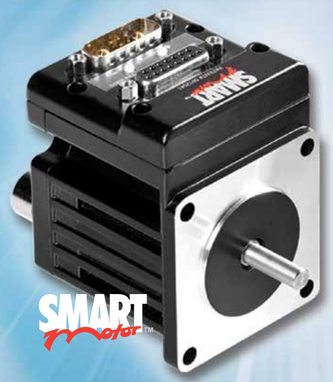
Quick Start Guide

for Class 5 SmartMotor™



For complete installation and programming details, see the *Class 5 SmartMotor™ Installation and Startup Guide* and the *SmartMotor™ Developer's Guide*.

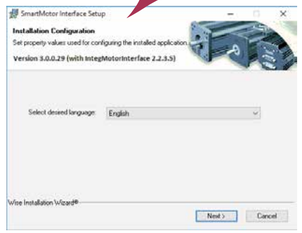
MOOG ANIMATICS

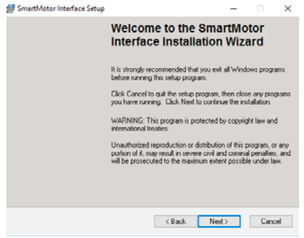
SMI* Introduction

Install the SMI Software

SMI_3_x.msi

Download SMI from **animatics.com/smi**, then follow the on-screen instructions to complete the installation.

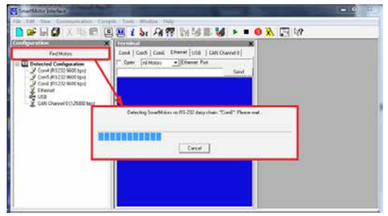




*SMI: SmartMotor™ Interface

Establish Communication

To establish communication between the PC and SmartMotor, use the Find Motors button.

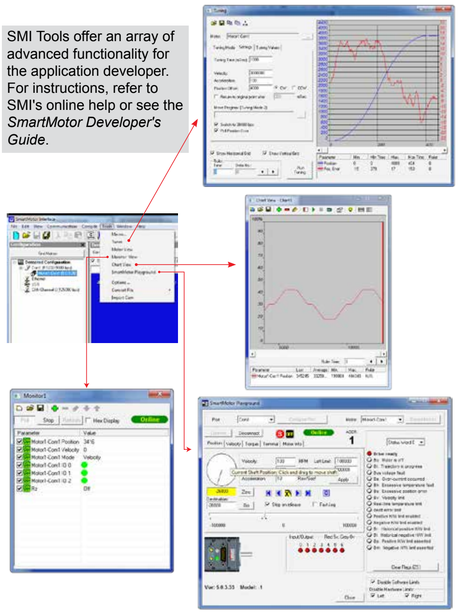


If communication is successful, the motor icon appears, indicating the hardware connections and software installation are correct.

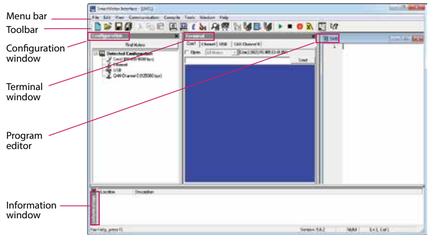
Double-click the motor icon. The Motor View shows key information, including position, firmware version and model.

SMI Tools

SMI Tools offer an array of advanced functionality for the application developer. For instructions, refer to SMI's online help or see the *SmartMotor Developer's Guide*.



SMI Standard Control Interface



Menu bar: All of the windows and functions of the SMI software can be accessed through the menu bar. Many of these are also accessible through the icons on the toolbar.

Toolbar: Used for accessing the primary features of the SMI software. Depending on the current state of the SMI software and the currently active window, some toolbar buttons may be disabled.

Configuration window: Displays all connections to PC, all motors connected to PC, and status on all motors.

Terminal window: Used to directly access real-time command and control of the motor.

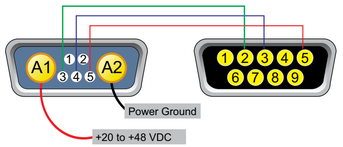
Program editor: Used to manage, edit and print the user program that will be sent to the motor.

Information window: Display the results of user operations.

