Application Note #56:
Testing Inputs

Functional Description

Skeleton Code Name: Testing an Input to Execute a Desired Action

Software: Q Programmer

Hardware: ST10-Q Stepper Drive

Reference: [Host Command Reference Manual]

- Commands
  - TI - Test Input
  - QJ - Queue Jump
  - NO - No Operation
  - QG – Queue Go

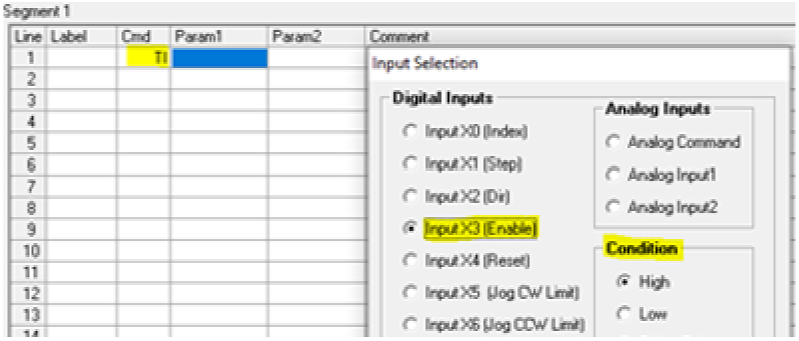
Notes:

- Q Programmer does NOT accept empty command lines, so NO is used in lieu of that.
- NO commands can help segment code for readability.
- If inputs are used for purposes other than their default purpose, they should be set to General Purpose during driver configuration.

Goals

Goal 1: How to command the driver to test an input

a. Use the Test Input (TI) command to command the driver to test desired input
   i. TI’s Param1 value must be the desired input number and the condition to test for e.g., 3H for Input 3, HIGH condition. This test yields TRUE or FALSE.



Segment 1

Line	Label	Cmd	Param1	Param2	Comment
1		TI			
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

Input Selection

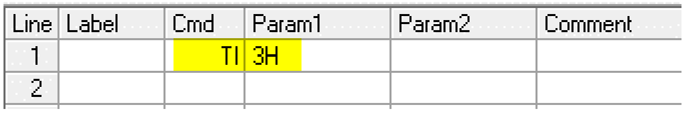
Digital Inputs
○ Input X0 (Index)
○ Input X1 (Step)
○ Input X2 (Dir)
● Input X3 (Enable)
○ Input X4 (Reset)
○ Input X5 (Jog CW Limit)
○ Input X6 (Jog CCW Limit)

Analog Inputs
○ Analog Command
○ Analog Input1
○ Analog Input2

Condition
● High
○ Low

Figure 1 Configuring TI Command

b. The final command should look as follows:



Line	Label	Cmd	Param1	Param2	Comment
1		TI	3H		
2					

Figure 2 Test Input 3 for HIGH State

c. The command above tests input 3 for HIGH (Open) state.

Goal 2: Making sure tested condition affects program execution

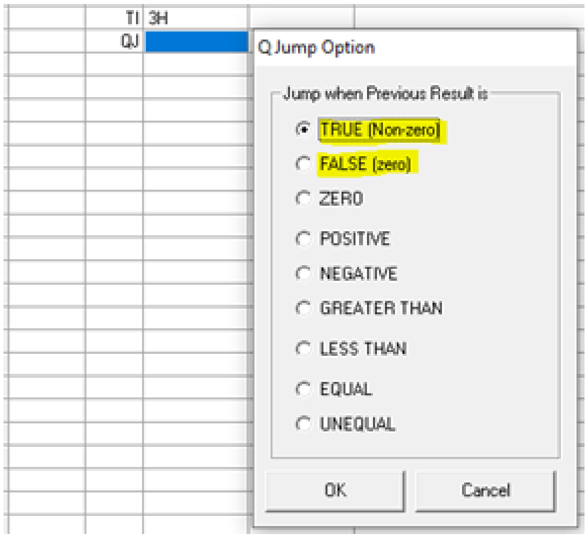
a. Using TI alone is not enough for the purpose of affecting program flow. TI simply tells the driver to read for the desired condition at an input. What we want, based on whether a condition is met (TRUE/FALSE), is to change the program flow to execute a specific action outside of the normal program flow. To do that, we command the driver to jump to the desired action when the desired state (TRUE/FALSE) of the tested input condition is met.

   i. NOTE: If we do not command the driver to jump, the program will simply keep running sequentially, executing the lines directly after the TI command.

b. To jump to desired action, use the Queue Jump (QJ) command after TI. For its Param1, it accepts the desired TRUE/FALSE state of tested input. For its Param2, it accepts the line # at which the code to be executed begins at.

