

Use SMI software to program your SmartMotor motion control applications.

Note: Some commands may not apply to specific firmware releases. Please refer to our web site at www.animatics.com or contact the factory for more details. To see the SmartMotor's shaft turn, enter the following four commands into the SmartMotor Interface window (just enter the bold type):

A=100 'Set Acceleration | **V=1000000** 'Set Velocity | **P=1000000** 'Set Position | **G** 'Go, start the move

Position Commands:

A value of absolute acceleration

A=unsigned exp.31 set acceleration for position and velocity modes

V value of buffered requested velocity

V=exp.32 set required velocity for position and velocity modes

D Value of buffered relative position, phase offset, and dwell (F=16, F=128)

D=exp.32 set relative distance, and buffered relative position mode for D non zero and modes MFR & MSR, see set Mode Ratio with Phase Offset, dwell if mode cam and F=16 or F=128 is set

P value of buffered target position

P=exp.32 set buffered target position for position mode

G start buffered motion profile or trajectory; if MFR/MSR mode, start phase offset move D,V, if D not 0

TWAIT halt program command execution until trajectory completed

X decelerate to a stop using present buffered acceleration value

I value of hardware index encoder count capture

O=exp.32 reset position to a 32bit signed value; in MFR/MSR mode, O=0 reset origin and CTR to zero on the fly (read O=0 as "O=ZERO")

E value of maximum allowable position error

E=unsigned exp.15 set maximum allowable position error

AMPS value of the power limit

AMPS=unsigned exp set power limit, 0 to 1023

F=1 decelerate motor on limit switch contact

OFF turn off SmartMotor™ servo

Internal Encoder Motion Commands:

MP set buffered position mode, pending a G

MV set buffered mode velocity, pending a G

MT immediately set mode torque

T=exp set torque magnitude and direction, -1023 to 1023

External Encoder Motion Commands:

MF0 reset secondary encoder counter to zero

MFx immediately set mode follow where x= 1,2,4 times mode

MS0 reset secondary encoder to zero

MS set immediate step and direction mode

MSR set buffered mode step with ratio mode, pending a G

MFDIV value of mode step/follow ratio divisor

MFDIV=exp.16 set mode step/follow ratio divisor

MFMUL value of mode step/ follow ratio multiplier

MFMUL=exp.16 set mode step/follow ratio multiplier

MC initialize Mode Cam awaiting a G

MCx initialize Mode Cam awaiting a G, where x=2,4,8 times result

CI mode Cam apply buffered dwell value (F=16 or F=128)

BASE=exp.15 cam mode periodic encoder base where SIZE < BASE <= 32767

SIZE=exp number of array points for Cam Mode

* not for ServoStep motors

operation where 2 <= SIZE <= 100

CTR counts measured by the secondary encoder

ENC0 primary encoder used as primary encoder

ENC1 switch primary and secondary encoder functions

Program Flow Structures:

Nesting program flow structure is permitted (6 levels deep)

IF exp ... **ENDIF** IF code block

ELSEIF exp next IF test case, extended only if IF test false

ELSE remaining IF test case

ENDIF end of IF, ELSEIF, and ELSE code block

SWITCH exp ... **ENDS** SWITCH code block (SWITCH commands nested to any level now use only variable zzz)

CASE exp individual SWITCH test case

BREAK jump to exit of WHILE or SWITCH

DEFAULT if all SWITCH test cases false

ENDS end of SWITCH code block

WHILE exp WHILE code block

LOOP end of WHILE code block

Program Flow Commands:

RUN executed the stored EEPROM program, from the beginning

! suspended program execution until host channel character received

RUN? stop program executing at point of command until RUN command is received

BREAK jump to exit of WHILE or SWITCH

GOSUBnnn execute subroutine at statement label nnn, and then return to next statement

GOTOnnn jump to program statement label nnn

C# program statement label, C0 to C999

RETURN return from subroutine to program address on the stack

WAIT=exp suspend program execution for exp cycles, ~4069=1sec

Z perform software reset of SmartMotor™

END stop program execution

Code EEPROM Read/Write Commands:

LOAD receive and store in EEPROM a

SmartMotor™ program file

UPLOAD up load user EEPROM program to host terminal

UP upload user EEPROM compiled program to host terminal

RCKS report program EEPROM checksum

Variable EEPROM Read/Write Commands:

***EPTR=n** set user EEPROM memory pointer where n is 0 to 32255

VLD(var,number) load contiguous user variables from user EEPROM, number is the number of variables to be loaded

VST(var,number) store contiguous user variables into user EEPROM, number is the number of variables to be stored