Basic Wiring

Single RS-232 Communication

SmartMotor Combo Connector **PC RS-232 Connector**



7 Pin Combo D-Sub Connection

1 I/O – 6 GP, Index Input or "G" Command; For -CDS7, CAN-L only
2 +5 VDC Out; For -CDS7, CAN-H only
3 RS-232 Transmit (Tx)
4 RS-232 Receive (Rx)
5 Common Ground (typ. SIG Ground)
A1 Main Power: +20-48 VDC
A2 Common Ground (req'd. POWER Ground)

9 PIN RS-232 Connection

2 RS-232 Receive (Rx)
3 RS-232 Transmit (Tx)
5 RS-232 Ground (Gnd)

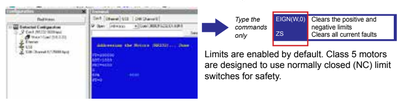
NOTE: Do not reverse the positive and negative leads from the power supply.

High-speed operation of the application requires 48 VDC input.

Basic Operating Instructions

Send Commands to the Motor

Use the Terminal window to enter single commands.



Limits are enabled by default. Class 5 motors are designed to use normally closed (NC) limit switches for safety.

The following commands can be entered in the Terminal window. **NOTE:** Comments on white background are for information only and cannot be typed.

Terminal Window - Sending Control Commands to the Motor

Position Mode introduced	
MP	Set Position Mode (default power-up mode)
ADT=100	Set acceleration to 100 (1 rev / sec ^ 2 = 4, default is 0)
VT=32768	Set speed to 32768 (1 rps = 32768 = 60 rpm, default is 0)
PRT=4000	Set the relative position of 4000 (1 rev = 4000, default is 0)
G	Start motion
RPA	Returns the absolute position of the motor
O=0	The current position set to 0
PT=40000	Set the absolute position of 40000 (10rev)
VT=327680	Set speed to 327680 (10rev/s)
G	Start motion
RPA	Returns the absolute position of the motor
Velocity Mode introduced	
MV	Set Velocity Mode
G	Start motion
VT=VT/2	Set half the rate of speed
G	Start motion
VT=-VT	Motor reverse
G	Start motion
X	Motor decelerates to a stop
Torque Mode introduced	
MT	Set Torque Mode
T=1600	Set the torque to ~5% (range +/- 32767)
G	Start motion
T=-T	Motor reverse
G	Start motion
T=T*2	Double the commanded torque
G	Start motion
X	Motor decelerates to a stop
Variables introduced	
a=123	Set the variable a value to 123
Ra	Returns the value of a variable
b=a+2	Set variable b value to the variable a plus 2
Rb	Returns the value of variable b
OFF	Turn off power to the motor coils and terminate motion
Z	Total reset, equivalent to power off and then on

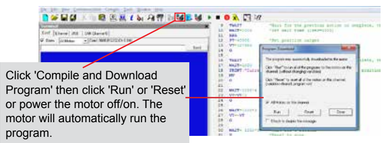
Enter the following in the Program Editor window.

Simple Motion Control Program

```
EIGN(W,0)                 'Disable hardware limits
ZS                        'Clear all current faults

MP                        'Set position mode
ADT=100                   'Set accel/decel target
VT=32768                  'Set velocity target
PRT=4000                  'Set relative position target
G                         'Start motion

TWAIT                     'Wait for the previous action to complete, then continue
WAIT=1000                 'Set wait time (1sec=1000)
RPA
PT=40000                  'Set position target
VT=327680                 'Set velocity target
G

TWAIT                     'Wait for the previous action to complete, then continue
WAIT=1000
PRINT("Current Position=",PA,#13)   'Show the actual position, then go to the next line
MV                        'Set velocity mode
G

WAIT=1000*4               'Wait for 4 seconds
VT=VT/2
G

WAIT=1000*3               'Wait for 3 seconds
VT=-VT
G

WAIT= 1000*3              'Wait for 3 seconds
X                         'Decel to stop
TWAIT                     'Wait for the previous action to complete, then continue
MT                        'Set torque mode
T=1601                    'Set torque
G

WAIT=1000*3               'Wait for 3 seconds
T=1761                    'Set torque
G

WAIT=1000*3               'Wait for 3 seconds
T=2242                    'Set torque
G

WAIT=1000*3               'Wait for 3 seconds
T=1281                    'Set torque
WAIT=1000*3               'Wait for 3 seconds
X                         'Decel to stop
END                       'Program end
```

More programs are available at **animatics.com/sample-programs**

Download Program to the SmartMotor



Click 'Compile and Download Program' then click 'Run' or 'Reset' or power the motor off/on. The motor will automatically run the program.

