

## F=# Function Command Overview

The F command value allows Enabling or Disabling of special firmware functions of the SmartMotor Processor and Drive Stage.

**Syntax: F=value**

The Value is a Binary Bit weighted value with each bit controlling a specific feature.

Bit Value Functions are as follows:

- 1 Decelerate to stop on limit switch input (as opposed to just turning off)
- 2 \* Invert Communication (Changes Shaft rotation)
- 4 Any Report commands transmit to Com 1 only. (Use with Extreme Caution)
- 8 Clear PID integral term at trajectory-end to avoid possible slow settling
- 16 \* Mode Cam positions are relative for each re-entry into CAM table (from either direction)
- 32 \* GOSUB1 is issued under motor fault condition C1 can not be called again prior to receiving a RETURNF
- 64 \* GOSUB2 is issued on user input G transition from high to low C2 can not be called again prior to receiving a RETURNI
- 128 \* Internal Slave Counter = base + dwell module while in CAM Mode
- 256 \*\* Set T.O.B. to be active for entire move profile
- 512 \*\* Suppress T.O.B. until Target Velocity has been reached
- 1024 \*\* Enables Port G to Index trigger latch function (only in SM2316D/DT and > = 4.93 firmware)

\* Note: Only Applies to >=v4.77 and higher, "plus"

\*\*Note: Must specify 4.78T firmware

**F Command is Binary Bit flag additive:**

Example:

F=21 would break down to F=(16+4+1).

Motor would run CAM Mode relative, redirect print statements to port 1, and decelerate on limits collectively.

**Warning: C1 has priority over C2. C1 can be activated when in C2.**

The F value can be changed on-the-fly while in an Interrupt subroutine to change its effect.

An example would be turning off the G interrupt once in C2 to prevent any subsequent calls.

## Modes of Operation:

- MP** Set Controller for Position Mode, pending a G
- MV** Set Controller for Velocity Mode, pending a G
- MT** Immediately set controller to Torque Mode
- MFx** Immediately set Controller to Mode-Follow (Electronic Gearing to follow External Encoder) where x = 1,2,4
- MS** Immediately Set Controller to Step-Mode (Step and Direction Input)
- MC** Initialize Mode Cam awaiting a G
- MTB** Mode Torque Brake (Dynamically brake)  
Note: MTB applies to PLS firmware only.

## Position Commands:

- A** Value of absolute acceleration
- A=expression** Set Acceleration for Position and Velocity Modes (unsigned 16-bit value)
- V** Value of buffered requested velocity
- V=expression** Set required Velocity for Position and Velocity Modes (Signed 32-bit value)
- D** Value of buffered relative position, phase offset, and [Dwell (F=16, F=128)]
- D=expression** Set Relative Distance for Relative Position Mode, (signed 32-bit value)  
Set Phase Offset Distance In Electronic Gearing, Set Dwell in Cam Mode (See F-Function Commands for more)
- P** Value of buffered target position
- P=expression** Set buffered target Position for Absolute Position Mode (signed 32-bit value)
- G** Start buffered motion profile or trajectory;  
Initiate Mode Follow Ratio in Electronic Gearing  
Initiate Phase Offset Move in Electronic Gearing  
Initiate all buffered move profile values such as Velocity, Acceleration, etc.
- TWAIT** Halt program command execution until trajectory completed
- X** Decelerate to a stop using present buffered acceleration value
- S** Decelerate to a stop using firmware fixed high rate of deceleration
- I** Index Pulse Position of Internal Encoder at last point of capture
- O=expression** Reset Origin in Position Register (to a signed 32-bit value)
- E** Value of Maximum Allowable Following Error in Encoder Counts
- E=expression** Set Maximum allowable Position Error (unsigned 0-300000 max)
- AMPS** Value of the power limit
- AMPS=expression** Set PWM Power limit, 0 to 1023 represents 0-100% allowable PWM
- OFF** Turn off Drive Stage of SmartMotor™ servo
- T** Value of Commanded Torque (Open-Loop Commanded PWM to Drive Stage)
- T=expression** Set torque magnitude and direction, (signed values of -1023 to 1023)