   i. NOTE: Once a line # is assigned to QJ, Q Programmer automatically assigns a label to that line. Let Q Programmer assign the labels as it can dynamically change them for you if needed.

   ii. For example, if you tested the driver for Input 3 High (3H), but you only want the driver to jump to a different code snippet if the condition was FALSE, the set up would be as follows. Here, we jump to Line 6 to execute our 'code':



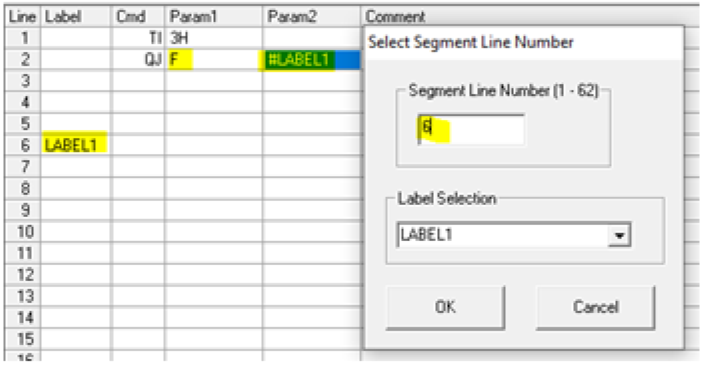
Cmd	Param1
TI	3H
QJ	

Q Jump Option

Jump when Previous Result is

- ● TRUE (Non-zero)
- ○ FALSE (zero)
- ○ ZERO
- ○ POSITIVE
- ○ NEGATIVE
- ○ GREATER THAN
- ○ LESS THAN
- ○ EQUAL
- ○ UNEQUAL

Figure 3 Configure QJ Command



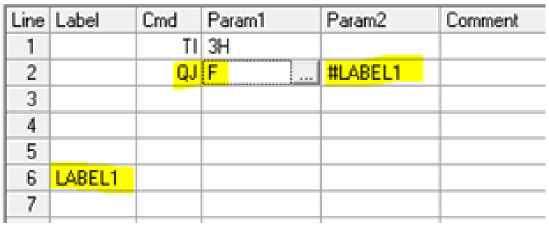
Line	Label	Cmd	Param1	Param2	Comment
1		TI	3H		
2		QJ	F	#LABEL1	
3					
4					
5					
6	LABEL1				
7					
8					
9					
10					
11					
12					
13					
14					
15					

Select Segment Line Number

Segment Line Number (1 - 62): 6

Label Selection: LABEL1

Figure 4 Select Line to Jump to



Line	Label	Cmd	Param1	Param2	Comment
1		TI	3H		
2		QJ	F	#LABEL1	
3					
4					
5					
6	LABEL1				
7					

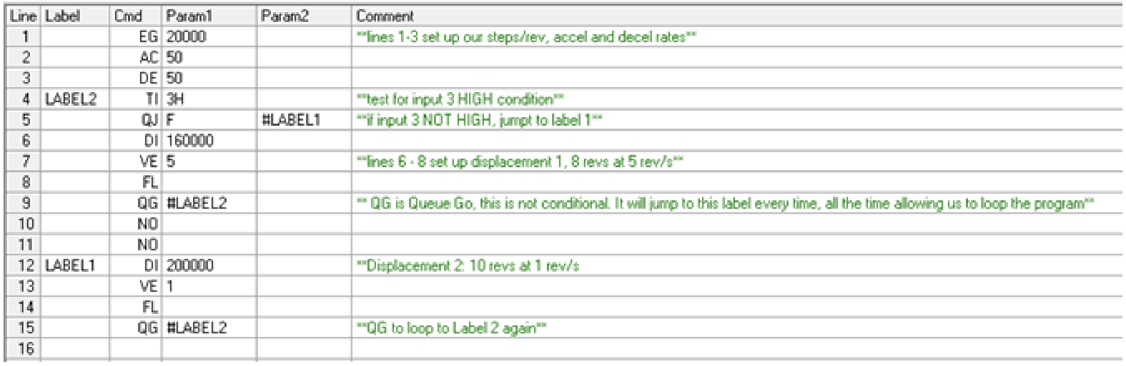
Figure 5 Conditional Jump to Label 1

iii. If Input 3 is HIGH, the program continues to Line 3. If Input 3 is LOW, then the program jumps to Line 6.

iv. If Param1 for QJ command were to be changed to True (T), we switch the logic. Meaning, if Input 3 is HIGH, the program would jump to Line 6. If Input 3 is LOW, then the program continues to Line 3.