I/O Control and Application

I/O Configuration Instructions

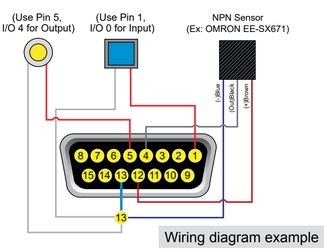
The SmartMotor has seven pins (I/O 0 - 6) to provide digital input (DI), digital output (DO) and analog input (AI); the following is a description of the SmartMotor 15-pin I/O:

15 PIN D-Sub I/O

1 I/O – 0 GP or Enc A or Step Input	10 RS-232 Transmit; For -CDS/7, CAN-L only
2 I/O – 1 GP or Enc B or Direction Input	11 RS-232 Receive; For -CDS/7, CAN-H only
3 I/O – 2 Positive Over Travel or GP	12 +5 VDC Out
4 I/O – 3 Negative Over Travel or GP	13 Common Ground (typ. SIG Ground)
5 I/O – 4 GP, IIC or RS-485 A (Com ch. 1)	14 Common Ground
6 I/O – 5 GP, IIC or RS-485 B (Com ch.1)	15 Main Power: +20-48 VDC; if DE option, Control Power separate from Main Power
7 I/O – 6 GP, Index Input or "G" Command	
8 Phase A Encoder Output	
9 Phase B Encoder Output	

NOTES:
- I/O 2 – Positive overtravel limit (for the CW direction of motor shaft)
- I/O 3 – Negative overtravel limit (for the CCW direction of motor shaft)
- I/O 6 – Enables motor movement (equivalent to the G command)
- I/O ports input impedance = 5 kohm (5 kohm pull-up resistor)
- GP I/O 0-6 are 25 mAmp Sink or Source, 10 Bit 0-5 VDC A/D

I/O Configuration Example



Wiring diagram example

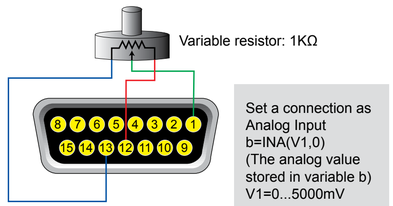
I/O Control Program Example

I/O Applications Introduced

```
EIGN(W,0)                 'Disable hardware limits
ZS                        'Clear all current faults

OR (4)                    'Reset I/O port 4 output OFF (0V)

GOSUB (1)                 'Execute C1 'Move to sensor' subroutine

WHILE 1                   'Logic loop (to determine the value 1 for infinite loop)
    IF IN (0)==0          'Logic test to see if the condition is met, if so,
                          '    execute the code within
        VT=500000
        ADT=100
        G
        WAIT=1000         'Pause 1 second
        GOSUB (2)         'Execute C2 'Toggle I/O port output' subroutine

        VT=-VT/2          'Reverse direction and run at 1/2 previous velocity
        ADT=100
        G
        WAIT=1000         'Pause 1 second
        GOSUB (2)         'Execute C2 'Toggle I/O port output' subroutine

    ENDIF                 'End of the IF structure

LOOP                      'End of the WHILE loop

END

'------------------------ 'Move to sensor' subroutine ----------------------
C1                        'Motor will move in one direction until input 1 goes high
MV                        'Velocity Mode
ADT=100                   'Set accel and decel target
VT=32767                  'Set velocity target
OUT(3)=0                  'Set output 3 low to indicate motion
G                         'Start moving

WHILE IN(1)==0            'While input 1 is low, keep moving
LOOP

OUT(3)=1                  'Set output 3 high to indicate stop
X                         'Decelerate to a stop

WHILE IN(1)==1            'While input 1 is high, don't move
LOOP

OUT(3)=0                  'Set output 3 low to indicate motion
G                         'Start moving again

RETURN

'------------------------ 'Toggle I/O port output' subroutine ----------------------
C2
OS (4)                    'Set I/O port 4 output ON (5V)
WAIT=2000                 'Pause 2 seconds
OR (4)                    'Reset I/O port 4 output OFF (0V)
WAIT=2000                 'Pause 2 seconds
RETURN
```

