

# Brushless IDEA™ Drive

Communication Manual

PBL4850E



[www.haydonkerk.com](http://www.haydonkerk.com)

All Rights Reserved

3-2015

## Table of Contents

Revision History .....	4
Introduction.....	5
IDEA Drive Communications Basics .....	6
Commands .....	7
<i>Abort</i> .....	8
<i>Check Password</i> .....	8
<i>Comment</i> .....	8
<i>Configure Encoder</i> .....	8
<i>Configure Input Interrupts</i> .....	9
<i>E-Stop</i> .....	9
<i>Execute Program</i> .....	10
<i>Go At Speed</i> .....	10
<i>Goto</i> .....	11
<i>Goto If</i> .....	11
<i>Goto Sub</i> .....	11
<i>Index</i> .....	12
<i>Interrupt on Position</i> .....	13
<i>Jump N Times</i> .....	13
<i>Label</i> .....	13
<i>Move To Position</i> .....	14
<i>No-op</i> .....	14
<i>Program</i> .....	15
<i>Read Control Gain</i> .....	15
<i>Read Current Position and Speed</i> .....	15
<i>Read Encoder Settings</i> .....	16
<i>Read Executing</i> .....	16
<i>Read Faults</i> .....	16
<i>Read Feedback Configuration</i> .....	17
<i>Read Firmware Version</i> .....	17
<i>Read IO</i> .....	17
<i>Read Max Current</i> .....	18
<i>Read Motor Parameters</i> .....	18
<i>Read Move Profile</i> .....	18
<i>Read Moving</i> .....	18
<i>Read Program Names</i> .....	19
<i>Read Startup Program</i> .....	19
<i>Recall Program</i> .....	19
<i>Remove Password</i> .....	19
<i>Remove Program</i> .....	20
<i>Restore Factory Defaults</i> .....	20
<i>Return</i> .....	20
<i>Return To</i> .....	20
<i>Run Program</i> .....	21
<i>Set Control Loop Gain</i> .....	21
<i>Set Feedback Configuration</i> .....	21

<i>Set Motor Parameters</i> .....	22
<i>Set Move Profile</i> .....	22
<i>Set Outputs</i> .....	22
<i>Set Password</i> .....	23
<i>Set Position As</i> .....	23
<i>Set Startup Program</i> .....	23
<i>Software Reset</i> .....	23
<i>Stop</i> .....	24
<i>Wait For Move</i> .....	24
<i>Wait Time</i> .....	25

## Revision History

<b>Date</b>	<b>Description</b>
October 2010	Initial release
January 2011	Added "Execute Program" command.
May 2011	Corrected response from Program command
September 2011	Added information about faults Added Read Moving command Updated configure encoder command Alphabetized commands
December 2011	Corrected configure encoder example
April 2013	Corrected program description Corrected table of contents
March 2015	Revised manual for brushless drive

## Introduction

The communication structure was initially developed for the IDEA stepper motor drive. With the introduction of the brushless drive to the IDEA lineup, it was our intention to maintain the communication structure for the brushless version. In this manner, our users which are familiar with programming the stepper drives have a smooth transition into the brushless environment. As the two driving methods vary greatly, the commands for the brushless drive may not use some of the parameters implemented in the stepper version.

## IDEA Drive Communications Basics

The IDEA drive line of products are commanded through the use of an Ascii based language developed by Haydon Kerk. Each command consists of a character identifying the command, followed by between 0 and 12 parameters separated by commas, and then followed by a carriage return. One difference between this language and those used by competing products is that each motion command encapsulates all parameters needed by the move; there are no parameters to set before a move command is issued. While this makes manual entry of commands into a terminal cumbersome, this is not the intended use of the language. Creation of these commands can be done simply in the software of the controller used to command the drives.

The IDEA drive adheres to a master/slave communications model. The master controller initiates all communications. If information is required from the drive, as in the case of requesting the drive's current position, the controller first sends the command requesting the drive's position, then the drive responds with the requested information, enclosed by several characters to identify the response. The extra characters can then be parsed, and the response read.

ONE MAJOR DIFFERENCE BETWEEN USING THIS COMMAND SET TO CONTROL THE DRIVE, AND USING THE IDEA DRIVE USER INTERFACE IS, THERE ARE NO PROTECTIONS WHEN USING THE COMMAND LANGUAGE. THE USER INTERFACE ENSURES THAT BASED UPON THE MOTOR PARAMETERS ENTERED, NO IMPROPER VALUES ARE SENT TO THE DRIVE; WITH THIS COMMAND SET, IT IS THE RESPONSIBILITY OF THE USER TO ENSURE THAT NO DAMAGE IS DONE TO THE DRIVE, MOTOR, OR OTHER EQUIPMENT THROUGH THE INCORRECT USE OF COMMANDS.

The parameters for serial communication are as follows:

Bits per Second: 57600

Data bits: 8

Parity: none

Stop Bits: 1

Flow Control: None

## Commands

The following describes the commands that make up the IDEA drive communications language, as well as the format for any response required from the drive. When quotation marks are present, the text in between the quotation marks is the important string, and the quotation marks themselves should not be included. When [cr] is shown, it is referring to the Ascii carriage return character, not to be confused with a line feed character. When [parameter] is shown, where parameter is the name of a parameter, it is representing some variable with that name, and the brackets will not be part of the string.

The contexts listed below indicate when each command can be used. Realtime commands can only be executed by direct command to the drive, such as requesting the current position. Program commands can only be a part of a program, and are generally branching or similar commands, such as Goto. Realtime/Program commands can be used anytime, and are generally motion related commands, such as Index. For further explanation of the commands, refer to the IDEA drive users' manual.

The following commands are used to configure the drive per the connected motor and will have an impact on drive performance. These parameters are saved to non-volatile drive memory and will only need to be set each time a new motor is used with the drive.

- Set Motor Parameters
- Set Feedback Configuration
- Set Control Loop Gain
- Set Move Profile Type

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Abort</i>	A	Realtime/Program	none	None
<b>Description</b>	This command causes the drive to immediately stop, and ends the execution of any programs.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	You want to stop all drive activity.			
<b>Command</b>	"A" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Check Password</i>	c	Realtime	Password	"`cYES[cr]`c#[cr]" or "`cNO[cr]`c#[cr]"
<b>Description</b>	This command checks to see if a password is the correct password.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Password	The password in question.			A string, exactly 10 characters long
<b>Example</b>	You want to check if the password is "password ".			
<b>Command</b>	"cpassword " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Comment</i>	C	Program	Comment	None
<b>Description</b>	This command creates a comment in the program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Comment	A string, must be exactly 10 characters long.			
<b>Example</b>	You want to add a comment that says "Extend 1in".			
<b>Command</b>	"CExtend 1in" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Configure Encoder</i>	z	Realtime/Program	0, 0, 0, 0, Encoder CPR, 1	None
<b>Description</b>	This command configures the encoder.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
N/A	Must be zero			0
N/A	Must be zero			0
N/A	Must be zero			0
N/A				0
Encoder Resolution	The resolution of the encoder being used in cycles per revolution			50 to 25000
N/A	Must be a value of 1			1
<b>Example</b>	You have an encoder with a cycles per revolution value of 1000			
<b>Command</b>	"z0,0,0,0,1000,1" followed by a carriage return.			



