Applied Motion Products

Application Note

www.applied-motion.com

Subject: **Analog positioning using a Q program**
Applies to: All Q drives
Date: September 28, 2010
Author: Glenn Johnson

Contents

Overview

Many Applied Motion drives have the capability of running in analog positioning mode directly (command mode 22), but this is useful only for very short moves of no more than a few revolutions. For true linear motion systems, where the overall travel may span many hundreds of revolutions, a different approach is required.

This application note uses a Q program to smoothly control a motor over a stroke of arbitrary length.

Intended Audience, Disclaimer

Analog positioning is a fairly advanced motion control technique. It is assumed that the reader is familiar with Applied Motion equipment and has taken all appropriate safety measures during installation and wiring. Some knowledge of the Q language is preferable, as this document will not go into great detail on any specific command.

This document is intended as an example only. It is the responsibility of the user to ensure safe operation of any machine during testing.

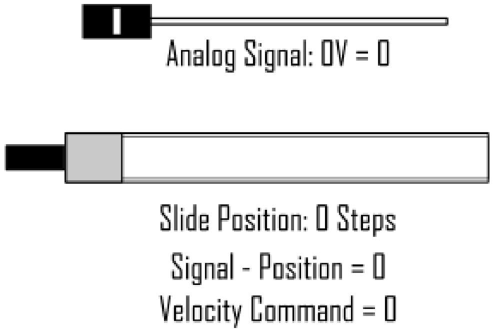
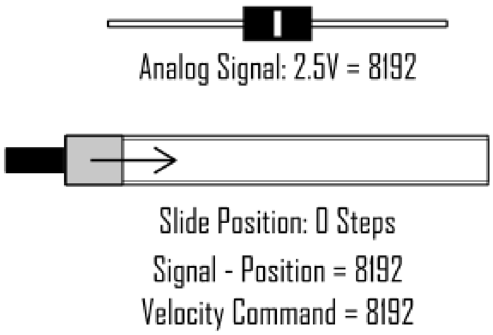
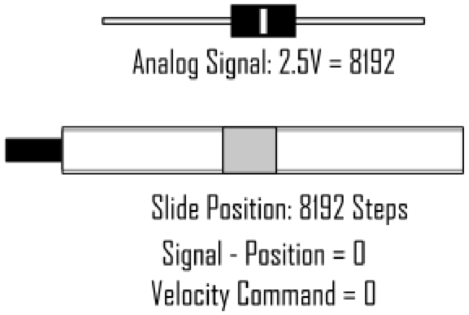
Assumptions

This document assumes that the drive and motor are properly wired and configured for use. If applicable, Quick Tuner or the appropriate Configurator software package should be used to initialize the drive for the specific motor, I/O, encoder, etc.

Limit switches should also be installed, at either end of the mechanism travel range. Besides safeguarding the machine against damage, these signals are useful during the homing routine. It is assumed that limit switches are in place and properly configured.

Note, for the purposes of this example a linear analog signal is used that generates a voltage range of 0-5V. It should be noted that the actual input circuit will accept a range of input voltages, depending on the drive used. Consult the documentation on your specific drive for details.

