ANIMATICS

SmartMotor™ Motion Control Chip Version 4.12
Expanded Capabilities List

Animatics is upgrading all SmartMotors™ with its new Version 4.12 Motion Control Firmware. This new version is 100% backward compatible with your older programs, but has substantially increased functionality. This document is intended to serve as an addendum to the existing manual until a new manual can be published.

A CD containing the Windows based terminal software should come with the manual. The software is also downloadable on our website at www.smartmotor.com or www.animatics.com. For first time users, refer to the Help section. In the terminal screen, type the word "RSP" and see that the motor responds by returning the sample period and firmware version. A "/41X" means you have version 4.1 motor. A "/QX" means you have a version 4.00 motor. A "-MX" means you have a version 3.4 motor.

NOTE: The motor will not execute a stored program on the EEPROM if a character is transmitted to the motor ½ second after power up.

The following pages describe functional enhancements that have been added to the firmware but are not described in the current version of the SmartMotor™ manual.

COMPUTING V, A, and WAIT values for Version 4.00-QX Firmware

For NEMA17 (SM17XX) series SmartMotor™
Sample Rate = 24576.25 seconds / 100,000,000 samples = 4068.9695 samples / second
WAIT=n for 1 second wait = 4068.9695 servo samples
For a 2000 count encoder:
1 rev = 131,072,000 scaled counts
V = 1 rev / sec = 32212.578 scaled counts / servo sample
A = 1 rev / sec^2 = 7.9166433 scaled counts / servo $sample^2$

For NEMA23 and NEMA34 (SM23XX and SM34XX) series SmartMotor™
Sample Rate = 24824.24 seconds / 100,000,000 samples = 4028.3207 samples / second
WAIT=n for 1 second wait = 4028.3207 servo samples
For a 2000 count encoder:
1 rev = 131,072,000 scaled counts
V = 1 rev / sec = 32537.63 scaled counts / servo sample
A = 1 rev / sec^2 = 8.07722 scaled counts / servo $sample^2$
For a 4000 count encoder:
1 rev = 262,144,000 scaled counts
V = 1 rev / sec = 65075.26 scaled counts / servo sample
A = 1 rev / sec^2 = 16.15444 scaled counts / servo $sample^2$

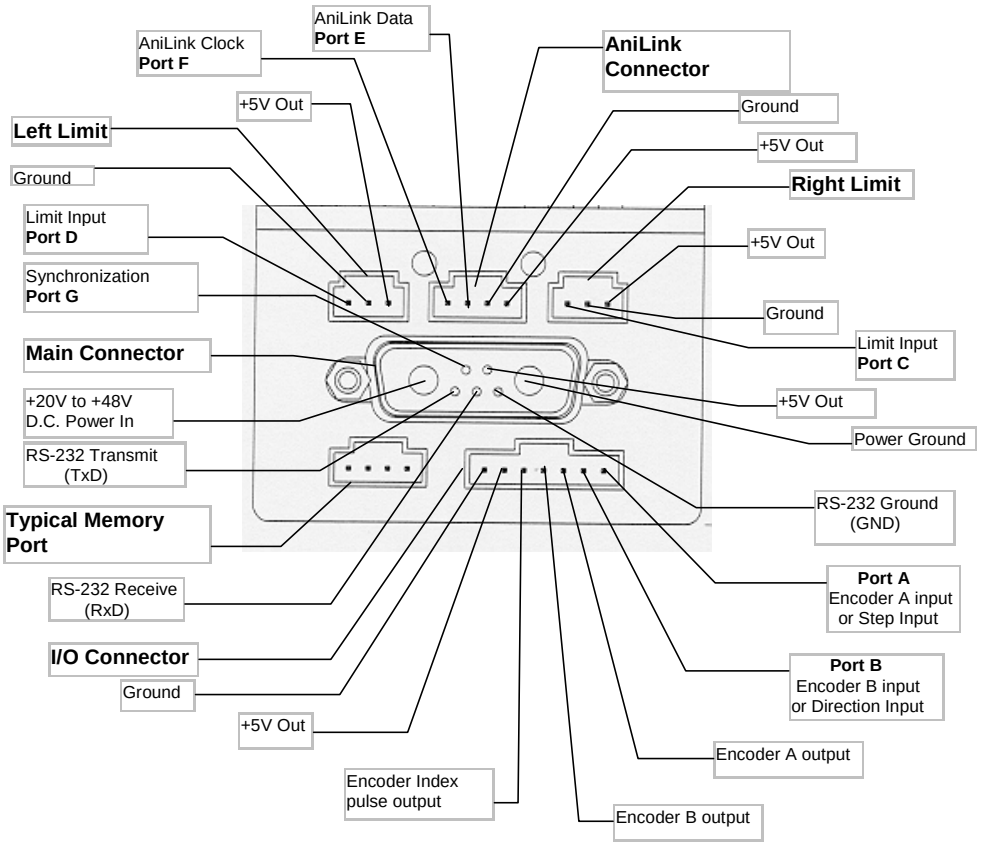
COMPUTING V, A, and WAIT values for Version 4.11 and 4.12 Firmware