<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Configure Input Interrupts</i>	<b>i</b>	Program	Input1 config, input2 config, input3 config, input4 config, input1 destination, input2 destination, input3 destination, input4 destination, input1 priority, input2 priority, input3 priority, input4 priority	None
<b>Description</b>	This command is used to configure the interrupt settings for the inputs.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Config	What kind of interrupt the input should be. 1 for Falling edge, 2 for rising edge, 3 for both edges, 0 for disabled.			0,1,2,3
Destination	The address of the subroutine that should handle the interrupt.			0 to 87036, multiples of four only.
Priority	The priority of the interrupt; lower numbered priorities are handled first.			0 to 4
<b>Example</b>	You want to set a rising edge interrupt on input 2, whose destination is address 512 and priority is 1, and all other input interrupts disabled.			
<b>Command</b>	"i0,2,0,0,0,512,0,0,4,1,4,4" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>E-Stop</i>	<b>E</b>	RealTime/Program	Decel Current, Hold Current, Delay Time	None
<b>Description</b>	This command stops the motor without decelerating.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Decel Current	The maximum peak current, in milliamps, used to stop the motor.			0 to 6500
Hold Current	The maximum peak current, in milliamps, for after the motor has stopped.			0 to 5656
Delay Time	Time, in milliseconds, of the transition from the deceleration current to the hold current			10 - 300
<b>Example</b>	You wish to immediately stop the motor with a maximum decel current of 2.0 Apeak, then apply a maximum hold current of 1.0 Apeak to keep the load at position after 50 milliseconds.			
<b>Command</b>	"E2000,1000,50" followed by a carriage return			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Execute Program</i>	<b>m</b>	Realtime	Program name	None
<b>Description</b>	This command begins the execution of a program without changing the state of the outputs or motor.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Program Name	The name of the program to run.			A string, exactly 10 characters long
<b>Example</b>	You want to run a program named "program 1 ", without returning to the default state.			
<b>Command</b>	"mprogram 1 " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Go At Speed</i>	<b>Q</b>	RealTime/Program	Speed, Start Speed, End Speed, Accel, Decel, Run Current, Hold Current, Accel Current, Decel Current, Delay Time, 1	None
<b>Description</b>	This command accelerates the motor up to a specified speed, with the given parameters.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Run Speed	The number of encoder counts per second the motor should move at the top speed			0 > to 16777215
Start Speed	Must be a value of zero			0
End Speed	Must be a value of zero			0
Accel Rate	Rate at which the speed should rise from Start Speed to the Run Speed.			0 > to 16777215
Decel Rate	Rate at which the speed should fall from the Run Speed to End Speed.			0 > to 16777215
Run Current	The maximum peak current, in milliamps for the move.			0 > to 5656
Hold Current	The maximum peak current, in milliamps, for after the move has completed.			0 > to 5656
Accel Current	The maximum peak current, in milliamps, for the acceleration portion of the move.			0 > to 6500
Decel Current	The maximum peak current, in milliamps, for the deceleration portion of the move.			0 > to 6500
Delay Time	Time, in milliseconds, of the transition from the deceleration current to the hold current			10 - 300
N/A	Must be a value of 1			1
<b>Example</b>	Desired move backwards at a speed of 3200 counts per second, accelerating at a rate of 40000 counts per second <sup>2</sup> , decelerating at a rate of 100000 counts per second <sup>2</sup> , with a maximum run current of 1.6 Apeak, a maximum accel current of 1.9 Apeak, a maximum decel current of 2.0 Apeak, and changing to a maximum hold current of 1.0 Apeak after a 50 millisecond delay.			
<b>Command</b>	"Q-3200,0,0,40000,100000,1600,1000,1900,2000,50,1" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Goto</i>	<b>G</b>	Program	Destination	None
<b>Description</b>	This command causes the program to continue execution at the specified address.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Destination	The address of the command that should be run			0 to 86012, multiples of four only. Must be the address of a valid command.
<b>Example</b>	You want to continue execution at address 1024.			
<b>Command</b>	"G1024" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>													
<i>Goto If</i>	<b>L</b>	Program	Destination, Condition	None													
<b>Description</b>	This command causes the program to continue execution at the specified address if the condition is met.																
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>													
Destination	The address of the command that should be run.			0 to 86012, multiples of four only. Must be the address of a valid command.													
Condition	2 bytes indicating which I/O are tested, and the test values for each. The least significant byte corresponds to the inputs, and the most significant byte corresponds to the outputs. For each byte, the least significant nibble represents the condition being tested, a 1 meaning a high input or output, and a 0 representing a low input or output. The more significant nibble decides which of those conditions are to be tested, with a 1 representing an input or output should be tested. The least significant bit corresponds to input1, the next to input 2, and so on.			0 to 65535													
<b>Example</b>	You want to continue execution at address 1024 if input 2 is high.																
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Total	
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0		34
<b>Command</b>	"L1024, 34" followed by a carriage return.																