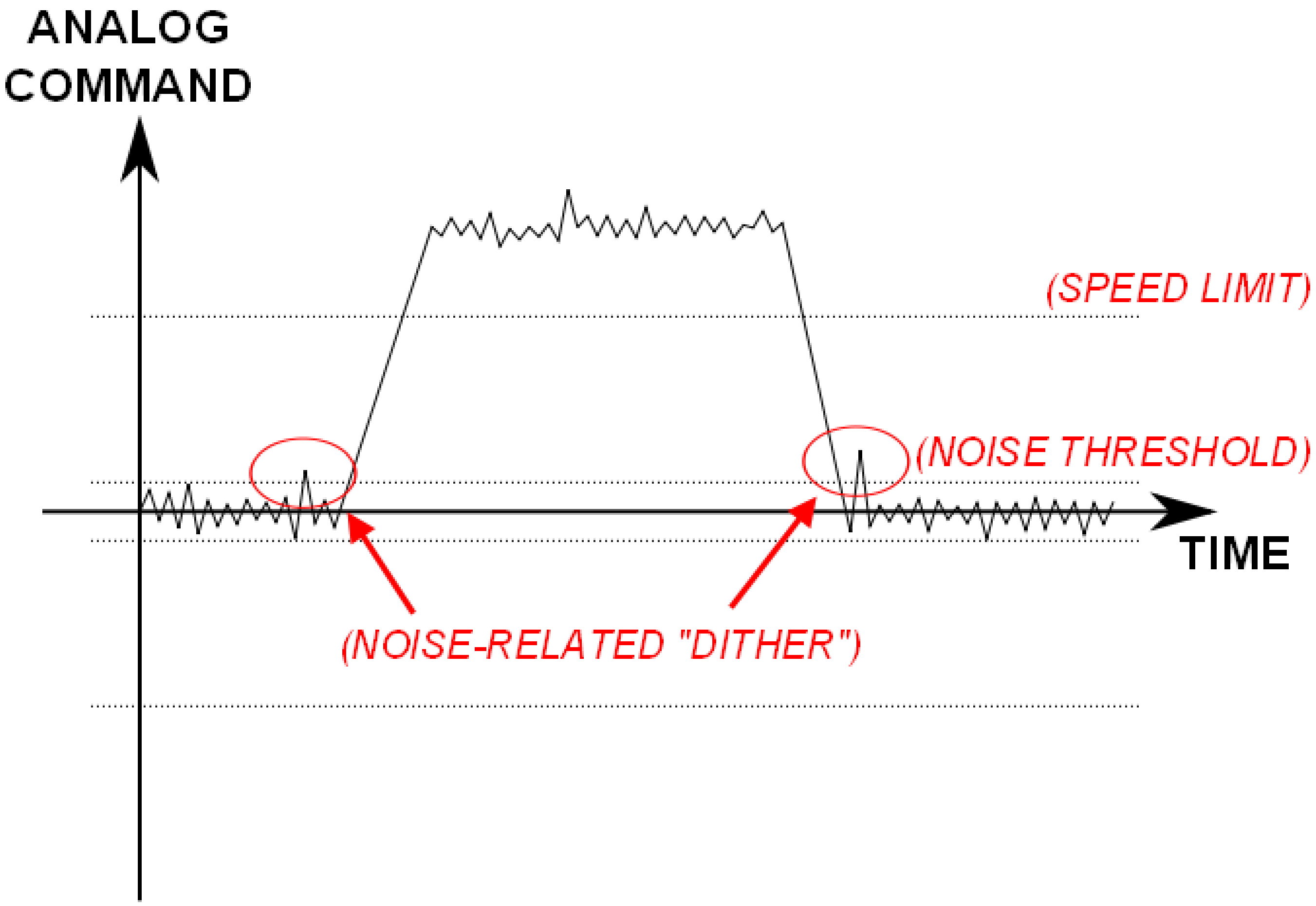
Operation Illustration

<table>
<tr>
<td>An analog slider is used, generating a signal from 0-5V. This is converted by the drive’s A/D to a range of 0-16384. Shown here at the zero position.
The motor is homed, likewise sitting at its zero position. The difference is therefore zero, and no motion occurs.</td>
<td>
</td>
</tr>
<tr>
<td>The analog slider is moved to the center of its travel, generating a 2.5V analog command. This is converted to a value of 8192 by the A/D.
The motor’s current position is still zero, so the difference between them (analog signal – motor position) is 8192.
This value is commanded to the motor as a target velocity.</td>
<td>
</td>
</tr>
<tr>
<td colspan="2">While not depicted here, the program performs two additional checks on the generated speed command at this point. See the illustration on the following page.
First, it compares against a user-defined maximum speed. The difference between analog signal command and motor position may well be larger than this value, but no number greater than the maximum speed will be commanded.
Next, a deadband calculation is performed. This ensures that the difference between analog signal and motor position is **greater** than a user-defined value before the move command is generated. This compensates for any noise on the analog signal line, eliminating the dithering effect so prevalent in traditional analog positioning algorithms.</td>
</tr>
<tr>
<td>The slider is still at its center of travel, generating a 2.5V signal interpreted by the A/D as a command of 8192.
The motor has reached its target position of 8192 counts, so the difference once again drops to zero and no further motion is commanded.</td>
<td>
</td>
</tr>
</table>

This is an illustration of the maximum speed and deadband values. The jagged line represents a noisy analog signal, and the dotted lines show the user-configurable rejection levels.