Variables/System-Variables:

@P value of measured position

@PE value of measured position error

@V value of measured velocity

a to **z** 32bit signed value variables

aa to **zz** 32bit signed value variables, share memory location with array variables

aaa to **zzz** 32bit signed value variables, share memory location with array variables

ab[0] to **ab[200]** 8bit signed array variables, share memory location with aa-zz, aaa-zzz

aw[0] to **aw[100]** 16bit signed array variables, share memory location with aa-zz, and aaa-zzz

al[0] to **al[50]** 32bit signed array variables, share memory location with aa-zz, aaa-zzz

System State Flags:

The follow binary values can be tested by IF and WHILE control flow expressions, or assigned to any variable. They may all be reported using RB{bit} commands. Some may be reset using Z{bit} commands.

RW reports sixteen of these flags in combination. (listed by status bit)

Ba =1 if over current occurred, (status bit 14)

Bb =1 if comm parity error occurred

Bc =1 if comm buffer overflow occurred

Bd =1 if math overflow occurred, (status bit 11)

Be =1 if position error occurred, (status bit 5)

Bf =1 if comm framing error occurred

Bi =1 if new index report available, (status bit 3)

Bk =1 if EEPROM I/O error occurred, (status bit 15)

Bl =1 if negative limit crash occurred, (status bit 2)

Bm=1 if negative limit presently contacted, (status bit 10)

Bo=1 if motor is OFF, (status bit 7)

Bn=1 if calibration error occurs (for ServoStep only)

Bp=1 if positive limit presently contacted, (status bit 9)

Br =1 if positive limit crash occurred, (status bit 1)

Bs =1 if syntax error occurred, (status bit 13)

Bt =1 if trajectory in progress, (status bit 0)

Bu =1 if user array index error occurred, (status bit 12)

Bh =1 if overheat occurred, (status bit 6)

Bw=1 if position wrap around occurred, (status bit 4)

Bx =1 if index presently contacted, (status bit 8)

By =1 if step direction change overrun occurred (V4.40 only)

Reset System State Flag:

Za reset (Ba) over-amps flag bit

Zb reset (Bb) comm parity flag bit

Zc reset (Bc) comm overflow flag bit

Zd reset (Bd) math overflow flag bit

Zf reset (Bf) comm framing flag bit

Zl reset (Bl) negative limit crash flag bit

Zr reset (Br) positive limit crash flag bit

Zs reset (Bs) syntax error flag bit

Zu reset (Bu) array index error flag bit

Zw reset (Bw) position wrap flag bit

Zy reset (By) step dir bit (V4.40 only)

ZS reset all reset-able system flags

Motor I/O Commands:

LIMD Makes Limits Directional. A new occurrence of either limit still halts the motor. A move begun on a limit is only allowed to move in the opposite direction of the limit.

LIMH Set Limits to active High.

LIMN Makes Limits Non-directional, This is the default for $\leq V4.15$.

LIML Set Limits active-Low, This is the default for $\leq V4.15$.

UCP assign pin C to positive limit switch input, (default state)

UDM assign pin D as negative limit switch input, (default state)

UG assign pin G to synchronous "GO"(default State)

U{pin}O assign pin to be an output

U{pin}=exp. set pin output latch to 0 or 1 where 0 is zero volts, and 1 is 5VDC

U{pin}I assign pin to be a general input

var=U{pin}I Assign digital value of pin to variable (returns a 0 or 1)

var=U{pins}A Assign 10 bit analog value of a pin to a variable

In all above cases:

{pin} is A, B, C, D, E, F, or G

exp. is 0 or 1

var is any variable a thru z, aa thru zz, aaa thru zzz, ab[0] thru ab[200], aw[0] thru aw[100], or al[0]

*AniLink™ I/O Commands:

AIN{port}{input} value of 8 bit analog input

AOUT{port},{exp.8} output byte to analog port

DIN{port}{channel } AniLink™ digital input byte

DOUT{port}{channel},{exp.8} output digital byte value to AniLink™
{port} is A, B, C, D, E, F, G, or H
{input} is 1, 2, 3, or 4
{channel} is 0 thru 63
exp.8 is 0 thru 255

Communication Commands:

ADDR=exp set motor address between 0 and 99

BAUDX set baud rate to (x=2400,4800,9600, 19200,38400) bps

SADDRaaa set SmartMotor™ address, were aaa = 0 to 115

ECHO channel 0 to echo comm bytes received

***ECHO1** channel 1 to echo comm bytes received

ECHO_OFF channel0 NOT to echo bytes received channel1 NOT to echo comm bytes received