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Goto Sub</i>	<b>S</b>	Program	Destination	None
<b>Description</b>	This command causes the program to execute the subroutine at the given destination.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Destination	The address of the subroutine that should be run.			0 to 86012, multiples of four only. Must be the address of a valid command.
<b>Example</b>	You want to run a subroutine at address 1024.			
<b>Command</b>	"S1024" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Index</i>	I	RealTime/Program	Distance, Speed, Start Speed, End Speed, Accel, Decel, Run Current, Hold Current, Accel Current, Decel Current, Delay Time, 1	None
<b>Description</b>	This command moves the motor forward or backwards a defined number of counts, with the given parameters.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Distance	The positive or negative number of counts the motor should move.			-18446744073709551616 to 18446744073709551615
Run Speed	The number of counts per second the motor should move at the top speed.			0 > to 16777215
Start Speed	Must be set to zero			0
End Speed	Must be set to zero			0
Accel Rate	Rate at which the speed should rise from Start Speed to the Run Speed.			0 > to 16777215
Decel Rate	Rate at which the speed should fall from the Run Speed to the End Speed.			0 > to 16777215
Run Current	The maximum peak current, in milliamps for the move.			0 > to 5656
Hold Current	The maximum peak current, in milliamps, for after the move has completed.			0 > to 5656
Accel Current	The maximum peak current, in milliamps, for the acceleration portion of the move.			0 > to 6500
Decel Current	The maximum peak current, in milliamps, for the deceleration portion of the move.			0 > to 6500
Delay Time	Time, in milliseconds, of the transition from the deceleration current to the hold current			50 - 300
N/A	Must be a value of 1			1
<b>Example</b>	Desired move is backwards 9600 counts per second, at a speed of 3200 counts per second, accelerating at a rate of 40000 counts per second <sup>2</sup> , decelerating at a rate of 100000 counts per second <sup>2</sup> , with a maximum run current of 1.6 Apeak, a maximum accel current of 1.9 Apeak, a maximum decel current of 2.0 Apeak, and a maximum hold current of 1.0 Apeak after a delay time of 50 milliseconds.			
<b>Command</b>	"I-9600,3200,0,0,40000,100000,1600,1000,1900,2000,50,1" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Interrupt on Position</i>	T	Program	Position, Destination, Priority	None
<b>Description</b>	This command sets an interrupt to occur at a given position.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Position	The position where the interrupt should be triggered.			-18446744073709551616 to 18446744073709551615
Destination	The address of the subroutine to be run when the interrupt is triggered.			0 to 86012, multiples of four only. Must be the address of a valid command.
Priority	The priority of the interrupt; lower values are a higher priority.			0 to 4, 10 to disable
<b>Example</b>	You want to set a trip point at position 0, that runs a subroutine at address 1024, and has the highest priority.			
<b>Command</b>	"T0,1024,0" followed by a carriage return			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Jump N Times</i>	J	Program	Destination, Jumps	None
<b>Description</b>	This command causes the program to continue execution at the specified address a specified number of times.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Destination	The address of the command that should be run.			0 to 86012, multiples of four only. Must be the address of a valid command.
Jumps	The number of times execution should branch to the destination address.			0 to 65535
<b>Example</b>	You want to continue execution at address 1024, and do so 3 times.			
<b>Command</b>	"J1024, 3" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Label</i>	B	Program	Label name	None
<b>Description</b>	This command creates a label in the program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Label Name	A string, must be exactly 10 characters long.			
<b>Example</b>	You want to add a label called "Start".			
<b>Command</b>	"BStart " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Move To Position</i>	<b>M</b>	RealTime/Program	Position, Speed, Start Speed, End Speed, Accel, Decel, Run Current, Hold Current, Accel Current, Decel Current, Delay Time, 1	None
<b>Description</b>	This command moves the motor to a position, with the given parameters.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Position	The positive or negative position, in counts, the motor should move to.			-18446744073709551616 to 18446744073709551615
Run Speed	The number of counts per second the motor should move at the top speed			0 > to 16777215
Start Speed	Must be a value of zero			0
End Speed	Must be a value of zero			0
Accel Rate	Rate at which the speed should rise from Start Speed to the Run Speed.			0 > to 16777215
Decel Rate	Rate at which the speed should fall from the Run Speed to End Speed.			0 > to 16777215
Run Current	The maximum peak current, in milliamps for the move.			0 > to 5656
Hold Current	The maximum peak current, in milliamps, for after the move has completed.			0 > to 5656
Accel Current	The maximum peak current, in milliamps, for the acceleration portion of the move.			0 > to 6500
Decel Current	The maximum peak current, in milliamps, for the deceleration portion of the move.			0 > to 6500
Delay Time	Time, in milliseconds, of the transition from the deceleration current to the hold current			50 - 300
N/A	Must be a value of 1			1
<b>Example</b>	Desired move is to position 0, at a speed of 3200 counts per second, accelerating at a rate of 40000 counts per second <sup>2</sup> , decelerating at a rate of 100000 counts per second <sup>2</sup> , with a maximum run current of 1.6 Apeak, a maximum accel current of 1.9 Apeak, a maximum decel current of 2.0 Apeak, and changing to a maximum hold current of 1.0 Apeak after a 50 millisecond delay.			
<b>Command</b>	"M0,3200,0,0,40000,100000,1600,1000,1900,2000,50,1" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>No-op</i>	<b>w</b>	Program	none	None
<b>Description</b>	This command is used to insert an extra line in a program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	This command would be used in a custom user interface.			
<b>Command</b>	"w" followed by a carriage return.			

<i>Program</i>	P	Realtime	(Program Name, Start Location, Length) or none	None or "P[Program size][CR]P#[CR]"
<b>Description</b>	This command starts and ends the process of writing a program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Program Name	The name for the program, if it is the same as a program already on the drive, the old program will be removed.			A string; must be exactly 10 characters.
Start Location	The page number where the program should begin. If the program overlaps with any other program, the old program will be deleted. Each page has 1024 bytes of space.			1 to 75
Length	The number of pages the program will take up.			1 to 75
<b>Example</b>	You want to write a program name program 1, on the first page of memory with a program length of less than of 1 page.			
<b>Command</b>	"Pprogram 1 , 1,1" followed by a carriage return. Then followed by the commands that make up the program, each separated by a carriage return, followed by "P" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Control Gain</i>	)	Realtime	None	"")[Kps],[Kis],[Kds],[Kpp],[Ktff][cr]`)#[cr]" where [value] is a number.
<b>Description</b>	This command reads the gains of the control loop used to control the motor.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check settings of the various gains which make up the drives' control loop. Gains are in the following order: Velocity loop proportional gain, velocity loop integral gain, velocity loop derivative gain, positional loop proportional gain, and current loop feed forward gain.			
<b>Command</b>	")" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Current Position and Speed</i>	I	Realtime	None	"I[position],[position error],[velocity],[velocity error][cr]I#[cr]"
<b>Description</b>	This command requests the current position of the motor, the position tracking error, the current velocity of the motor, and the velocity tracking error			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check the position of the encoder, the position error, the current speed of the motor, and the error in speed			
<b>Command</b>	"I" followed by a carriage return.			

<i>Read Encoder Settings</i>	<b>b</b>	Realtime	None	"`b[NA],[NA],[Encoder CPR][cr]`b#[cr]"
<b>Description</b>	This command requests the encoder configuration of the drive.			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
None				
<b>Example</b>	You want to check the encoder settings on the drive.			
<b>Command</b>	"b" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Executing</i>	<b>r</b>	Realtime	None	"`rYES[cr]`r#[cr]" or "`rNO[cr]`r#[cr]"
<b>Description</b>	This command requests whether the drive is actively running a program.			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
None				
<b>Example</b>	You want to check if the drive is executing a program.			
<b>Command</b>	"r" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>				
<i>Read Faults</i>	<b>f</b>	Realtime	None	"`f[value][cr]`f#[cr]" where value represents the errors present. Each bit represents a specific error, as defined below.				
<b>Description</b>	This command requests the error status of the drive.							
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>					
None								
<b>Example</b>	You want to check the error status of the drive.							
<b>Command</b>	"f" followed by a carriage return.							
Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Over Speed	Bad Checksum	Current Limit	Loop Overflow	Int Queue Full	Encoder Error	Temperature	Stack Overflow	Stack Underflow



<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Feedback Configuration</i>	>	Realtime	None	">[hall cell],[commutation],[encoder],[complement],[Rotor Position][cr]`>#[cr]" where [value] is a number.
<b>Description</b>	This command requests hall cell spacing, commutation direction, encoder direction, and complement setting of the drive.			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
None				
<b>Example</b>	You want to check the hall cell spacing, commutation direction, encoder direction, and complement setting of the drive.			
<b>Command</b>	">" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Firmware Version</i>	v	Realtime	None	"`v[value][cr]`v#[cr]" where [value] is a number.
<b>Description</b>	This command requests the firmware version of the drive.			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
None				
<b>Example</b>	You want to check the firmware version on the drive.			
<b>Command</b>	"v" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>				
<i>Read IO</i>	:	Realtime	none	"`:[value][CR]`#[CR]", Where [value] is a number between 0 and 255, formed from 1 byte, with ones being highs, zeros being lows, the most significant bit corresponding to output4, and the least significant bit corresponding to input1.				
<b>Description</b>	This command requests the status of the inputs and outputs.							
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>					
none								
<b>Example</b>	Want to know the status of the input and outputs. For this example, outputs 1 and 2 will be high, and inputs 2, 3, and 4 will be high, all others will be low.							
<b>Command</b>	":#" followed by a carriage return.							
Output4	Output 3	Output 2	Output 1	Input 4	Input 3	Input 2	Input 1	Value
0	0	1	1	1	1	1	0	62