The amount of analog noise on the signal line depends on the installation site. The deadband setting should be configured to reject as much noise as possible while still allowing accurate motion.

Note that this setting also represents the smallest increment of motion, and therefore also dictates the accuracy of the system.



Configuration

There are only a few pieces of information required to adapt this program to a physical stage. You will need to know the following:

- Linear distance traveled per motor revolution. This is dependent upon either the lead of the screw or the circumference of the belt drive wheel.
- Desired motor steps/rev setting. Higher settings provide smoother travel. Typical settings of 10,000 – 20,000 steps/revolution are easily implemented, although lower values may be used. **
- Voltage swing of the analog signal. A 0-5V signal is translated to its numeric equivalent of 0-16384, while a 0-10V signal would translate to 0-32768.
- Your machine's maximum traverse speed, in revolutions/minute (RPM). Note that the actual value used in the program has the units of 0.25 RPM, so multiply your target maximum by 4.
- Deadband setting. This depends on the ambient electrical noise, and will be discussed in detail in Appendix A.

** For low step-count systems, investigate the AF command. Lowering this value will help to keep the motor travel smooth.

Calculations / Program Modification

Note that all commands are explained in detail in the Host Command Reference document, available on the Q Programmer help menu.

These modifications reference the example Q program below.

- Adjust AF, low-pass filter applied to analog input
    - Segment 2, Line 2
- Set EG
    - Segment 1, Line 3
- Set deadband, in counts (converted A/D units)
    - Segment 2, Lines 5 and 6
    - See Appendix
- Set linear scalar, increasing travel distance
    - Segment 2, Line 7
    - See Appendix