For ALL Version 4.1X series SmartMotor™
Sample Rate = 24576.25 seconds / 100,000,000 samples = 4068.9695 samples / second
WAIT=n for 1 second wait = 4068.9695 servo samples
For a 2000 count encoder:
1 rev = 131,072,000 scaled counts
V = 1 rev / sec = 32212.578 scaled counts / servo sample
A = 1 rev / sec^2 = 7.9166433 scaled counts / servo $sample^2$
For a 4000 count encoder:
1 rev = 262,141,000 scaled counts
V = 1 rev / sec = 64425.156 scaled counts / servo sample
A = 1 rev / sec^2 = 15.833286 scaled counts / servo $sample^2$

SmartMotor Input/Output Capability

Typical SmartMotor™ Connector Configuration

Note: All ports (A-G) can be set as digital I/O or analog input pins.
Anilink port may be configured to RS-485



The following commands can be used with Version 4.00–Q3 and below series SmartMotors™

CONTROL FLOW

The number of subroutine labels have been expanded. Following are the new limitations:

GOSUBn	n = 0...999	(This results in 1000 possible subroutine labels)
GOTOn	n = 0...999	
Cn	n = 0...999	(Subroutine labels)
STACK	reset user program stack pointer	(Repairs stack damage from GOTOing out of a nested subroutine)

The "IF" statement now has optional "ELSE" and "ELSEIF.

Example:

```
IF a==1
    GOSUB1
ELSEIF a==2
    GOSUB2
ELSE
    GOSUB3
ENDIF
```

A "SWITCH" structure has been added

Example:

```
SWITCH a            'Depending on value of variable "a" the following cases select.
    CASE 1          'Where a==1
        GOSUB1
        BREAK
    CASE 2          'Where a==2
        GOSUB2
        BREAK
    DEFAULT         'Where a is anything else
        GOSUB3
        BREAK
ENDS                'Ends the SWITCH
```

There is now no limit to the number of WHILE…LOOPs

ADDITIONAL SIGNED VARIABLES

The number of variables have been expanded dramatically. The user now has the following variables at their disposal:

a-z	32 bits (first set of 26 variables)	(32 signed bits)
aa,bb, cc ,dd…..-zz	32 bits (second set of 26 variables)	(32 signed bits)
aaa, bbb, ccc -zzz	32 bits (third set of 26 variables)	(32 signed bits)

As an alternative to the above two and three letter variables, sharing the same data space, the following arrays could be used. The same space can be used for three different variable types: 8 bit; 16 bit; & 32 bit. Naturally, the number of variables yielded from the space varies in accordance with the variable data lengths.

ab[i]	8 bits	i = 1...200	overlays aa-zzz	(8 signed bits)
aw[i]	16 bits	i = 1...100	overlays aa-zzz	(16 signed bits)
al[i]	32 bits	i = 1...50	overlays aa-zzz	(32 signed bits)

"i" can be a variable, a to z, or a constant. Alternatively, "i" can be the sum or difference of any two variables, a+b or a-b. (If the sum or difference is used, discrete numbers like i+1 can not be used.)

The array variables can be read from a host just as the traditional variables are, by prefixing the variable name with a capital "R".