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Max Current</i>	j	Realtime	None	"j[value][cr]`j#[cr]" where [value] is a number.
<b>Description</b>	This command requests the maximum current setting of the drive.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check the maximum current of the drive.			
<b>Command</b>	"j" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Motor Parameters</i>	-	Realtime	None	"-[resistance],[inductance],[pole count],[voltage constant][cr]`-#[cr]" where [value] is a number.
<b>Description</b>	This command requests the motor characteristic settings currently implemented on the drive.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check the motors' resistance, inductance, magnetic pole count, and voltage constant settings currently implemented by the drive. Resistance, inductance, and voltage constant values will be multiplied by 1000.			
<b>Command</b>	"- " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Move Profile</i>	_	Realtime	None	"_[value][cr]`_#[cr]" where [value] is a number.
<b>Description</b>	This command requests the move profile setting of the drive.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check the move profile setting, trapezoidal or s-curve, of the drive.			
<b>Command</b>	"_ " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Moving</i>	o	Realtime	None	"oYES[cr]`o#[cr]" or "oNO[cr]`o#[cr]"
<b>Description</b>	This command requests whether the drive is moving.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to check if the drive is moving.			
<b>Command</b>	"o" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Program Names</i>	N	Realtime	none	"N[program1 name],[start page],[end page][CR] N[program2 name],[start page],[end page][CR] N#[CR]" More programs would have more entries.
<b>Description</b>	This command requests that all program names and addresses be sent.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	You want to know what programs are residing on the drive.			
<b>Command</b>	"N" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Read Startup Program</i>	K	Realtime	none	"K[program name][CR] K#[CR]" If there is no startup program, [program name] will be an empty string.
<b>Description</b>	This command requests the name of the startup program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	Want to know what program is set to run on power up.			
<b>Command</b>	"K" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Recall Program</i>	@	Realtime	Password, Program Name	The commands that make up the program, unless the password was incorrect, in which case there is no response.
<b>Description</b>	This command requests the program be read back.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Password	The password for the drive			A string; must be exactly 10 characters.
Program Name	The name of the program to be read back.			A string; must be exactly 10 characters.
<b>Example</b>	Want to read back a program named "program 1" from the drive, with no password.			
<b>Command</b>	"@ ,program 1 " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Remove Password</i>	q	Realtime	Password	None
<b>Description</b>	This command removes a password.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Password	The current password			A string, exactly 10 characters long
<b>Example</b>	You want to remove the password "password " .			
<b>Command</b>	"qpassword " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Remove Program</i>	<b>D</b>	Realtime	Program name	None
<b>Description</b>	This command removes a program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Program Name	The name of the program to be deleted.			A string, exactly 10 characters long
<b>Example</b>	You want to remove a program named "program 1 " from the drive.			
<b>Command</b>	"Dprogram 1 " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Restore Factory Defaults</i>	<b>a</b>	Realtime	None	None
<b>Description</b>	This command removes the drive password and deletes all the programs on the drive.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
None				
<b>Example</b>	You want to remove the password on a drive, but forgot that password.			
<b>Command</b>	"a" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Return</i>	<b>X</b>	Program	none	None
<b>Description</b>	This command returns from a subroutine.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	You want to return from a subroutine to where the subroutine was called from.			
<b>Command</b>	"X" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Return To</i>	<b>V</b>	Program	Destination	None
<b>Description</b>	This command exits a subroutine, branches to a location, and clears all pending interrupts, the return stack and the loop counters.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Destination	The address to which the program should branch.			0 to 87036, multiples of four only.
<b>Example</b>	You want to exit a subroutine and continue execution somewhere other than where the subroutine was called from, in this case, address 32.			
<b>Command</b>	"V32" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Run Program</i>	Y	Realtime	Program name	None
<b>Description</b>	This command begins the execution of a program, first returning to step 0 and setting all outputs low.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Program Name	The name of the program to run.			A string, exactly 10 characters long
<b>Example</b>	You want to run a program named "program 1 ", starting from the default state.			
<b>Command</b>	"Yprogram 1 " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Control Loop Gain</i>	(	Realtime	Set Control Gain	None
<b>Description</b>	This command sets the gains of the control loop used to control the motor, affecting motor response.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Kps	Velocity loop proportional gain			0 to 10000
Kis	Velocity loop integral gain			0 to 10000
Kds	Velocity loop derivative gain			0 to 10000
Kpp	Positional loop proportional gain			0 to 10000
Ktff	Current loop feed forward gain			0 to 10000
<b>Example</b>	You want to set the default gain settings of the drive: Kps = 100, Kis = 100, Kds = 0, Kpp = 20, and Ktff = 0			
<b>Command</b>	"(100,100,0,20,0" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Feedback Configuration</i>	<	Realtime	Hall Cell Spacing, Commutation Direction, and Encoder Direction	None
<b>Description</b>	This command sets the hall cell spacing of the motor, sets the commutation direction, encoder direction, and allows users to invert the hall cell logic			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Hall Cell Spacing	The hall cell spacing of the motor.			213087 for 60° spacing 215642 for 120° spacing
Commutation Direction	The commutation sequence direction to drive the motor.			0 for A→B→C 1 for B→A→C
Encoder Direction	The encoder signal direction			0 for CW (Ch. A leading Ch. B) 1 for CCW (Ch. B leading Ch. A)
Complement Hall Cell Logic	Whether to invert state of hall cells			0 to keep standard hall cell logic 1 to invert the hall cell logic
<b>Example</b>	You want to set the drive for 120° hall cell spacing, with standard commutation, a standard encoder direction, and standard hall cell logic			
<b>Command</b>	<215642,0,0,0" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Motor Parameters</i>	+	Realtime	Motors' Resistance, Inductance, Pole Count, and voltage constant	None
<b>Description</b>	This command configures the drive per a specific motor			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
Resistance	The resistance, per phase, of the motor (ohms multiplied by 1000)		200 to 100000	
Inductance	The inductance, per phase, of the motor (millihenries multiplied by 1000)		100 to 100000	
Pole Count	The magnetic pole count of the motor.		4,6,8, or 10	
Voltage Constant	The voltage constant of the motor (Volts/rad/sec multiplied by 1000)		1 to 10000	
<b>Example</b>	You want to set a resistance of 2.2 ohms, and inductance of 2.6 mH, a magnetic pole count of 4, and a voltage constant of 0.037 volts/rad/sec.			
<b>Command</b>	"+2200,2600,4,37." followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Move Profile</i>	=	Realtime	Password	None
<b>Description</b>	This command sets the move profile type for the drive to implement while performing moves.			
<b>Arguments</b>	<b>Argument Description</b>		<b>Valid Values or Range</b>	
Profile Type	Either trapezoidal or s-curve profile.		0 for Trapezoidal 1 for S-curve	
<b>Example</b>	You want to set the move profile type to trapezoidal.			
<b>Command</b>	"=0" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>				
<i>Set Outputs</i>	○	Realtime/Program	Output Value	None				
<b>Description</b>	This command sets the state of the outputs.							
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>				
Output Value	1 byte indicating which outputs should be set and what they should be set to. The most significant nibble indicates which outputs are being set, and the least significant nibble controls what they are being set to.			0 to 255				
<b>Example</b>	You want to set output 3 high, output 2 low, and want to leave outputs 1 and 4 unchanged.							
Bit 8 = 128	Bit 7 = 64	Bit 6 = 32	Bit 5 = 16	Bit 4 = 8	Bit 3 = 4	Bit 2 = 2	Bit 1 = 1	Total
0	1	1	0	0	1	0	0	100
<b>Command</b>	"O100" followed by a carriage return.							