**External Encoder Motion Commands:**

<b>MF0</b>	Reset secondary encoder counter to zero
<b>MS0</b>	Reset secondary encoder to zero
<b>MFDIV</b>	Value of Mode Follow Ratio Divisor
<b>MFDIV=expression</b>	Set Ratio divisor value (16-bit signed value)
<b>MFMUL</b>	Value of Mode Follow Ratio Multiplier
<b>MFMUL=expression</b>	Set Ratio Multiplier value (16-bit signed value)
<b>MSR</b>	Calculate New Buffered Step Mode Ratio values from MFMUL and MFDIV, pending a G
<b>MFR</b>	Calculate New Buffered Follow Mode Ratio values from MFMUL and MFDIV, pending a G
<b>MCx</b>	Initialize Cam Mode awaiting a G, where x =2, 4, or 8 times result
<b>CI</b>	Mode Cam Table Index Value, (present Cam table pointer)
<b>BASE=expression</b>	Cam Mode periodic encoder base where SIZE < BASE <= 32767
<b>SIZE=expression</b>	Number of Array Points in Cam Table for Cam Mode operation where 2 <= SIZE <= 100
<b>CTR</b>	External Encoder Position Register Value
<b>CTR=0</b>	Set External Encoder Register to Zero
<b>ENC0</b>	Close Position Loop on Internal Encoder (Default State)
<b>ENC1</b>	Close Position Loop on External Encoder (Optional State)

**Program Flow Structures:**

Nesting program flow structure is permitted (6 levels deep)

<b>IF expression ...</b>	Beginning of "IF" code block
<b>ELSEIF expression</b>	Next "IF" test case, extended only if "IF" above is false
<b>ELSE</b>	Remaining "IF" test case
<b>ENDIF</b>	End of IF, ELSEIF, and ELSE code block
<b>SWITCH expression ...</b>	ENDS SWITCH code block (resultant value of expression stored in the variable zzz)
<b>CASE value</b>	Individual SWITCH test case
<b>BREAK</b>	Jump to exit of WHILE or SWITCH
<b>DEFAULT</b>	If all SWITCH test cases false
<b>ENDS</b>	End of SWITCH code block
<b>WHILE expression</b>	WHILE code block
<b>LOOP</b>	End of WHILE code block
<b>RUN</b>	Executed the stored EEPROM program, from the beginning
<b>!</b>	Suspend program execution until ANY Incoming Communications is received
<b>RUN?</b>	Stop program executing at point of command until RUN command is received

<b>BREAK</b>	Jump to exit of WHILE or SWITCH
<b>GOSUBnnn</b>	Execute subroutine at statement label nnn, and then return to next statement
<b>GOTOOnnn</b>	Jump to program statement label nnn
<b>C#</b>	Program Location Label for GOT and GOSUB calls, C0 to C999
<b>RETURN</b>	Return subroutine to program address on the stack (just below GOSUB call)
<b>WAIT=expression</b>	Suspend program execution for given number of PID cycles, ~4069cycles = 1sec
<b>Z</b>	Perform Software CPU Reset of SmartMotor™
<b>END</b>	Stop Program Code Execution

**User Program EEPROM Read/Write Commands:**

<b>LOAD</b>	Receive and Store into EEPROM a compiled SmartMotor™ program file
<b>UPLOAD</b>	Upload User Program to host terminal
<b>UP</b>	Upload Compiled User Program and Header file to host terminal
<b>RCKS</b>	Report Compiled User Program EEPROM checksum

**Variable/Data Storage EEPROM Read/Write Commands:**

<b>EPTR=expression</b>	Set user EEPROM memory pointer where n is 0 to 32255
<b>VLD (variable, number)</b>	Load contiguous user variables from user EEPROM number is the number of variables to be loaded
<b>VST (variable, number)</b>	Store contiguous user variables into user EEPROM, number is the number of variables to be stored

**Variables/System-Variables:**

<b>@P</b>	Value of measured position
<b>@PE</b>	Value of measured position error
<b>@V</b>	Value of measured velocity
<b>a thru z</b>	32-bit Signed Integer value variables
<b>aa thru zz</b>	32-bit Signed Integer value variables, (shares memory location with array variables)
<b>aaa thru zzz</b>	32-bit Signed Integer value variables, (shares memory location with array variables)
<b>ab[0] thru ab[200]</b>	8-bit Signed Integer Array Variables, (shares memory location with aa-zz, and aaa-zzz)
<b>aw[0] thru aw[100]</b>	16-bit Signed Integer Array Variables, (shares memory location with aa-zz, and aaa-zzz)
<b>al[0] thru al[50]</b>	32-bit Signed Integer Array Variables, (shares memory location with aa-zz, and 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 and are ideal for Fault Detection and control when operating via Serial Communications.