Rany_user_variable ex: Rab[4]

For flexibility, a long (signed 32 bit number) can be written and four bytes read from the same space, or visa-versa.

VARIABLE STORAGE - LONG TERM

An additional 8k EEPROM chip has been added inside the SmartMotor for the long term storage of data. It is accessed by first locating a pointer into that memory space and then doing single or bulk stores and loads. The associated commands are:

EPTR Locates the EEPROM pointer, range (0……..7999)

VST	Stores data at that location and forward
VLD	Loads (retrieves) data from that location and forward

VST Example

EPTR=1000	Set the pointer to 1000 to a memory location
VST(aw[15],5)	This command stores five words from the array starting with aw[15]. Each word is two bytes, the five words will take up10 bytes of space. Memory required for the variable type is automatically allocated.
p=15	In this part of the example we initialize variable p and q
q=5	
EPTR=2000	Set the pointer arbitrarily to 2000
VST(ab[p],q)	You can use variables (like 'aw') in place of constants

VLD Example

EPTR=1300	Set the pointer to read the required memory address in the EEPROM
VLD(t,6)	This command reads six variables from the EEPROM and writes to t and then u,v,w,x, and y.
VLD(aw[10],3)	Read three words from EEPROM and write to aw[10], aw[11], aw[12].

VARIABLE INITIALIZATION

Because of the vastly greater number of variables and additional variable types, we have implemented a convenient way of initializing a series of variables on a single line. It is shown in the following example, note the period after the last line entry:

Example

a 1 -2 13 24 35.	Performs: a=1 b=-2 c=13 d=24 e=35
aw[12] 15 20 30.	Performs: aw[12]=15 aw[13]=20 aw[14]=30
al[s] -1 -2 -3 4.	Performs: al[s]=-1 al[s+1]=-2 al[s+2]=-3 al[s+3]=4

COMM CHANNELS

Dramatic changes have been implemented in the area of serial communications. The primary RS-232 channel has been enhanced so that it can input string data. By executing in the "DAT" command, the primary channel will no longer interpret commands, but rather input characters and place them in an input buffer for the user program to access. An RS-485 port has been added to the motor. It uses the same pins as the AniLink port. The AniLink port can be used for RS485

or the Animatics AniLink network. It is not possible to use both functions at the same time. The RS485 port can be opened with a vast array of different parameters and will input commands or data. It responds to the same commands with the addition of the number "1".

CMD Define primary RS-232 port as command input port
CMD1 Define secondary RS-485 port as command input port
DAT Define primary RS-232 port as data input only
DAT1 Define secondary RS-485 port as data input only
SLEEP1, WAKE1, TALK1, SILENT1 All for additional RS-485 Channel

Each port can be opened with a variety of choices:

OCHN(TYPE,CHANNEL,PARITY,BAUDRATE,STOPBITS,DATABITS,SPEC)

Example:	OCHN(RS2,0,N,9600,1,8,C)	Primary host command channel 9600
Example:	OCHN(RS2,0,N,9600,1,8,D)	Primary host data channel 9600
Example:	OCHN(RS4,1,N,38400,1,8,C)	Second port
Example:	OCHN(RS4,1,N,19200,1,8,D)	Second port
Example:	OCHN(IIC,2,N,C1,1,8,M)	AniLink (IIC) channel

To close a comm channel:

CCHN(TYPE,CHANNEL)

Example:	CCHN(RS2,0)	close host channel
Example:	CCHN(RS4,1)	close secondary channel
Example:	CCHN(IIC,2)	close AniLink (IIC) channel

To identify the existence of characters in comm channel buffer:

LEN, LEN1

To get characters from a comm channel:

GETCHR, GETCHR1

Example

```
IF LEN>0          ensure there are characters in the buffer before using GETCHR
a=GETCHR          fetch a character from host communications channel buffer
```

IF LEN1>0	ensure there a characters in the secondary channel (usually RS485 buffer) before using GETCHR1
ab[3]=GETCHR1	fetch a character from secondary communications channel buffer to array

Reporting commands exist to track the status of the communication channels:

RCHN	report comm channel status - all
RCHN0	report comm host channel status
RCHN1	report comm channel 1 status

where bit0 = overflow error
bit1 = framing error
bit2 = command scan error
bit3 = parity error

Printing to a channel

Because multiple comm channels exist now, enhancements were made in the PRINT commands:

PRINT(...)	print to the primary host channel
PRINTA(...) - PRINTH(...)	Print to the AniLink channels A – H
PRINT1(...)	print to channel 1 (usually the default RS485 channel)

MODE STEP & MODE FOLLOW WITH RATIO

SmartMotor™ users wanted variable mode follow ratios. To address that need with a fixed point system, we implemented a numerator/denominator function. This uses an 'integer fraction' to provide the equivalent of a floating point relationship between the motor and an incoming encoder, or step and direction, signal. For example, a mode follow relationship of 4.125 can be produced by setting MFDIV=8 and MFMUL=41.

MSR	Engage Mode Step Ratio
MFR	Engage Mode Follow Ratio
MFMUL=expression	Set numerator (signed 16 bit integer)
MFDIV=expression	Set denominator (signed 16 bit integer)

Example:

MF4	Set counter mode (resets external counter to zero)
MFDIV=1000	Set ratio divisor
MFMUL=33	Set ratio multiplier
MFR	Set Follow-Ratio Mode, initiating ratio calculation
G	Start (applies new ratio to all subsequent external encoder changes)

MOTION COMMANDS

OFF	Turn motor servo off.

SYSTEM STATES

State Variables: State variables are binary variables either set (1) or reset (0).
They reported using the RB<?> command.
Some RB<?> are not supported by High Resolution Hardware

RS Report status byte

Variable	Explanation	Bit#	Value	
Bo =1	Motor is OFF	bit7	128	real time
Bh =1	Excessive temperature	bit6	64	real time
Be =1	Excessive position error occurred	bit5	32	reset by G
Bw =1	Wraparound occurred	bit4	16	reset by G,MT,&Zw
Bi =1	Index report available	bit3	8	reset by RI
Bl =1	Historical left limit	bit2	4	reset by Zl, ZS, RS & RW
Br =1	Historical right limit	bit1	2	reset by Zr, ZS, RS & RW
Bt =1	Trajectory in progress	bit0	1	real time

RW Report status word

Variable	Explanation	Bit#	Value	
Bk =1	User program check sum error occurred	bit15	32768	real time
Ba =1	Over current state occurred	bit14	16384	reset by Za & ZS
Bs =1	Syntax error occurred	bit13	8192	reset by Zs & ZS
Bu =1	User array index range error occurred	bit12	4096	reset by Zu & ZS
Bd =1	User math overflow occurred	bit11	2048	reset by Zd & ZS
Bm =1	Left limit asserted	bit10	1024	real time

Bp	=1	Right limit asserted	bit9	512	real time
Bx	=1	Hardware index input level	bit8	256	real time
Bo	=1	Motor is OFF	bit7	128	real time
Bh	=1	Excessive temperature	bit6	64	real time
Be	=1	Excessive position error occurred	bit5	32	reset by G
Bw	=1	Wraparound occurred	bit4	16	reset by G,MT,&Zw
Bi	=1	Index report available	bit3	8	reset by RI
Bl	=1	Historical left limit	bit2	4	reset by Zl, ZS, RS & RW
Br	=1	Historical right limit	bit1	2	reset by Zr, ZS, RS & RW
Bt	=1	Trajectory in progress	bit0	1	real time

Bb =1 Parity error occurred
Bc =1 Communication overflow occurred
Bf =1 Communications framing error occurred

States Resets: State variables are reset to zero with the following commands.

Za Reset hardware current limit violation indication.
Zb Reset serial data parity error indication.
Zc Reset communications buffer overflow indication.
Zd Reset user math overflow indication.
Zf Reset communications framing error indication.
Zl Reset left limit "seen" indication.
Zr Reset right limit "seen" indication.
Zs Reset user command syntax error indication.
Zu Reset user array indexing out of range indication.
Zw Reset wraparound indication.

Global Resets:

Z Software reset; the user program counter, user variables, and all firmware states are reset.
ZS Reset all individual user system bits listed above without performing a complete [Z] reset.