I/O Control and Application

Analog Input Application Example

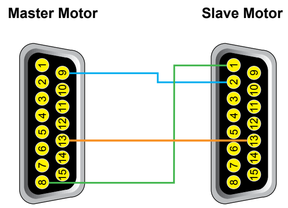


Set a connection as Analog Input
b=INA(V1,0)
(The analog value stored in variable b)
V1=0...5000mV

Changes in Motor Position Using Analog Input

```
VT= 100000        'Set speed
ADT=100           'Set acceleration
WHILE 1           'Loop (1 for infinite loop)
    b=INA(A,0)    'Declare I/O for the Analog
                  'input, and receive value into
                  'variable b
                  'A=0...32767

    PT=b          'From variable b, set the
                  'position target value
    G             'Execute action
LOOP              'Loop
END               'Program end
```

Motor to Follow the Movement and the Use of Electronic Gearing



Master Encoder outputs on pins 8 and 9, Slave receives external Encoder signals on pins 1 and 2.

Mode Follow with Ratio (Electronic Gearing)

```
'(Shows use of MF0, MFDIV and MFMUL)

EIGN(W,0)       'Make all onboard I/O general inputs;
                'disable travel limits
ZS              'Clear all current faults
MF0             'Reset CTR(1)
MFDIV=-10       'Divisor = -10
MFMUL=21        'Multiplier = 21
MFR             'Calculate Ratio, input -10 external counts
                'Resulting motion 21 counts
G               'Start following external encoder
END
```

Application of the Basic Program Flow

Basic Logic Structures

IF, ELSEIF, ELSE, ENDIF structure:

```
IF a<b
    PRINT ("a is less than b", #13)
ELSEIF q==123
    PRINT ("q equals 123", #13)
ELSE                    'if no condition above was true
    PRINT ("nothing above was true", #13)
ENDIF
```

WHILE, LOOP structure:

```
a=0
WHILE a<10
    a=a+1
LOOP
PRINT ("loop code executed 10 times", #13)
```

GOTO, GOSUB structure:

```
C1
    IF a>b
        GOTO1
    ELSEIF b>c
        GOSUB5
    ENDIF
    GOTO6
C5
    PRINT ("b is greater than c", #13)
RETURN
C6
END
```

SWITCH, CASE, BREAK structure:

```
SWITCH v
        CASE 1
                PRINT(" v = 1 ",#13)
        BREAK
        CASE 2
                PRINT(" v = 2 ",#13)
        BREAK
        CASE 3
                PRINT(" v = -23 ",#13)
        BREAK
        DEFAULT
                PRINT(" v IS NOT 1, 2 OR -23",#13)
        BREAK
ENDS
```

Torque Control Mode

```
MT                  'Enter Torque Mode
T=16000             'Set torque value
G                   'Go
WAIT=500            'Give the motor time to start moving

WHILE VA > 5000     'While moving faster than 1 RPM
LOOP

MP                  'Enter Position Mode
ADT=100             'Set Accel/Decel Value
VT=3200             'Set Velocity
PT=-400             'Set target position
G                   'Go

TWAIT               'Wait for move to complete
PRINT("Move complete",#13)

END
```

Watchdog for Serial Communication Based off Interrupts and TMR Command

```
'Setup portion of code
'Set up watchdog timer: Interrupt 0, Word 4, Bit 0, trigger low, subroutine 88
ITR(0,4,0,0,88)

'Start the timer 0 for 1 second
TMR(0,1000)

'Turn on the interrupt
EITR(0)
ITRE
'=============================================
'Main program loop
WHILE 1     'Forever
    '[Stuff happens here]
LOOP
END         'End of program
'=============================================
'Interrupt subroutine
C88
            DITR(0)
            S           'Stops motors abruptly
            WHILE ABS(VA)>0 LOOP        'While velocity is not 0
            WAIT=100    'Just to make sure the system is settled

            WHILE 1     'To keep the program in this error mode
                OFF     'Freewheel
            LOOP
RETURNI
```

More programs are available at animatics.com/sample-programs

Multi-Axis Control Instructions

Type the following commands in the Terminal window.

NOTE: Comments on white background are for information only and cannot be typed.

