shows good square 3v signals. Check that PROFIBUS A and B signals are not reversed.
	Motor nonvolatile settings.	Check that motor MACID has been set to the value expected by the host PLC. Check that polling rate is set to 0, which allows fastest response.
	Wrong type of cable.	Check that cable is a PROFIBUS cable. For details, see Cables and Diagram on page 17.
	Wrong GSD file.	Verify that the correct GSD file was used to configure the master and connect the follower motor as part of the PROFIBUS network.
	Terminator and bias resistors are not correct.	Check that terminators are at each end of the bus, and each terminator's total resistance is 110 ohms. If using PROFIBUS connectors with built-in terminators, check that the switches are ON at each end of the bus only, and all other terminator switches on the bus must be OFF.
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
	Byte order of command / response code or data is wrong.	Check that the byte-order parameter is set as intended.
	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 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.

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 5 SmartMotor™ PROFIBUS Guide, Rev. E
SC80100009-001