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Password</i>	p	Realtime	Password	None
<b>Description</b>	This command sets a password, if none exists.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Password	The desired password.			A string, exactly 10 characters long
<b>Example</b>	You want to set the password as "password ".			
<b>Command</b>	"ppassword " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Position As</i>	Z	Realtime/Program	New Position	None
<b>Description</b>	This command adjusts the position counter.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
New Position	The position, in counts, you would like the current position to become.			-18446744073709551616 to 18446744073709551615
<b>Example</b>	After homing, you want to set the current location to 0.			
<b>Command</b>	"Z0" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Set Startup Program</i>	U	Realtime	Program name	None
<b>Description</b>	This command sets a program as the startup program.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Program Name	The name of the program to start on power up or reset.			A string, exactly 10 characters long
<b>Example</b>	You want to set a program named "program 1 " as the startup program.			
<b>Command</b>	"Uprogram 1 " followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Software Reset</i>	R	Realtime/Program	none	None
<b>Description</b>	This command causes the drive to restart, acts the same as cycling power.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	You want to restart the drive.			
<b>Command</b>	"R" followed by a carriage return.			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Stop</i>	H	RealTime/Program	End Speed, Decel Rate, Run Current, Decel Current, Hold Current, Delay Time, 1	None
<b>Description</b>	This command stops the motor using an optional deceleration ramp.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
End Speed	Must be a value of zero			0
Decel Rate	Rate at which the speed should fall from the current speed to the 0.			0 > to 16777215
Run Current	The maximum peak current, in milliamps			0 > to 5656
Hold Current	The maximum peak current, in milliamps, for after the move has completed.			0 to 5656
Decel Current	The maximum peak current, in milliamps, for the deceleration portion of the move.			0 > to 6500
Delay Time	Time, in milliseconds, of the transition from the deceleration current to the hold current			50 to 300
N/A	Not Applicable			1
<b>Example</b>	You wish to stop the motor, decelerating at a rate of 100000 counts per second <sup>2</sup> , with a maximum run current of 1.6 Apeak, a maximum decel current of 2.0 Apeak, and changing to a maximum hold current of 1.0 Apeak after a delay of 50 milliseconds.			
<b>Command</b>	"H0,100000,1600,2000,1000,50,1" followed by a carriage return			

<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Wait For Move</i>	F	Program	none	None
<b>Description</b>	This command causes the program to delay execution of the next command until the motor has stopped moving.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
none				
<b>Example</b>	You have started a move command and do not want the next command to execute until the move has finished.			
<b>Command</b>	"F" followed by a carriage return.			



<b>Command</b>	<b>Symbol</b>	<b>Context</b>	<b>Arguments</b>	<b>Response</b>
<i>Wait Time</i>	W	Program	Time	None
<b>Description</b>	This command causes the program to delay execution of the next command for a specified time.			
<b>Arguments</b>	<b>Argument Description</b>			<b>Valid Values or Range</b>
Time	The amount of time, in milliseconds, that the command should be delayed.			0 to 65535
<b>Example</b>	You have started a move command and do not want the next command to execute for 1 second.			
<b>Command</b>	"W1000" followed by a carriage return.			

# Brushless IDEA™ Drive

Hardware Manual

PBL4850E



**Haydon**  
Motion Solutions



[www.haydonkerk.com](http://www.haydonkerk.com)

All Rights Reserved

03-2015

## Table of Contents

Revision History .....	3
Introduction.....	4
Specifications .....	5
Engineering Drawings .....	6
Connections.....	7
Basic Wiring Diagram .....	8
Accessories .....	8
Encoder Inputs .....	9
Encoder Wiring.....	10
Hall Cell Inputs.....	11
Hall Cell Wiring.....	11
Digital I/O Pin Descriptions.....	12
Open Collector Output Pin Description.....	12
Input Pin Description .....	12
Digital I/O Wiring.....	13
Digital Output Wiring Examples .....	14
Digital Input Wiring Examples.....	14

## Revision History

<b>Date</b>	<b>Description</b>
January 2015	Initial version

## Introduction

This manual is intended to provide basic hardware specifications for the Haydon Kerk Brushless IDEA drive. For detailed information on use and programming of the drive, please refer to the IDEA Drive User's Manual. For detailed information on the command structure of the drive for coding purposes, please refer to the IDEA drive Communication Manual. All manuals are available at [idea-drive.com](http://idea-drive.com).