Goal 3: Using the first two goals, execute a simple relative motion program that executes one of two displacements based on an input

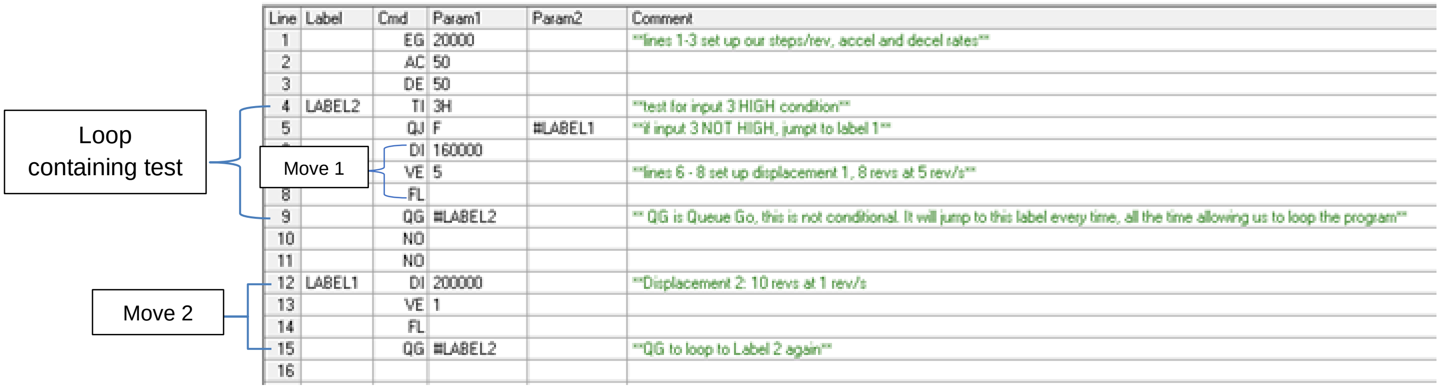
a. The following code snippet can execute one of two moves. Using TI 3H as our test, move 1 will occur if Input 3 is HIGH and move 2 will *only* occur if Input 3 is LOW. For testing purposes, the program will loop continually to test for 3H (Input 3 = HIGH). Focus on Lines 4-9 and 12-15. Lines 1-3 simply configure some parameters.



Line	Label	Cmd	Param1	Param2	Comment
1		EG	20000		**lines 1-3 set up our steps/rev, accel and decel rates**
2		AC	50		
3		DE	50		
4	LABEL2	TI	3H		**test for input 3 HIGH condition**
5		QJ	F	#LABEL1	**if input 3 NOT HIGH, jumpt to label 1**
6		DI	160000		
7		VE	5		**lines 6 - 8 set up displacement 1, 8 revs at 5 rev/s**
8		FL			
9		QG	#LABEL2		** QG is Queue Go, this is not conditional. It will jump to this label every time, all the time allowing us to loop the program**
10		NO			
11		NO			
12	LABEL1	DI	200000		**Displacement 2: 10 revs at 1 rev/s
13		VE	1		
14		FL			
15		QG	#LABEL2		**QG to loop to Label 2 again**
16					

Figure 5 Full Solution

b. Note how Label 1 is outside of the loop (Lines 4-9, Label 2) that contains the original test input command. If Label 1 code (Lines 11-15) were kept inside the Label 2 loop (Lines 4 – 9) and was inserted between Lines 8 – 9, the code of Label 1 would execute regardless of whether input 3 is HIGH or LOW. This is why it is important that we create a break in flow between our regular program and our input handling code.



Line	Label	Cmd	Param1	Param2	Comment
1		EG	20000		**lines 1-3 set up our steps/rev, accel and decel rates**
2		AC	50		
3		DE	50		
4	LABEL2	TI	3H		**test for input 3 HIGH condition**
5		QJ	F	#LABEL1	**if input 3 NOT HIGH, jumpt to label 1**
6		DI	160000		
7		VE	5		**lines 6 - 8 set up displacement 1, 8 revs at 5 rev/s**
8		FL			
9		QG	#LABEL2		** QG is Queue Go, this is not conditional. It will jump to this label every time, all the time allowing us to loop the program**
10		NO			
11		NO			
12	LABEL1	DI	200000		**Displacement 2: 10 revs at 1 rev/s
13		VE	1		
14		FL			
15		QG	#LABEL2		**QG to loop to Label 2 again**
16					

Figure 6 Solution Breakdown

Click here for the Sample Code.

Try it out today!

If you have questions about this or any other application, please reach out to our Apps Engineering Group for any assistance at 1-800-525-1609 or support@applied-motion.com.