Terminal Window - Sending Commands to Multiple Axes

0EIGN(W,0)	All motors, set local I/O in word 1 as gen-use, disable travel limits
0ZS	All motors, clear all current faults
0MP	All motors, set to position mode
0ADT=500	All motors, set accel/decel to 500 (4 = 1 rev / sec ^ 2)
0VT=32768	All motors, set velocity target 32768 (32768 = 1rps = 60 rpm)
0PRT=4000	All motors, set position relative target 4000 (4000 = 1rev)
0G	All motors, start motion
1RPA	Motor 1, report the position
2RPA	Motor 2, report the position
0O=0	All motors, set current position to 0
1MV	Motor 1, set to velocity mode
1VT=300000	Motor 1, set velocity target 300000
1ADT=25	Motor 1, set accel/decel to 25
1G	Motor 1, start motion
1X	Motor 1, slow motion to stop
1VT=-100000	Motor 1, set velocity target -100000
1G	Motor 1, start motion
1X	Motor 1, slow motion to stop
2MT	Motor 2, set to torque mode
2T=1600	Motor 2, set torque 1600
2G	Motor 2, start motion
2X	Motor 2, slow motion to stop
2T=-1280	Motor 2, set velocity target -1280
2G	Motor 2, start motion
2X	Motor 2, slow motion to stop

READ THIS BEFORE UNPACKING MOTOR

MOOG ANIMATICS

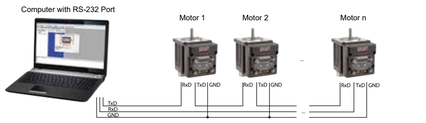
Failure to follow these safety instructions can result in damage to the motor and hazards to personnel.

1. **AVOID HOT PLUGGING:** Never connect the motor under voltage (hot plug). Hot plugging can cause damage to the motor and other electrical components.
2. **OBSERVE POLARITY:** Always ensure that the correct polarity (+ / -) is used when connecting the motor to its power source.
3. **APPLY CONTROL POWER FIRST:** For DE and M-Style motors, always connect and activate control power first and then drive power. Always power down in reverse order.
4. **USE SHUNTS:** Shunts are needed to protect the servo controller and drive stages from overvoltage, which originates from:
   - Back EMF due to back driving the motor
   - Hard stop crashes
   - Sudden or hard decelerations
   - Vertical load drops
5. **READ THE GUIDES:** Alway read the installation, programming and related guides before attempting to install or program the motor. See www.animatics.com/manuals
6. **NO TOOLS:** Never use tools to tighten M-style connectors – they must be finger tightened only! Use of a tool can cause overtightening of the connection, which may damage it and will void the warranty.

www.animatics.com

Multi-Axis Control Wiring and Addressing

RS-232 Series



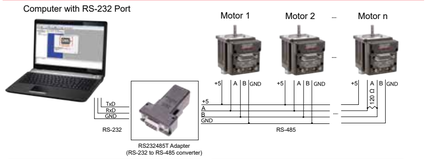
9 Pin RS-232

2 RS-232 Receive (Rx)
3 RS-232 Transmit (Tx)
5 RS-232 Ground (Gnd)

7 Pin Combo D-Sub

A1 +20 to +48 VDC
A2 Power Ground
1 I/O 6; For -CDS7, CAN-L only
2 +5V Out; For -CDS7, CAN-H only
3 RS-232 Transmit (Tx)
4 RS-232 Receive (Rx)
5 RS-232 Ground (Gnd)

RS-485 Connection



RS232485T Adapter

1 +5 VDC
2 GND
3 RS-485A
4 RS-485B

15 Pin D-Sub

12 +5 VDC out
13 Ground
5 I/O 4
6 I/O 5

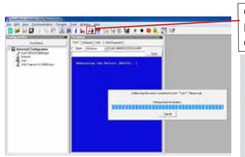
* RS-485A connected to I/O 4, RS-485B connected to I/O 5.
* Adapters provided by Moog Animatics have built-in biasing resistors. However, extensive networks should add bias at the very last motor in the chain.
* Proper cabling would include a shielded twisted pair to minimize transmission interference.
* RS232485T adapter's +5 VDC power requirement should be provided by an external (non-motor) power supply.