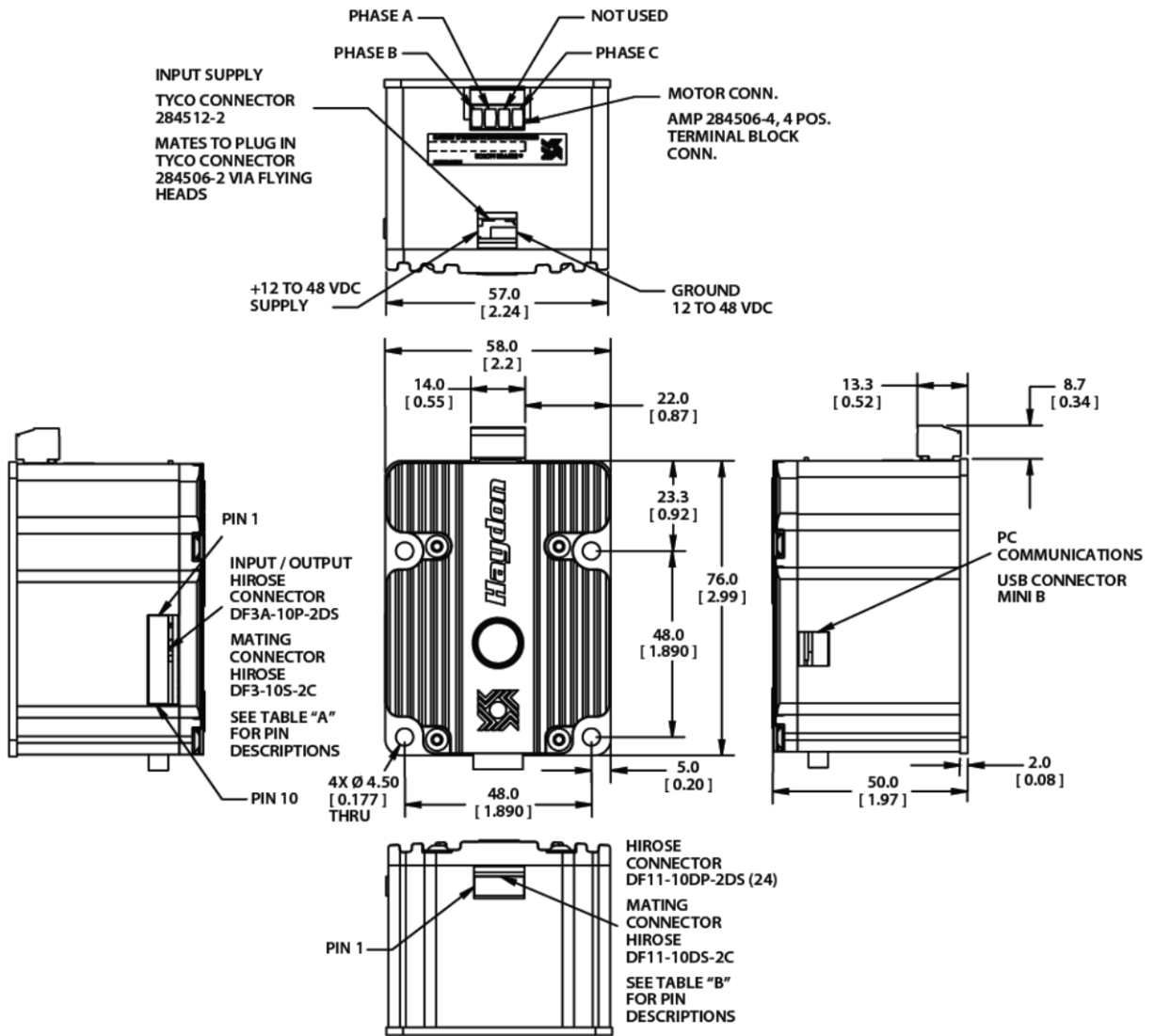
# PBL4850E IDEA™ DRIVE

## Specifications

<b>Attribute</b>	<b>Value</b>
Drive Input Voltage Range	12 – 48 VDC
Maximum Drive Current (per phase)	4.0 Arms (Up to 6.5 Apeak current boost capability during ramping)
Motor Type	3 Phase Brushless
Commutation Type	Sinusoidal with Hall Cell Initialization
Hall Cell Spacing	60° / 120°
Type of Ramping	Trapezoidal S-Curve
Communications	USB (Mini B connector)
Digital I/O Voltage Range	5-24Vdc
Digital Inputs	4
Digital Sinking Outputs	4
Digital Output Maximum Sinking Current	200mA (each)
Digital Input Maximum Current	8mA (each)
Maximum Temperature	70°C (Measured at heat sink)
Program Storage Size-Type	85 Kbytes-Flash
Maximum Number of Stored Programs	85, Referenced by 10 character program names
Position counter range	64bit
Interrupt sources	4 inputs (rising, falling or both edges), internal position counter (when reaching a programmed position).

# PBL4850E IDEA™ DRIVE Engineering Drawing

## Engineering Drawings



## Connections

**TABLE "A"**

PIN #	DESCRIPTION	NOTES
1	GROUND I/O SUPPLY	5 - 24 VDC
2	+ I/O SUPPLY	5 - 24 VDC
3	INPUT 1	
4	INPUT 2	
5	INPUT 3	
6	INPUT 4	
7	OUTPUT 1	
8	OUTPUT 2	
9	OUTPUT 3	
10	OUTPUT 4	

**TABLE "B"**

PIN #	DESCRIPTION
1	ENCODER CH A
3	ENCODER CH B
5	ENCODER INDEX
7	ENCODER 5VDC
9	ENCODER GND
2	HALL CELL A
4	HALL CELL B
6	HALL CELL C
8	HALL CELL 5 VDC
10	HALL CELL GND

**Basic Wiring:** To connect power to the drive and control it with the IDEA Drive

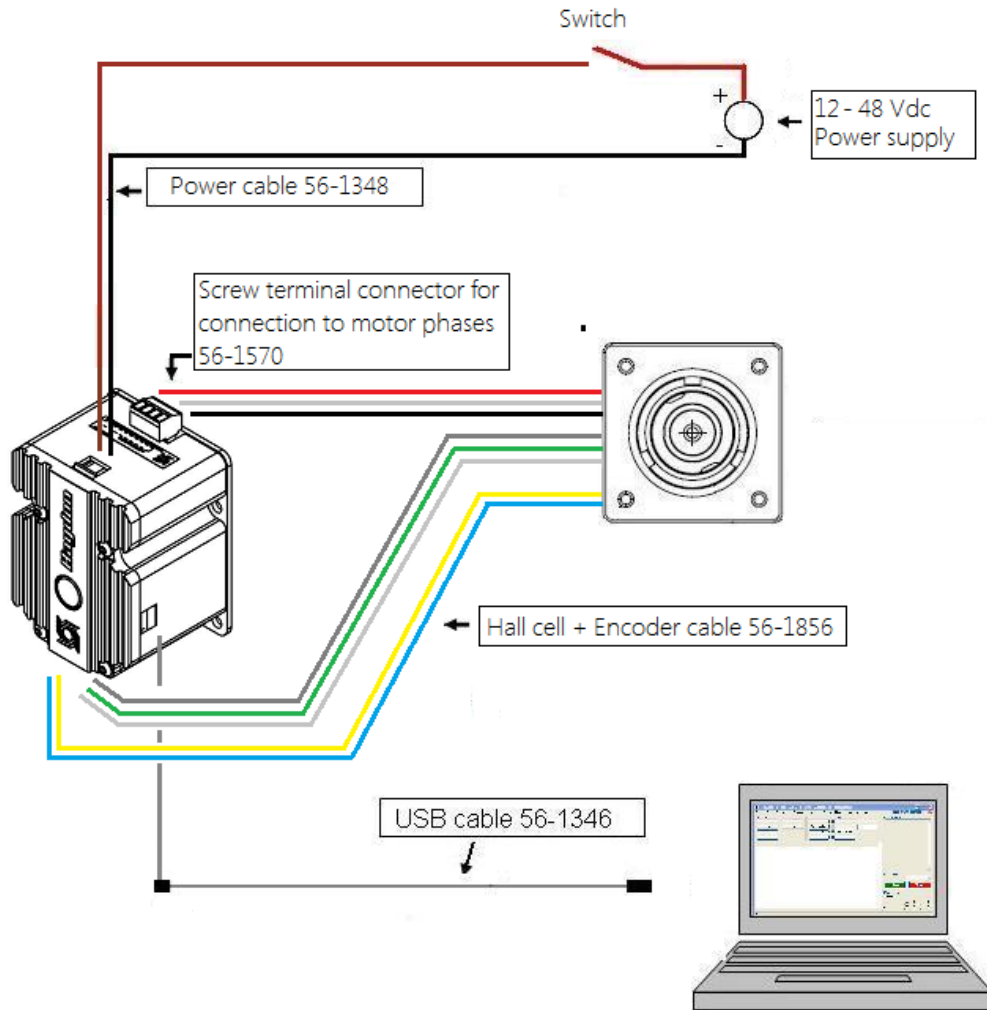
User Interface you will need the following:

- A power supply, minimum of 12VDC.
- A PC
- Power cable ( available from Haydon Kerk p/n 56-1348)
- Hall cell / Encoder harness (available from Haydon Kerk p/n 56-1856)
- Motor connected with screw terminal block ( available from Haydon Kerk p/n 56-1570
- 10 wire I/O cable (available from Haydon Kerk p/n 56-1352).
  - Note: this cable is only required if the drive is interacting with an external device.
- USB to Mini B USB cable ( available from Haydon Kerk p/n 56-1346 )

The following page contains the proper wiring diagram for the IDEA drive, power supply and PC. The I/O and encoder cables are omitted.