REPORTS

RF report last F = value
RMODE report SmartMotor MODE of operation
Where P Absolute position move

R Relative position move
V Velocity mode
T Torque mode
F Follow mode using MF1, MF2, or MF4
S Step & direction mode
C Cam mode
X Follow mode or step & direction mode with ratio
E Position Error
O Motor off

RCS1 Report and Clear 8 bit check sum of channel 1 communication bytes.

LOADING AND UPLOADING USER PROGRAMS

UP	Transmit stored SmartMotor program in tokenized format to the host terminal.
ES400	Force EEPROMS to be read from & written to at 400 bits per second.
ES1000	Force EEPROMS to be read from & written to at 1000 bits per second. (Newer EEPROMS only)
RCKS	Return check sums. If an “F” appears following the check sums, there is a definite read/write error.

CAM MODE

For reciprocating motion such as the output from a variable profile CAM, there is a new mode initiated by the command "MC" (Mode Cam). This function allows a complex relationship between an incoming encoder signal and the motor's own encoder. It is described by as many as 100 sixteen bit words loaded in the array, and will linearly interpolate between those points.

The BASE represents the number of external encoder counts to define one cam cycle or cam revolution.

BASE = expression, where 2 <= BASE <= 32768

The number of cam table entries is given by

SIZE = expression, where 2<= SIZE <= 100 and SIZE <= BASE

16 bit data entries, where Data Table Stored at aw[0]…aw[SIZE-1]. The data entries must represent a sample point at evenly spaced intervals of the required offset

Example:

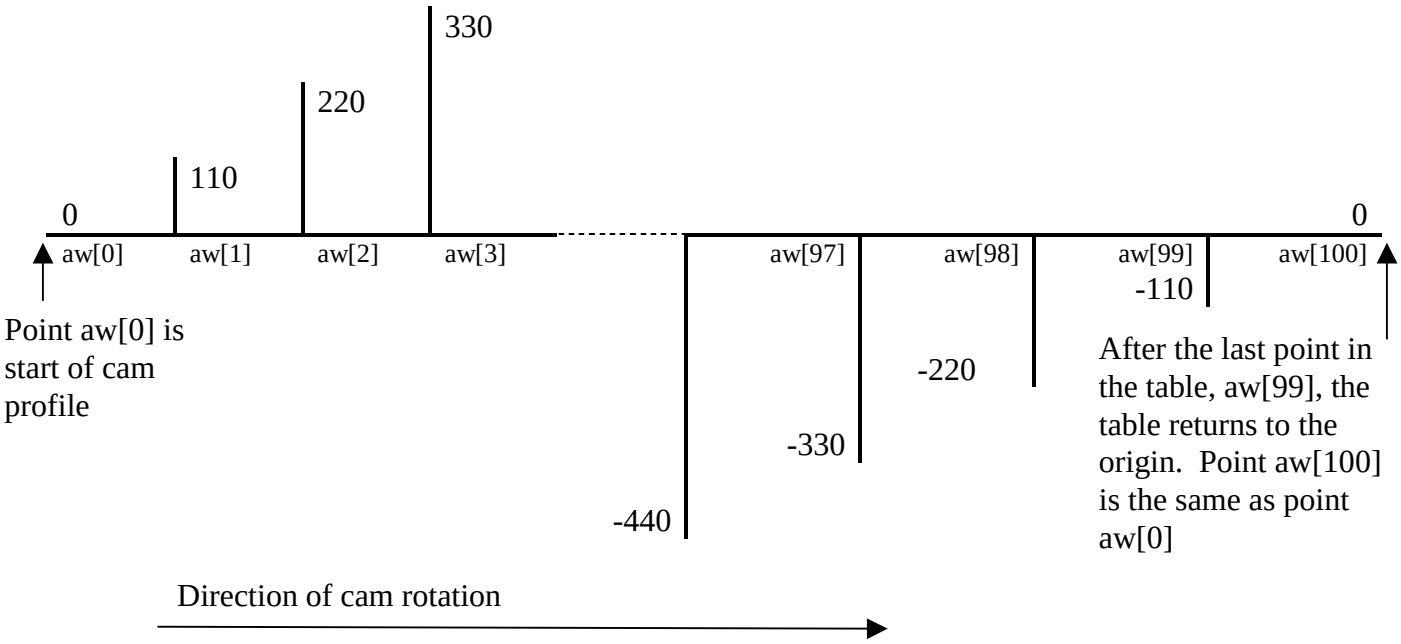
MF4	Reset external encoder mode and zero counter
BASE=10000	Set the number of external encoder counts per cycle
SIZE=100	Tell how many array points, or table entries, to use
E=10000	It may be necessary to increase error band to ensure the motor, when rotating at a high speed, does not assumme it is seeing a position error as it tries to move to the next point in the cam table. CAUTION: ensure that the error band is not set so high that it allows the motor to perform a violent move that damages machinery or personnel.