**RW** Report Status Word (See Individual Status Bits Below)

<b>Bt</b> =1 if trajectory in progress,	Bit: 0, value: 1
<b>Br</b> =1 if Positive Travel Limit Exceeded	Bit: 1, value: 2
<b>Bl</b> =1 if negative limit crash occurred	Bit: 2, value: 4
<b>Bi</b> =1 if new index report available	Bit: 3, value: 8
<b>Bw</b> =1 if Wrap Around occurred	Bit: 4, value: 16
<b>Be</b> =1 if position error occurred	Bit: 5, value: 32
<b>Bh</b> =1 if Exceeded Thermal Limit	Bit: 6, value: 64
<b>Bo</b> =1 if Drive Stage is OFF	Bit: 7, value: 128
<b>Bx</b> =1 if Drive Stage is OFF	Bit: 8, value: 256
<b>Bp</b> =1 if on Positive Travel Limit,	Bit: 9, value: 512
<b>Bm</b> =1 if on Negative Travel Limit ,	Bit:10, value:1024
<b>Bd</b> =1 if math overflow occurred,	Bit: 11 value:2048
<b>Bu</b> =1 if user array index error occurred,	Bit: 12, value:4096
<b>Bs</b> =1 if syntax error occurred,	Bit: 13, value:8192
<b>Ba</b> =1 if over current occurred,	Bit: 14, value:16384
<b>Bk</b> =1 if EEPROM I/O error occurred,	Bit :15, value:32768

### Other Status Bit Flags:

<b>Bb</b> =1 if comm parity error occurred
<b>Bc</b> =1 if comm buffer overflow occurred
<b>Bf</b> =1 if comm framing error occurred
<b>By</b> =1 if step direction change overrun occurred (V4.40 only)

## Reset System State Flag:

<b>Za</b>	Reset (Ba) over-amps flag bit
<b>Zb</b>	Reset (Bb) comm parity flag bit
<b>Zc</b>	Reset (Bc) comm overflow flag bit
<b>Zd</b>	Reset (Bd) math overflow flag bit
<b>Zf</b>	Reset (Bf) comm framing flag bit
<b>Zl</b>	Reset (Bl) negative limit crash flag bit
<b>Zr</b>	Reset (Br) positive limit crash flag bit
<b>Zs</b>	Reset (Bs) syntax error flag bit
<b>Zu</b>	Reset (Bu) array index error flag bit
<b>Zw</b>	Reset (Bw) position wrap flag bit
<b>Zy</b>	Reset (By) step dir bit (V4.40 only)
<b>ZS</b>	Reset all reset-able system flags

## 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}** i is 8 bit value: 0 thru 255

## 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.)

## Motor Over Travel Limit Commands:

<b>UCP</b>	Assign pin C to positive limit switch input, (default state) Note: Disable with either or UCO or UCI
<b>UDM</b>	Assign pin D as negative limit switch input, (default state) Note: Disable with either or UDO or UDI
<b>The following apply to non-PLS firmware only:</b>	
<b>LIMD</b>	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.
<b>LIMN</b>	Makes Limits Non-directional. This is the default for <=V4.40c.
<b>LIMH</b>	Set Limits to active High. Motor will fault when limit goes high.
<b>LIML</b>	Set Limits active-Low, Motor will fault when limit goes low This is the default for <=V4.40c.
<b>The following apply to PLS firmware only:</b>	
<b>SLD</b>	Disable software limits (always disable prior to changing values below)
<b>SLP=expression</b>	Assign value in encoder counts to Programmable Positive Software Travel Limit
<b>SLN=expression</b>	Assign value in encoder counts to Programmable Negative Software Travel Limit
<b>SLE</b>	Enable software limits

## Motor I/O Commands:

<b>UG</b>	Assign pin G to synchronous "GO" (default State)
<b>U{pin}O</b>	Assign pin to be an output
<b>U{pin}=expression</b>	Set pin output latch to 0 or 1 where 0 is zero volts, and 1 is 5VDC
<b>U{pin}I</b>	Assign pin to be a general input
<b>var=U{pin}I</b>	Assign digital value of pin to variable (returns a 0 or 1)
<b>var=U{pins}A</b>	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] thru al[100]