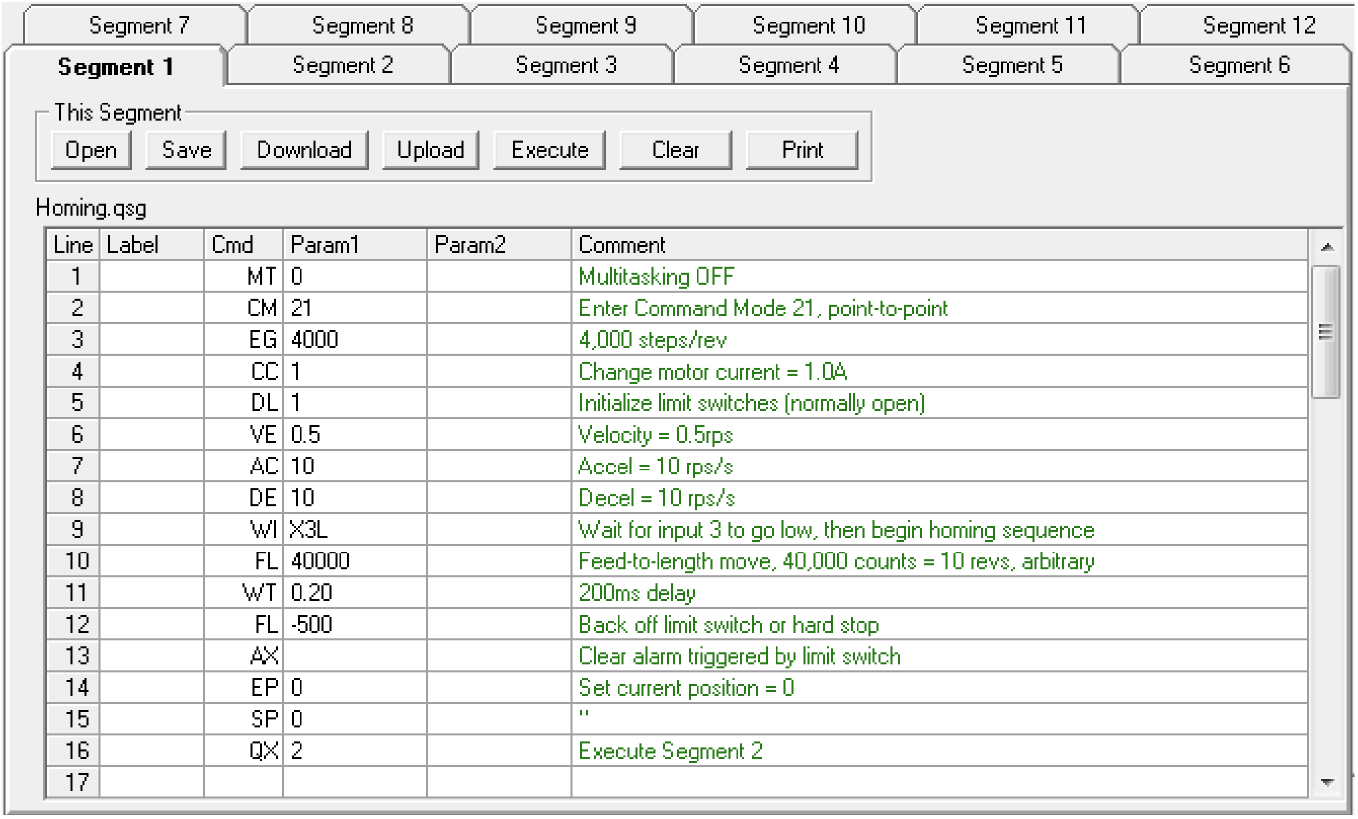
Code

This code will work on any Q drive with a properly configured analog input. It is possible to optimize this for use on servo or stepper systems by using slightly different Q commands (idle current reduction on a stepper, for example), but this is beyond the scope of this document.

For more information on the Q programming language, please visit www.applied-motion.com.

The code is divided into two segments. Segment 1 waits for an input and executes a simple homing routine.