aw[0] 0 110 220 330 440 … -440 -330 -220 -110. Load the array

The maximum line length is 128 characters. If more characters are required, use a carriage return. At the last point in the table, aw[99] the 'cam' has almost completed one revolution. At position aw[100] the cam has completed one revolution returns to the first point in the table, the origin aw[0].. Note the array must end with a period.

MC	Set Mode Cam, performs buffered calculation
G	Start, resets motor position to zero – points at beginning of table and engages CAM follow mode

The cam profile of this example is shown in the following sketch:



In this example, 10000 counts from the external encoder drives one 'rotation' of the cam. To continue cycling through the cam table, external encoder readings are translated to an offset within the range 0-10000. An external encoder measurement of 10350 translates to 10350 – 10000 = 350

Since the BASE = 10000 and there are 100 table entries, each table entry represents a segment of magnitude 10000/100 or 100 counts. The table entries, aw[ ], correspond to external encoder readings at 100 count intervals:

aw[0] 0, aw[1] 100, aw[2] 200, aw[3] 300, aw[4] 400,……..
……aw[96] 9600, aw[97] 9700, aw[98] 9800, aw[99] 9900

In our example, if the external encoder reading translates to 350, the sample point lies between aw[3] and aw[4]. The actual requested shaft position is calculated using linear interpolation.

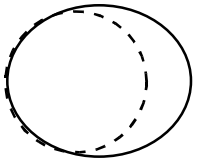
From the cam table:

aw[3] = 330
aw[4] = 440
If external encoder = 350,
Required position = sample position aw[3] + a fractional part of (aw[4] –aw[3])
Required position = 330 + ((350-300)/100 * (440 – 330) = 385
= 330 + 50/100 * 110
= 330 + 55 = 110

Note: The SmartMotor sets its position to ZERO (P=0) where the mode is engaged. The table offsets are relative to ZERO.
The external counter may reverse at any time.
Both positive and negative external encoder readings map to the data table.
The normal PID update rate (4 kHz) is maintained.
The required position calculation is rounded to the nearest encoder count.

Extreme cam profiles

At the end of each cam revolution, the table returns to the origin. Remember that abrupt changes in cam profile, or increases/decreases in cam diameter, will require high acceleration from the motor to ensure the shaft moves to the required angular position during one interval. Extreme changes could cause error problems, especially if the system is rotating at high speed.



Cam profile with smooth transitions

Cam profile with abrupt transitions

BRAKE COMMANDS

For motors equipped with built-in brakes a series of new commands and dedicated internal I/O facilitate the use of the brake

BRKENG	Engage the brake.
BRKRLS	Release the brake.
BRKSRV	Engage the brake whenever the motor is not servoing.