SILENT prohibit comm channel0 messages originating from within user program

***SILENT1** prohibit comm channel1 messages originating from within user program

SLEEP prohibit SmartMotor™ executing received channel 0 commands except WAKE

***SLEEP1** prohibit SmartMotor™ executing received channel 1 commands except WAKE1

* not for ServoStep motors

TALK permit channel0 comm messages originating from within user program

***TALK1** permit channel1 comm messages originating from within user program

WAKE permit received channel 0 comm commands to be executed

***WAKE1** permit received channel 1 comm commands to be executed

OCHN(type,comm,parity,bit rate,stop bits,data bits,specification) Open a comm channel type is RS2 or RS4 comm is 0, 1 baudrate 2400, 4800, 9600, 19200, or 38400 (bps) data bits is 8 stop bits is 1 specification is C (for command) or D (for data)

***F=4** redirect user program message to transmit through channel1

***PRINT1{#n,...,"text string",,exp.32,...}** print requested parameters to ch. 1

***PRINT{port}{#n,"textstring",exp.32,...}** print request parameters to AniLink™ port A-H

GETCHR fetch value of next character in host comm channel0 input buffer fetch value of next character in comm channel1 input buffer

LEN number of characters presently in host channel0 input buffer

***LEN1** number of characters presently in channel1 input buffer

PID Filter:

PIDX set PID update rate where (default update rate)/X; X=1,2,4,8

KA value of buffered acceleration feed forward gain coefficient

KA=exp.15 set buffered acceleration feed forward gain coefficient

KD value of buffered derivative gain coefficient

KD=exp.15 set buffered PID derivative gain coefficient

KG value of buffered PID constant coefficient

KG=exp.32 set buffered PID constant coefficient

KGOFF non active KG term off if position error ($\leq v4.15$)

KGON active KG term if position error ($\leq v4.15$)

KI value of buffered integral gain coefficient

KI=exp.15 set buffered PID integral gain coefficient

KL value of buffered PID integral term contribution limit

KL=exp.15 set buffered PID integral limit

KP value of buffered PID proportional gain coefficient

KP=exp.15 set buffered PID proportional gain coefficient

KS value of buffered KS differential sample rate coefficient

KS=exp.8 set buffered PID differential sample rate

KV value of buffered velocity feed forward gain coefficient

KV=exp.15 set buffered PID velocity feed forward gain

F=8 clear PID integral term at end of a trajectory.

F apply buffered filter coefficients to PID calculation

Miscellaneous:

CLK value of SmartMotor™ clock

F=32 GOSUB1 is executed upon motor entering protection fault from the following: Limit switch crashes, Over-Temp/RMS Over-current, Position Error. Code will continue from C1 until either RETURNF or STACK is encountered.

F=64 GOSUB2 is executed on user input G transition from high to low until

All code from C2 down will be executed until either RETURNI or STACK is encountered.

TEMP value of slave processor unit temperature in degrees C. It Must be assigned to a variable to be reported.

UIA value of motor current in 10ths of Amps It Must be assigned to a variable to be reported.

UJA value of motor DC bus Voltage in 10ths of Volts. It must be assigned to a variable to be reported.

Brake Commands:

BRKENG engage the brake (requires hardware brake)

BRKRLS release the brake (requires hardware brake)

BRKSRV engage break whenever servo off (requires hardware brake)

BRKTRJ engage break when trajectory is not running (requires hardware brake)

MTB apply electrical dynamic braking (in V4.76 firmware or higher only)

BRKC re-direct brake control from internal brake pin to Port C. (V4.15b or higher firmware only)

UCO must be issued prior to this command. Automatic Functionality follows BRKTRJ or BRKSRV commands as listed above.

BRKG re-direct brake control from internal brake pin to Port G. (V4.15b or higher firmware only)

UGO must be issued prior to this command. Automatic Functionality follows BRKTRJ or BRKSRV commands as listed above.

BRKI redirect brake control to internal brake control pin (Default state) (V4.15b or higher firmware only)

Report to Host Commands:

R{user variable} report user variable to host User variable is a thru z, aa thru zz, aaa thru zzz, ab[0] thru ab[200], aw[0] thru aw[100], or al[0]

R{X} report to host various commands (where {x} can be position commands, variables, system state flags, communication commands, etc)

Some New Commands to V4.76 and Higher Firmware:

SLD disable software limits

SLP=# assign value in encoder counts to positive software limit

SLN=# assign value in encoder counts to negative software limit

SLE Enable software limits. Care should be taken to disable soft limits prior to any changes. O=# will not shift soft limit values. Soft limits trigger the same status error bits as do hardware limits.