## Basic Wiring Diagram



## Accessories

Accessories	Part No.
USB Cable (A to mini B), 2 meters	56-1346
Power Cable, 1 meter	56-1348
I/O Cable, 1 meter	56-1352
Motor Connector Screw Terminal	56-1570
Hall Cell & Encoder Cable	56-1856

## Encoder Inputs

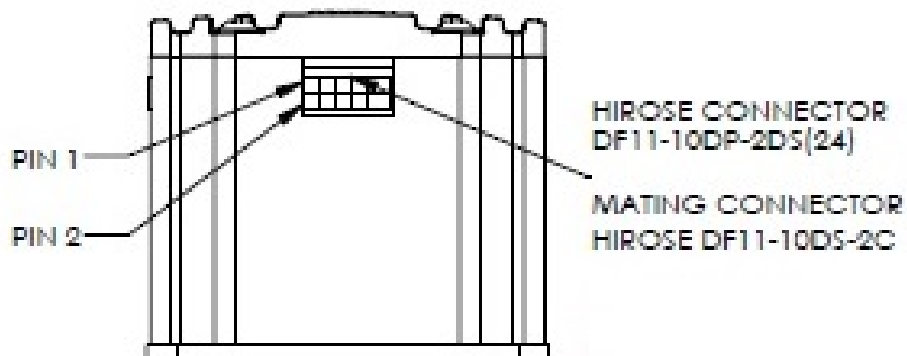
The IDEA drive is equipped with inputs for a single-ended, Quadrature encoder attached to the motor it drives. Quadrature encoders have 2 output signals, A and B, which are nominally 90 electrical degrees out of phase. On each rising or falling edge, the relative logic levels of the two phases can be used to determine the direction of rotation. The decoder within the drive interprets A leading B as motion in the clockwise direction, as viewed from the front face of the motor. This means that if a rising edge is detected on phase A, and phase B is at a logical high, then the motor just rotated counter-clockwise.

The IDEA drive watches for the rising and falling transitions on phase A and B, and increments or decrements the position counter accordingly. Using this method, a 1000 cycle per revolution optical rotary encoder would have 4000 counts per revolution.

## Encoder Wiring

The encoder connector can be wired to any 2 channel quadrature encoder that operates between 3.3Vdc and 5Vdc. For encoders that work on 5VDC, power to the encoder can be supplied through pin 4 of the encoder connector, otherwise a separate 3.3Vdc power supply is required. Whether or not power is being supplied by the drive, pin 5 must be connected to the same ground as the encoder. This is internally connected to the IDEA drive's ground connection. Pin 3 is for encoders with an index signal. This may be left unconnected, and is for future revisions which may make use of the index signal.

Pins 1 and 2 are the A and B connections, respectively. When the output shaft of the motor is rotating clockwise as viewed from the front of the motor phase A should lead phase B. Check your encoder's documentation to check if A and B need to be swapped, which can also be performed via software.

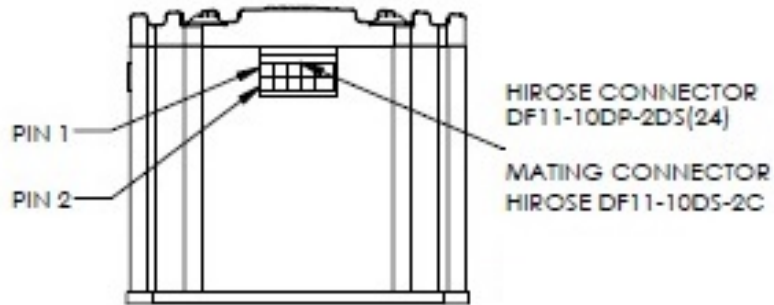


PIN #	DESCRIPTION
1	ENCODER CH A
3	ENCODER CH B
5	ENCODER INDEX
7	ENCODER 5VDC
9	ENCODER GND

## Hall Cell Inputs

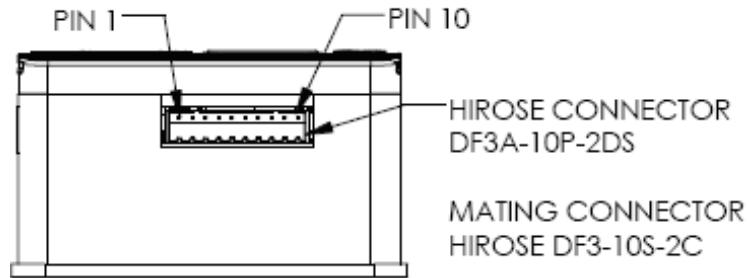
The brushless IDEA drive uses sinusoidal commutation and utilizes the hall cells for phase initialization. The IDEA drive accepts hall cell spacing configurations of 60° and 120°. Please follow the wiring diagram which came with the drive and motor combination for proper commutation and ideal motor performance.

## Hall Cell Wiring



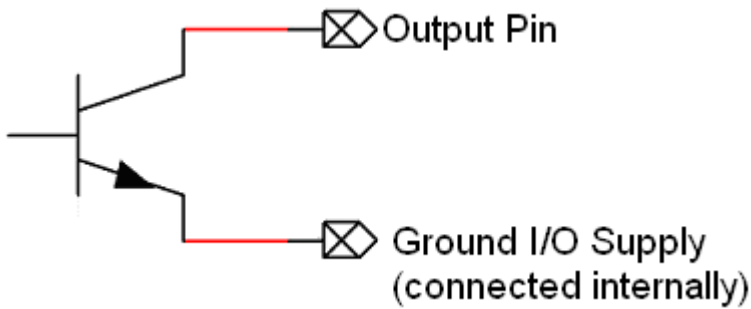
PIN #	DESCRIPTION
2	HALL CELL A
4	HALL CELL B
6	HALL CELL C
8	HALL CELL 5 VDC
10	HALL CELL GND

## Digital I/O Pin Descriptions

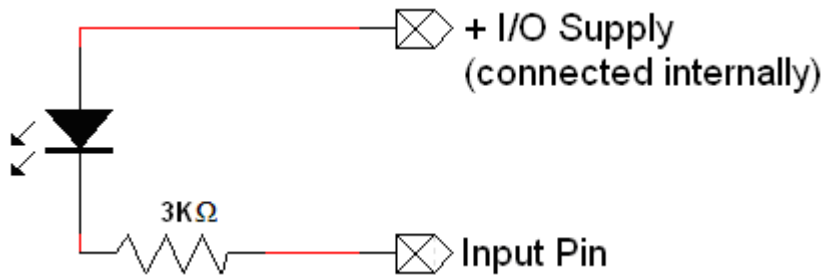


PIN POSITION	DESCRIPTION	NOTES
PIN 1	GROUND I/O SUPPLY	5 TO 24 VDC
PIN 2	+ I/O SUPPLY	5 TO 24 VDC
PIN 3	INPUT 1	
PIN 4	INPUT 2	
PIN 5	INPUT 3	
PIN 6	INPUT 4	
PIN 7	OUTPUT 1	
PIN 8	OUTPUT 2	
PIN 9	OUTPUT 3	
PIN 10	OUTPUT 4	