Addressing

Objective: To be able to communicate instructions to the specified motor, each motor must first have a unique address within the whole system.
Methods: SmartMotor provides two ways to define the motor address:

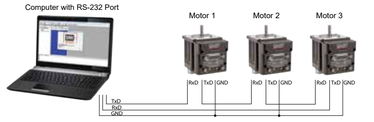
1. Automatic Addressing (Auto-Addressing): after each reboot, the system automatically sets the motor's address based on its order in the serial chain.
2. Pre-Addressing: The motor's address is stored within each motor's user program.

Auto-Addressing

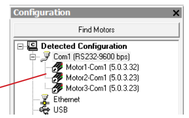


On the Toolbar, click (Find Motors) or Find Motors in the Configuration window.

The computer determines the order based on the motor's position in the serial chain — the closest to the computer is motor 1, followed by motor 2, etc. **NOTE:** If some are pre-addressed, you can re-address all motors, which may change the existing motor addresses.



When completed, the system displays the detected motors, their assigned addresses and firmware versions as shown.



Pre-Addressing

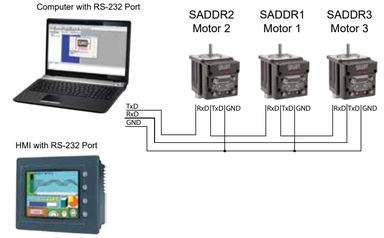
Enter the following commands into the motor's user program to have the SmartMotor self address:

```
SADDRn   'Pre-motor address, where n  = 1,2,3 ...
ECHO     'Open response (see NOTE below)
END
```

NOTE: For RS-232 multi-axis, all of the motor responses must be open (ECHO). For RS-485 multi-axis, all of the motor responses must be closed (ECHO_OFF).

Restart the power supply (Reset). After the motors have powered on, click (Detect Motors). SMI finds the pre-addressed motors and displays them in the Configuration window.

NOTE: If some motors are addressed and some are not, then you can choose to re-address all motors. This will probably change the address of any previously addressed motor.



Common SmartMotor Commands

Quick Reference for Frequently Used Commands

COMBITRONIC **RED TEXT** commands optionally support Combitronic™ syntax, which requires "-C, -DE, -CDS, -CDS7" or "CANopen or DeviceNet" product option.

REFERENCE KEY:
- is the I/O Bit Number
m - is the mask value of which bits are affected
W - defines it as a word (16 bits)
expression - an expression must contain no more than a total maximum of 32 operators, values, and parenthesis.
value - a number, variable or math expression with one operand
constant - means a fixed integer
gen# Trajectory generator number: 1 or 2
i - Interrupt number, valid values are from 0 to 7

COMMUNICATION COMMANDS:
CCHN(RS2,0) Close communication channel command
GETCHR Get the next character from channel 0
OCHN(RS2,0,N,9600,1,8,C,1000) Default: (RS-232,chan=0, no parity, 9600 baud,1 stop bit, 8 data bits, command,1000 ms timeout)
PRINT("Hello World",#13) Print command to say "Hello World", see print section for more detailed examples

PROGRAM FLOW COMMANDS:
CASE expression Switch case statement
C constant Subroutine label, e.g., C10 for subroutine 10, must have a RETURN for each C label
DEFAULT Default action for switch case statement
DITR(i) Individual interrupt disable
EITR(i) Individual interrupt enable
ELSEIF expression Used for IF statements to test another condition, if expression is true, then execute code
END End program execution
ENDIF End statement for IF code structures
ENDS Command for end of switch case statement
GOSUB(value) Call a subroutine, value up to 999
GOTO(value) Jump program execution to a label, value up to 999
IF expression Conditional Test, expression can be multiple math operations
ITR(i, status_wrd#, bit#, s, label#) Interrupt setup
ITRD Global interrupt scanner disable
ITRE Global interrupt scanner enable
LOOP Loop command for while loops
PAUSE Pause program execution, used for interrupts
RETURN Return from subroutine
RETURNI Return from interrupt
RUN Start program execution
RUN? Wait at this point for RUN command before program starts to execute

SWITCH expression Switch case statement
TWAIT Wait for trajectory to complete, only used in program
TWAIT(gen#) Wait for trajectory generator (gen#) to complete its move
WAIT=expression Set wait time in milliseconds
WHILE expression