Segment 1 – Homing



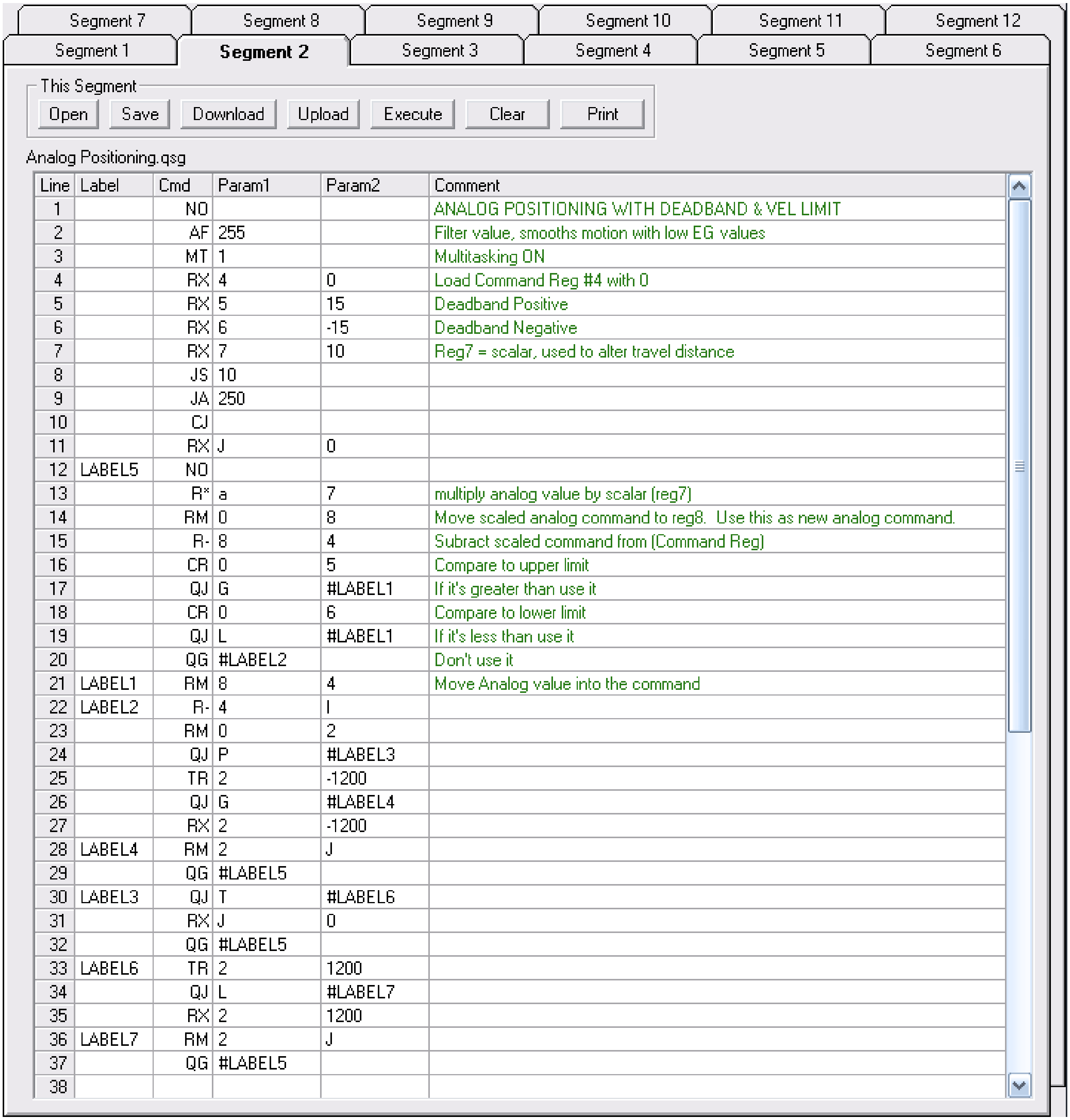
Homing.qsg

Line	Label	Cmd	Param1	Param2	Comment
1		MT	0		Multitasking OFF
2		CM	21		Enter Command Mode 21, point-to-point
3		EG	4000		4,000 steps/rev
4		CC	1		Change motor current = 1.0A
5		DL	1		Initialize limit switches (normally open)
6		VE	0.5		Velocity = 0.5rps
7		AC	10		Accel = 10 rps/s
8		DE	10		Decel = 10 rps/s
9		WI	X3L		Wait for input 3 to go low, then begin homing sequence
10		FL	40000		Feed-to-length move, 40,000 counts = 10 revs, arbitrary
11		WT	0.20		200ms delay
12		FL	-500		Back off limit switch or hard stop
13		AX			Clear alarm triggered by limit switch
14		EP	0		Set current position = 0
15		SP	0		''
16		QX	2		Execute Segment 2
17					

On a real system, the first FL command is arbitrary, but should be longer than the full stroke travel distance. This ensures that a limit switch will be tripped, stopping the move and raising an alarm condition. The second FL command moves the motor in the opposite direction to back off the limit switch. The AX command clears the end-of-travel limit alarm. The EP and SP commands set that new position as “0”, or home.

Note that a jog move is an acceptable alternative for line 10 if the system has a very long stroke. In this case lines 6, 7 and 8 must also be altered.