**Examples:** UAI, UBO, c=UDI, UE=0, f=UGA

**Brake Commands:**

<b>BRKENG</b>	Engage the brake (requires hardware brake)
<b>BRKRLS</b>	Release the brake (requires hardware brake)
<b>BRKSRV</b>	Engage break whenever servo off (requires hardware brake)
<b>BRKTRJ</b>	Engage break when trajectory is not running (requires hardware brake)
<b>BRKC*</b>	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
<b>BRKG*</b>	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
<b>BRKI*</b>	Redirect brake control to internal brake control pin (Default state) (V4.15b or higher firmware only)

\*Note: Not available with 440c firmware (i.e. SM2315D and SM2315DT)

**PID Filter Commands:**

<b>PIDx</b>	Set PID update rate where x=1, 2, 4, or 8 (default is PID1)
<b>KA</b>	Value of buffered acceleration feed forward gain coefficient
<b>KA=expression</b>	Set buffered acceleration feed forward gain coefficient
<b>KD</b>	Value of buffered derivative gain coefficient
<b>KD=expression</b>	Set buffered PID derivative gain coefficient
<b>KG</b>	Value of buffered PID constant coefficient
<b>KG=expression</b>	Set buffered PID constant coefficient
<b>KI</b>	Value of buffered integral gain coefficient
<b>KI=expression</b>	Set buffered PID integral gain coefficient
<b>KL</b>	Value of buffered PID integral term contribution limit
<b>KL=expression</b>	Set buffered PID integral limit
<b>KP</b>	Value of buffered PID proportional gain coefficient
<b>KP=expression</b>	Set buffered PID proportional gain coefficient
<b>KS</b>	Value of buffered KS differential sample rate coefficient
<b>KS=expression</b>	Set buffered PID differential sample rate
<b>KV</b>	Value of buffered velocity feed forward gain coefficient
<b>KV=expression</b>	Set buffered PID velocity feed forward gain
<b>F</b>	Apply buffered filter coefficients to PID calculation

**Communication Commands:**

<b>ADDR=exp</b>	Set motor address between 0 and 99
<b>BAUDX</b>	Set baud rate to (x=2400, 4800, 9600, 19200, 38400 bps)
<b>SADDRaddress</b>	Set SmartMotor™ address, where address = 0 to 115
<b>ECHO</b>	Set Channel 0 (Main RS-232 Port) to Echo all received data to the transmit line
<b>ECHO_OFF</b>	Turn off Echo function above, Default state is ECHO_OFF
<b>SILENT</b>	Prohibit outgoing messages onto Channel 0, (RS-232) originating from within user program
<b>SILENT1</b>	Prohibit outgoing messages onto Channel 1, (RS-485) originating from within user program
<b>SLEEP</b>	Prohibit SmartMotor executing received Channel 0 commands except WAKE
<b>SLEEP1</b>	Prohibit SmartMotor executing received Channel 1 commands except WAKE1
<b>TALK</b>	Permit outgoing messages originating from within user program to Channel 0 (RS-232)
<b>TALK1</b>	Permit outgoing messages originating from within user program to Channel 1 (RS-485)
<b>WAKE</b>	Permit any Received Commands on Channel 0 (RS-232) to be executed
<b>WAKE1</b>	Permit any Received Commands on Channel 1 (RS-232) to be executed
<b>OCHN</b>	(type,comm,parity,bit rate,stop bits,data bits, specification)

**Open a communications channel where:**

**type** is RS2 or RS4

**comm** is either primary channel 0 or secondary channel 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)

**PRINT( )** Print to Com Ch. 0 (RS-232 main channel)

**PRINT1( )** Print to Com Ch 1 (RS-485)

**PRINT{port}( )** Print to AniLink™ port choice of A thru H

**Note: See Animatics User's Guide for more information on PRINT commands**

**GETCHR** Capture next character from Com Ch.0 input buffer

**GETCHR1** Capture next character from Com Ch.1 input buffer

**LEN** Number of characters presently in Com Ch.0 buffer

**LEN1** Number of characters presently in Com Ch.1 buffer

**Note: See Animatics User's Guide for more information on PRINT commands**

**Miscellaneous Commands:**

<b>CLK</b>	Value of SmartMotor™ clock
<b>CLK=expression</b>	Set/Reset value of SmartMotor™ clock
<b>TEMP</b>	Value of Slave processor unit temperature in degrees C. (It must be assigned to a variable to be reported.)
<b>UIA</b>	Value of motor current in 100ths of Amps (It must be assigned to a variable to be reported)
<b>UJA</b>	Value of motor DC bus Voltage in 10ths of Volts. (It must be assigned to a variable to be reported)