PIN FUNCTIONALITY

New flexibility and functionality has been added to all I/O:

UA=expression	(set pin OUT LATCH to lsb, 0 or 1)
UAI	Assign pin A to input state
UAO	Assign pin A to output state
variable=UAA	Read pin a as 10 bit analog input
UB=expression	(set pin B output latch state)
UBI	Assign pin B to input state
UBO	Assign pin B to output state
variable=UBA	Read pin B as 10 bit analog input
UC= expression	(set pin C output latch state)
UCI	Reassigns right limit to input state
UCO	Reassigns right limit to output state
variable=UCA	Read pin C as 10 bit analog input
UCP	Restore pin to right (plus) limit function
UD= expression	(set pin D output latch state)
UDI	Reassigns left limit to input state
UD0	Reassigns left limit to output state
variable=UDA	Read pin D as 10 bit analog input
UDM	Restore pin to left (minus) limit function
UE= expression	(set pin E output latch state)
UEI	Assign Anilink Data pin to input state
UEO	Assign Anilink Data pin to output state
variable=UEA	Read pin E as 10 bit analog input
UF= expression	(set pin F output latch state)
UFI	Assign Anilink Clock pin to input state
UFO	Assign Anilink Clock pin to output state
variable=UFA	Read pin F as 10 bit analog input
UG	Restore G sync line functionality
UG=expression	(set pin G output latch state)
UGI	Assign pin G to input state
UGO	Assign pin G to output state
variable=UGA	Read pin G as 10 bit analog input

PROGRAM CHECKSUM

User programs and subroutine jump tables both have stored checksums. The command RCKS emits the checksums followed by 'F' or 'P' to indicate Fail/Pass. System bit Bk reflects the EEPROM Write/Fail state. A new upload command, "UP", permits a byte for byte comparison to the compiled version of the user source code, since the RCKS pass response does not absolutely verify the stored EEPROM program. The original UPLOAD command recreates the original uncompiled user source code.

The commands are:

RCKS

RCKS returns two checksums, from the label table and the program. The returned output is:

Label checksum Program checksum 'result' The result is either 'P' or 'F' for pass or fail.

RBk returns a system bit, k. If Bk = 1 it indicates the EEPROM not verified. A return 0 indicates no EEPROM failure detected.

PID UPDATE RATE

The user can now slow the PID update rate with the new PID command. Reducing the update rate can cause an increase in program execution speed. The commands are:

PID1	Invokes the default rate
PID2	Divides the rate by two
PID4	Divides the rate by four
PID8	Divides the rate by eight

Since velocity and acceleration as well as the WAIT= statement are functions of PID rate, expect to have to modify them to get equivalent motion or timing.

The following commands can only be used with Version 4.11 and below series SmartMotors™

FUNCTION COMMANDS

F = 2	Enables the emission of characters "!h", "!1", or "!e" upon command error dectection.
F = 4	Report command responses are re-directed to RS485 channel 1(Anilink Port) instead of to RS232 channel 0(Main Connector).

MOTION COMMANDS

KGON	This command reduces torque ripple and smooths motion against an externally applied constant force on the axis. An example of such a force is gravity acting on a lead screw driven elevator. At low speeds, the peak torque increases, but the continuous torque decreases. The stability also changes so the PID filter coefficients should be modified.
ENC0	(Default) Internal encoder is primary encoder
ENC1	External encoder is primary encoder and the shaft position is recorded in the counter(CTR)

LOADING AND UPLOADING USER PROGRAMS

LOCKP	Lock protects the user program from upload.

PHASE OFFSET

A phase offset may be introduced when using the mode following commands. A phase offset allows the motor shaft to be aligned relative to the signal being followed. The offset is initiated by the G command, if the D command variable holds a (non-zero) offset. The V command variable controls the rate at which the offset is applied. The offset value in D decreases in magnitude until it reaches 0. V remains unchanged. The A command variable is used for internal calculations, so any prior value will be lost. If no offset is desired, be sure D=0 when issuing the G command while in MFR or MSR mode. While the offset is being applied, changes to D,V, or A will have immediate effect.

Example:
(Enter these commands directly to the motor)

MF4	Set counter mode (resets external counter to zero)

MFDIV=1	Set ratio divisor
MFMUL=1	Set ratio multiplier
MFR	Set Follow-Ratio Mode, initiating ratio calculation
D=0	Ensure there is no unintended phase offset
G	Start (applies new ratio to all subsequent external encoder changes)

Let's say we observe the motor shaft off ¼ revolution relative to the external encoder, both turning about 1 rps.