Segment 2 – Analog Positioning



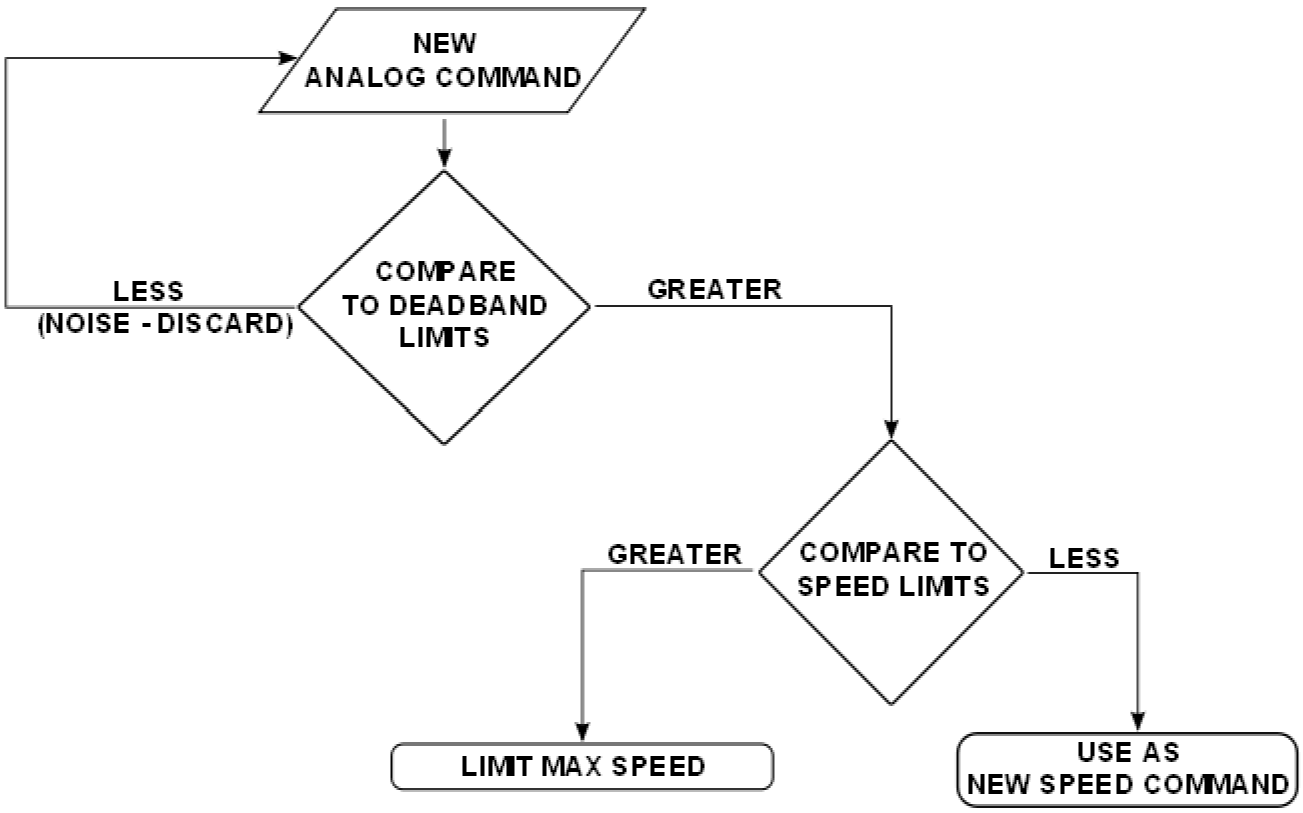
Analog Positioning.qsg

Line	Label	Cmd	Param1	Param2	Comment
1		NO			ANALOG POSITIONING WITH DEADBAND & VEL LIMIT
2		AF	255		Filter value, smooths motion with low EG values
3		MT	1		Multitasking ON
4		RX	4	0	Load Command Reg #4 with 0
5		RX	5	15	Deadband Positive
6		RX	6	-15	Deadband Negative
7		RX	7	10	Reg7 = scalar, used to alter travel distance
8		JS	10		
9		JA	250		
10		CJ			
11		RX	J	0	
12	LABEL5	NO			
13		R*	a	7	multiply analog value by scalar (reg7)
14		RM	0	8	Move scaled analog command to reg8. Use this as new analog command.
15		R-	8	4	Subract scaled command from (Command Reg)
16		CR	0	5	Compare to upper limit
17		QJ	G	#LABEL1	If it's greater than use it
18		CR	0	6	Compare to lower limit
19		QJ	L	#LABEL1	If it's less than use it
20		QG	#LABEL2		Don't use it
21	LABEL1	RM	8	4	Move Analog value into the command
22	LABEL2	R-	4	I	
23		RM	0	2	
24		QJ	P	#LABEL3	
25		TR	2	-1200	
26		QJ	G	#LABEL4	
27		RX	2	-1200	
28	LABEL4	RM	2	J	
29		QG	#LABEL5		
30	LABEL3	QJ	T	#LABEL6	
31		RX	J	0	
32		QG	#LABEL5		
33	LABEL6	TR	2	1200	
34		QJ	L	#LABEL7	
35		RX	2	1200	
36	LABEL7	RM	2	J	
37		QG	#LABEL5		
38					