I/O COMMANDS:
EIGN(#) Assign a single I/O point as general use input
EILN Set port C (I/O-2) as negative over travel limit
EILP Set port D (I/O-3) as positive over travel limit
EIRE Set I/O 6 to capture external encoder's current value
EIRI Set I/O 6 to capture internal encoder's current value
EOBK(#) Configure a given output to control an external brake
IN(#) x=IN(#), assign the state of a specific I/O to a variable (x in this case)
INA(A,#) x=INA(A,#), raw analog reading: 10 bit resolution spanned over signed 16 bit range
INA(V1,#) x=INA(V1,#), scaled 0-5 VDC reading in millivolts directly, 3456 would be 3.456 VDC
OR(value) Reset output (turn off)
OS(value) Set output (turn on)
OUT(#)=expression if expression LSB = 1, then it's true (1); otherwise, it's false (0)

MATH COMMANDS:
- Subtract
! Bitwise exclusive OR
!= Not equal to
% Modulo (remainder) division
& Bitwise AND
* Multiply
/ Divide
^ Power limited to 4th power and below, integers only
| Bitwise inclusive OR
+ Add
< Less than
<= Less than or equal to
== Equal to
> Greater than
>= Greater than or equal to
ABS(value) Absolute Value
ACOS(value) Arc Cosine
ASIN(value) Arc Sine
ATAN(value) Arc Tangent
COS(value) Cosine
FABS(value) Floating point absolute value
FSQRT(value) Floating point square root
RANDOM=expression Set the random seed value 0 to 2^31 - 1
RRANDOM Report the next available random number in the range 0 to 2^31 -1
SIN(value) Sine
SQRT(value) Square Root
TAN(value) Tangent
TMR(x,t) Sets timer x for t milliseconds

MOTION COMMANDS:
ADT=expression Set the accel/decel at once for a move
Ai(0) Arm index rising edge of internal encoder
Ai(1) Arm index rising edge of external encoder
AMPS=expression Current limit value. 0-1023
BREAK Break out of while loop
BRKENG Manually Engage the brake
BRKRLS Manually Release the brake
BRKSRV Brake Servo, engage the brake when the drive is not active (default)
BRKTRJ Brake Trajectory
CTR(0) Present value of internal encoder
CTR(1) Present value of external encoder
DEL=expression Set maximum allowable derivative error limit
DT=expression Set the deceleration target for a move
EL=expression Set maximum allowable following error limit
ENC0 Enable internal encoder for servo
ENC1 Enable external encoder for servo
F Set tuning values
G Go, initiates all buffered modes of operation
G(gen#) Go, initiate motion in trajectory generator (gen#)
KA=expression Feed forward gain
KD=expression Derivative gain coefficient
KG=expression Gravity offset
KI=expression PID integral gain
KL=expression PID integral limit
KP=expression PID proportional gain
KS=expression Differential sample rate
KV=expression Velocity feed forward gain
MC Initiate electronic camming
MC(2) Set Trajectory Generator 2 to run in electronic camming
MDB Enable Trajectory Overshoot Brake (TOB) when in one of the 2 trapezoidal modes
MDE Set motor to enhanced trapezoidal mode commutation by using encoder
MDS Set motor to sine mode commutation
MDT Set motor to trapezoidal mode commutation using hall sensors (default mode)
MFA(value) Accel over value master distance. Default is zero (off)
MFD(value) Decel over value master distance. Default is zero (off)
MFDIV=expression Assign Incoming counts Divisor
MFMUL=expression Assign Incoming counts Multiplier
MF0 Initiate and zero counter, but do not follow
MFR Select follow mode using quadrature encoder input.
MFSLEW(value) Stay at slew for value distance, then decel
MINV(0) Default motor commutation direction
MINV(1) Invert commutation, shaft rotates opposite direction
MP Initiate Position Mode
MSR Calculate Mode Step Ratio and prepare to follow
MT Initiate Torque Mode (Open Loop)
MTB Enable mode torque brake
MV Initiate Velocity Mode
O=expression Set origin, set present position to some value
OFF Turn the amplifier off
OSH=expression Origin shift of position counter on the fly
PML=expression Sets the position modulo limit wrap value
PMT=expression Set the position modulo target
PRT=expression Set the relative target position
PT=expression Set the absolute target position
S Instantly stop motor
S(gen#) Instantly stop trajectory generator (gen#)
T=expression Set the commanded torque while in MT mode
TH=expression Set maximum allowable thermal limit (degrees C)
VT=expression Set the velocity target for a move
X Decelerate to a stop at present deceleration rate