D=500		Set positive ¼ revolution relative offset for motors with a 500 line encoder
V=410		Want the relative offset to complete in 20 seconds. (500 counts/20 seconds * 65536 scaled counts/count * 1 second/4000 servo samples = 410 scaled counts/servo sample)
G		Begin phase offset
V=0		Pause phase offset
RD	201	Shaft move 299 counts relative to gear, 201 counts to go
V=410		Resume phase offset
RD	0	Phase offset complete

CAM MODE

MC2	Mode cam with final position request multiplied by 2
MC4	Mode cam with final position request multiplied by 4
MC8	Mode cam with final position request mutilplied by 8

BRAKE COMMANDS

BRKTRJ	Engage the brake whenever the motor is not completing a trajectory

MOTOR AND LOAD PROTECTION FEATURES

The SmartMotor™ servo motor is equipped with several protection features and diagnostic tools that allow the user to protect and perform diagnostic functions on the load. These are broken down into power limit, temperature, error and power monitoring functions.

Peak Power Limit

The internal peak power limit of the SmartMotor™ servo motor is set by the AMPS command. This function controls not simply current, but the amount of power that is delivered to the motor

coils. Note that this means that the AMPS command can be used to limit not only output torque, but maximum speed, as well.

The valid range of the AMPS command is 0 through 1023. The default setting is 1000. For example, a setting of

AMPS=512

will limit the output stall torque to ½ of the peak torque rating *and* limit the maximum velocity to ½ of the specification. To get full torque and speed, the value of AMPS must be set to 1023.

AMPS is can be assigned to a variable. For example,

i=AMPS

will store the value of AMPS in the variable i.

Error Limits

The position error range has increased to 23 bits, 0-8388607. This affects the following commands: E=, RE, =E, RPE, and =@PE. This change is useful in compensating for large position errors while using the VRE (Variable Resolution Encoder).

RMS Power and Temperature Limits

The RMS power consumption is constantly monitored by the SmartMotor™ servo motor. If the RMS power exceeds continuous output rating of the SmartMotor™ servo motor for a programmable amount of time, the amplifier will shut down and indicate an overheat error (see status bit Bh). This programmable time is set by the THD function. The valid range for THD is 0 through 65535, with units in servo samples. For example,

THD = 4069

will set the thermal shut down delay to one second. This means that the RMS input power must exceed the specification for 1 second before the amplifier will shut down. The default value for THD is 12000, or approximately three seconds.

THD is can not be assigned to a variable.

Furthermore, the SmartMotor™ servo motor monitors its internal temperature. If the internal temperature exceeds a programmable set point, the amplifier shuts down and indicates an overheat error (see status bit Bh). The SmartMotor™ servo motor will remain in an overheat condition until the internal temperature drops 5° C below the programmable set point. This set point is determined by the function TH. The valid range for TH is 0 to 70, with units in degrees Celsius. For example, if

TH=50

the amplifier will indicate an overheat if the internal temperature reaches 50°C and will come out of the overheat condition when the termperature falls below 45°C. The default value for TH is 70. THD can be assigned to a variable. For example,

t=TH

will assign the value of TH to the variable t.

Power and Temperature Monitoring

The Temperature and RMS power of the SmartMotor™ servo motor can be monitored for diagnostic, preventative maintenance and other reasons. The real time temperature is read by the TEMP function and is given in units of degrees Celsius. TEMP is used by assigning it to a variable. For example,

t=TEMP

will assign the internal temperature to the variable t.

The bus voltage is monitored by the user J analog input via the UJA function. User J is not physically accessible by the user. UJA will provide the input bus voltage in tenths of volts. The accuracy of the reading is ±1VDC. For example,

v=UJA will assign the input voltage to the variable v. If the reading is 336, the input voltage is 33.6±1 VDC.

The RMS current is monitored by the user I analog input via the UIA function. User I is not physically accessible by the user.

UIA will provide the measured RMS current in hundredths of ampere. The accuracy of the reading is 0.1A. For example,

i=UIA will assign the RMS current to the variable i. If the reading is 234, the measure current is 2.34±0.1 Amps.
The following commands can only be used with Version 4.12 and below series SmartMotors™

REPORTS

RPW report position followed by comm and then (16 bit) status word

NOTE: External EEPROMS may be a maximum 32k instead of 8k