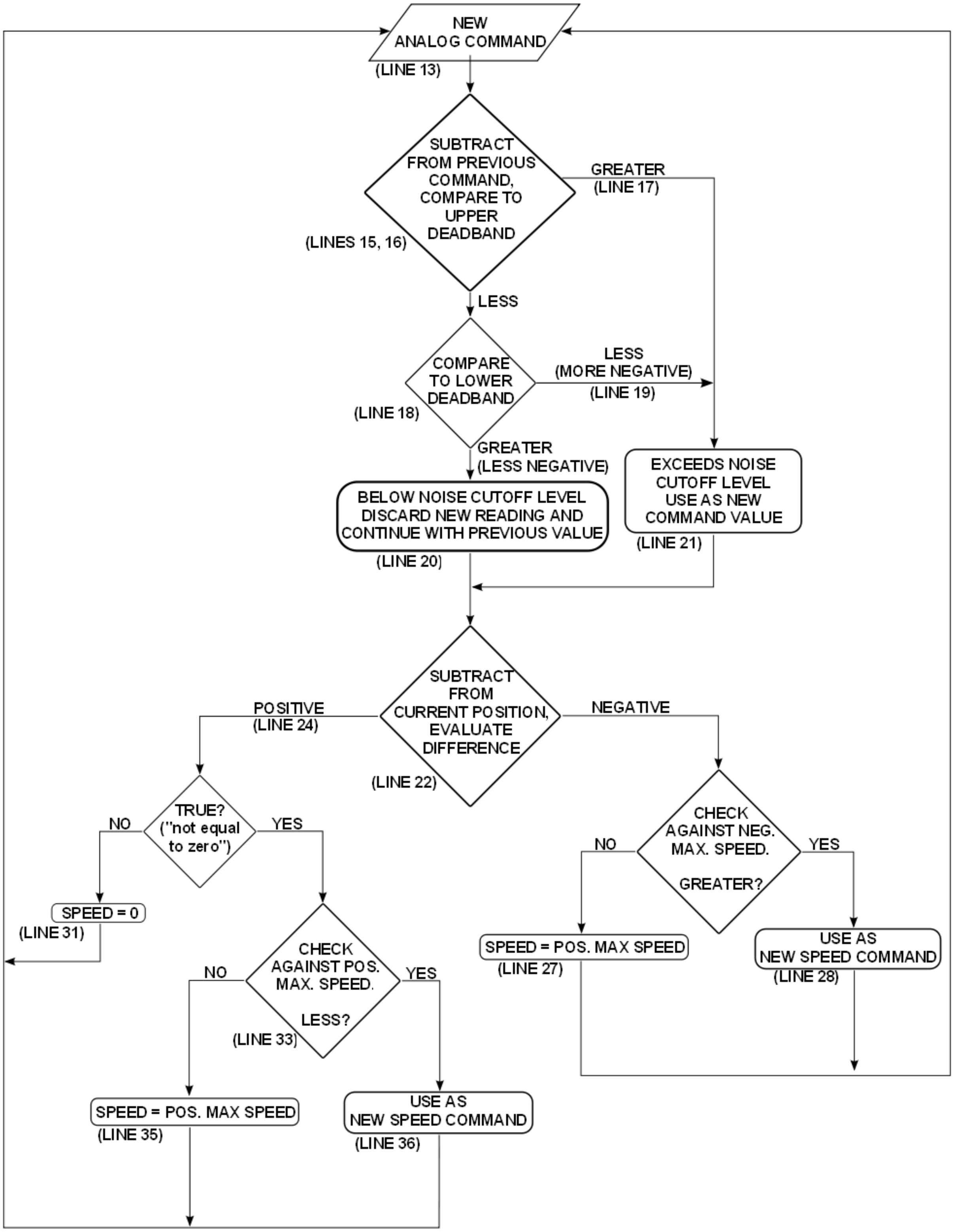
Flowcharts

These two diagrams illustrate the algorithm. A simple representation is shown below. A more complete explanation is shown on the following page.