MULTI-AXIS COMMANDS:
(All associated motors must be on same Combitronic™ CANbus network)
ADT=expression Set the accel/decel at once for a move
ADTS=expression Set sync accel/decel at once for a move
ATS=expression Set sync acceleration target for a move
DTS=expression Set sync deceleration for a move
GS Go synchronized, initiates linear interpolated moves
PRTS=(dist1;axis1,dist2;axis2,dist3;axis3) Set synchronized relative target position
PRTSS=(dis1;axis) Set supplemental synchronized relative target position
PTS=(dist1;axis1,dist2;axis2,dist3;axis3) Set synchronized absolute target position
PTSD Stores the synchronized target move linear distance
PTSS=(dis1;axis) Set supplemental synchronized absolute target position
PTST Stores the time for synchronized move to target position
TSWAIT Wait for synchronized trajectory to complete
VTS=expression Set synchronized velocity target for a move

STATUS COMMANDS:
Ba Over current bit, status word 0, bit 4 status word 1, bit 3
Be Excessive position error, status word 0, bit 6
Bh Excessive temperature occurred, status word 0, bit 5
Bl Left (-) over travel limit, status word 0, bit 13
Bm Left (-) over travel limit active, status word 0, bit 15
Bo Motor is off, status word 0, bit 1
Bp Right (+) over travel limit active, status word 0, bit 14
Br Right (+) over travel limit, status word 0, bit 12
Bt Trajectory in progress, status word 0, bit 2
Bv Velocity limit, status word 0, bit 7
CLK=expression System Clock value in milliseconds
RAC Report commanded acceleration
RAT Report target acceleration
RB(sw,b) Report status bit, b, from status word, sw
RCKS Report Checksum
RCLK Report system clock in milliseconds
RCTR(0) Report present value of internal encoder
RCTR(1) Report present value of external encoder
RDEA Report actual derivative error
RDEL Report commanded derivative error limit
RDT Report target deceleration
REA Report actual following error
REL Report commanded following error limit
RI(0) Report where the rising edge of the internal index was detected
RIN(#) Report the state of a I/O
RIN(W,0) Report the first word of local I/O
RINA(V1,#) Reports voltage level (scaled 0-5 VDC) of analog input value for a given I/O defined by #
RJ(0) Report where the falling edge of the internal index was detected
RJ(1) Report where the falling edge of the external index was detected
RPA Report present actual position
RPC Report present commanded position
RPC(gen#) Report commanded position for trajectory generator (gen#)
RPMA Report the current modulo counter
RPML Report position modulo limit
RPMT Report the most recent setting of PMT (position modulo target)
RPRA Report actual relative position
RPRC Report commanded relative position
RPRT Report present relative target position
RPT Report present target position
RRES Report encoder resolution of motor
RSP Report sampling rate and firmware version
RT Report commanded torque
RTMR(x) Report timer x (present time left in milliseconds)
RUIA Reports current (Amps=UIA/1000)
RUJA Reports bus voltage (Volts=UJA/1000)
RVC Report commanded velocity
RVT Report target velocity
RW(value) Report status word
Z(sw,b) Clears/zeros status word bits
Za Reset over current bit
Ze Reset position error bit
Zh Reset over temperature bit
Zl Reset left(-) historical limit bit
Zr Reset right(+) historical limit bit
ZS Clear all errors, reset system latches to power up state
Zw Reset wraparound bit

VARIABLE COMMANDS:
a=expression Variable, 32 bit signed integers, a-z, aa-zz, aaa-zzz, 78 total variables
ab[x]=expression Array variables, 8 bit byte arrays, x can be 0-203
af[x]=expression Floating point array variables, x can be 0-7
al[x]=expression Array variables, 32 bit long arrays, x can be 0-50
aw[x]=expression Array variables, 16 bit word arrays, x can be 0-101
EPTR=expression EEPROM pointer, non-volatile memory, use before VLD and VST commands
VLD(variable,quantity) Load values from EEPROM to variables starting at EPTR location
VST(variable,quantity) Store values to EEPROM from variables starting at EPTR location

NOTE: See the *SmartMotor Developer's Guide* for a complete list of commands, full syntax and code examples.