Operational Overview Flowchart (simplified):



Operational Overview Flowchart complete):


NEW ANALOG COMMAND
(LINE 13)
SUBTRACT FROM PREVIOUS COMMAND, COMPARE TO UPPER DEADBAND
(LINES 15, 16)
GREATER
(LINE 17)
LESS
COMPARE TO LOWER DEADBAND
(LINE 18)
LESS (MORE NEGATIVE)
(LINE 19)
GREATER (LESS NEGATIVE)
BELOW NOISE CUTOFF LEVEL DISCARD NEW READING AND CONTINUE WITH PREVIOUS VALUE
(LINE 20)
EXCEEDS NOISE CUTOFF LEVEL USE AS NEW COMMAND VALUE
(LINE 21)
SUBTRACT FROM CURRENT POSITION, EVALUATE DIFFERENCE
(LINE 22)
POSITIVE
(LINE 24)
NEGATIVE
TRUE? ("not equal to zero")
NO
YES
SPEED = 0
(LINE 31)
CHECK AGAINST POS. MAX. SPEED. LESS?
(LINE 33)
NO
YES
SPEED = POS. MAX SPEED
(LINE 35)
USE AS NEW SPEED COMMAND
(LINE 36)
CHECK AGAINST NEG. MAX. SPEED. GREATER?
NO
YES
SPEED = POS. MAX SPEED
(LINE 27)
USE AS NEW SPEED COMMAND
(LINE 28)

Appendix

While the basic idea of this program is simple, the code to execute it is somewhat complex. This section should help to relate some of the variables used, streamlining the troubleshooting process.

This is a set of data calculated using EG = 4000 and an analog input of 0-5Vdc (0 – 16384).

Reg 7 analog input scalar (steps / bit)	Reg 5, 6 deadband (oscillation per mV)	total distance traveled (rev)
1	3.28	4.10
2	6.55	8.19
3	9.83	12.29
4	13.11	16.38
5	16.38	20.48
6	19.66	24.58
7	22.94	28.67
8	26.21	32.77
9	29.49	36.86
10	32.77	40.96
20	65.54	81.92
30	98.30	122.88
40	131.07	163.84
50	163.84	204.80
60	196.61	245.76
70	229.38	286.72
80	262.14	327.68
90	294.91	368.64
100	327.68	409.60
200	655.36	819.20
300	983.04	1228.80

It is actually easiest to read this table in reverse, starting with the total distance traveled column. Your machine will dictate the stroke length, which must then be converted into motor revolutions. Once known, simply assign registers 5, 6 and 7 as shown in the preceding columns.

Example:

An actuator with 30” stroke and a 4 turn/inch screw would require 120 motor revolutions to fully extend. Scanning the table we can see that there is an entry for just over 122 revs, so the user would set the registers as follows:

- **Reg 5 = 98**
- **Reg 6 = -98**
- **Reg 7 = 30**
- **EG = 4000**

Tips:

- The table above calculates values for registers 5 and 6 in terms of oscillation *per mV of analog noise*. It may be necessary to increase these values by a factor of 2-3 or more, depending on the amount of ambient electrical noise in the installation area and the total move distance.

- The deadband (Reg 5 and 6) actually define the smallest increment of motion. This directly impacts accuracy.

  For example, let’s go back to our example system above. With a deadband of 98 – (-98) = 196, the motor’s effective step size is now 196 counts. Determine minimum angular displacement and linear travel as follows:

  Angular Displacement = (deadband / EG) * 360
  = (196 / 4000) *360
  = 17.64 degrees

  Linear Travel = (displacement / 360) * travel per revolution
  = (17.64 / 360) * 0.25 in
  = 0.01225 in

  Thus, the settings above will give us smooth travel over 30” with a minimum step of 0.01225 in. This system will reject analog line noise below 1mV.

  To reject noise of 2mV, double the deadband value. Overall travel is unaffected, but the smallest step size grows to 0.0245 in.