## Open Collector Output Pin Description



## Input Pin Description



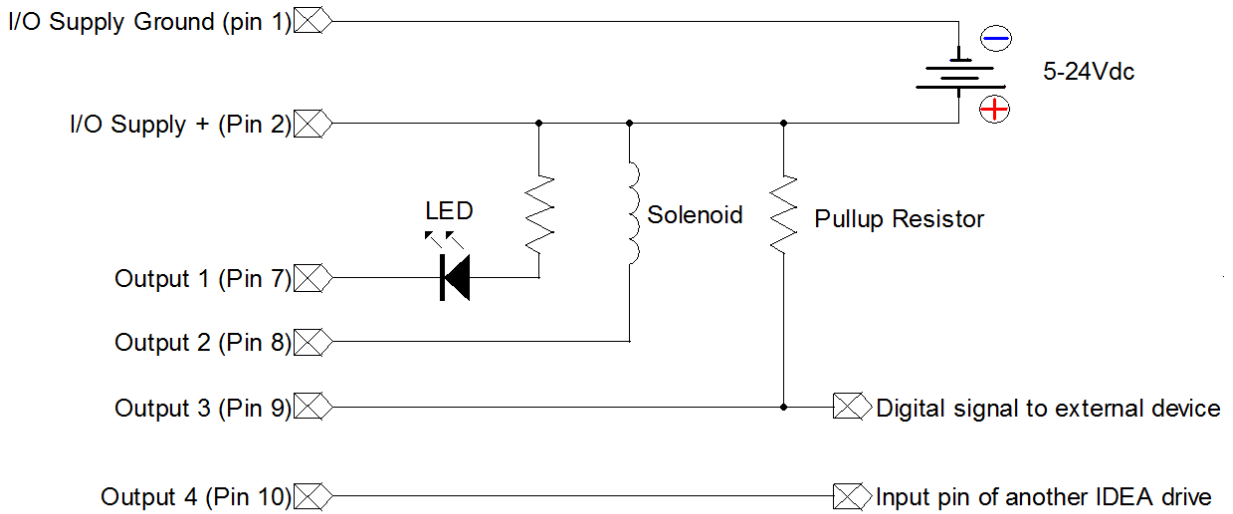
## Digital I/O Wiring

The IDEA drive has four optically isolated inputs and four optically isolated, open-collector outputs. A power supply is necessary to activate the opto-isolators with a voltage range of 5-24VDC. As the outputs are open-collector, they will need a pull-up resistor tied to the + I/O supply if a high level voltage is required. The outputs are capable of sinking up to 200mA each.

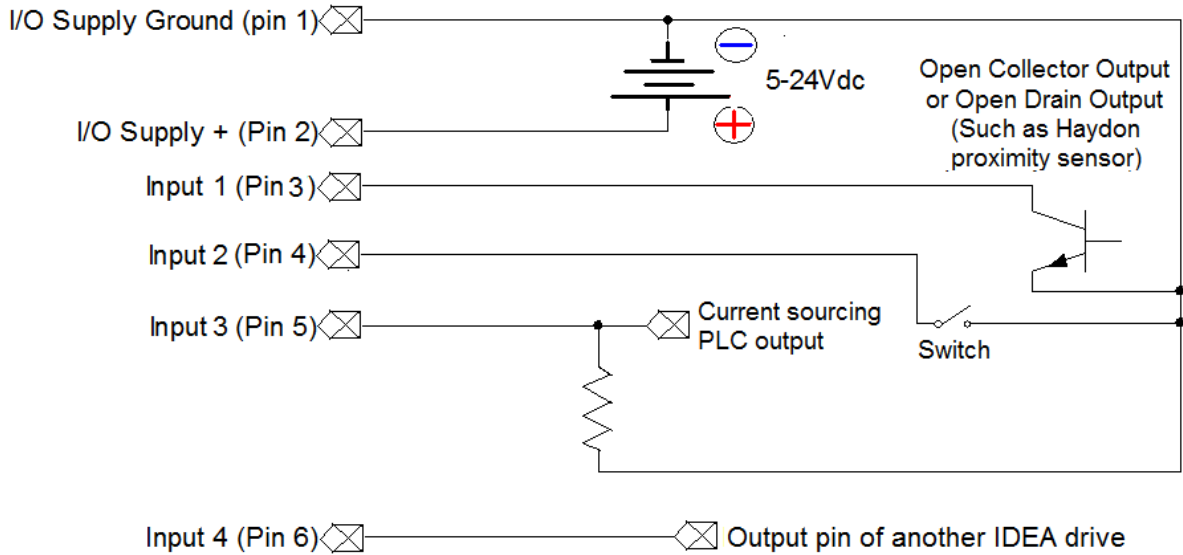
Note: The inputs can be used in two ways. They can be connected to logic levels that swing between I/O supply ground and + I/O supply, or they can be attached to a switch connected to I/O supply ground. In the second configuration, when the switch is open, the drive will see this as a logic high, when the switch is closed, and the input is connected to I/O supply ground, the drive will see this as a logic low.

Note: When an input is connected to a mechanical switch or relay, a phenomenon called “bounce” can occur. When the switch contact is almost closed, several electrical arcs can form. If an input is being used as an interrupt, each arc will be seen as a rising and falling edge, causing several false interrupts to trigger. Any input being used as an interrupt source should only be attached to solid state devices or a switch with de-bounce circuitry.

## Digital Output Wiring Examples



## Digital Input Wiring Examples



# Brushless IDEA™ Drive

Software User Manual

PBL4850E



**Haydon**  
Motion Solutions



[www.haydonkerk.com](http://www.haydonkerk.com)

All Rights Reserved

03-2015



## Table of Contents

Revision History .....	4
Introduction .....	5
<b>IDEA DRIVE and Software Basics .....</b>	<b>6</b>
Realtime Mode .....	6
Program Mode.....	7
<b>Startup.....</b>	<b>9</b>
Installing the Application.....	9
Getting Started.....	10
Drive Startup .....	13
<b>Features and Concepts .....</b>	<b>14</b>
Unit conversion .....	14
View Command String .....	15
Communications Modes .....	15
Maximum Speed.....	17
Ramping .....	18
Encoder.....	19
Saving Programs to the Drive .....	22
Removing Programs .....	22
Table of Contents.....	22
Startup Program.....	23
Saving/Loading/Combining Programs.....	23
Autosave.....	23
Over Current Protection.....	24
Accel/Decel Current Boost.....	24
Password Protection.....	24
Inputs and Outputs.....	25
Simulating Inputs.....	26
Debugger .....	26
Subroutines .....	27
Interrupts.....	28
Errors.....	30
<b>Explanation of Commands .....</b>	<b>32</b>
Extend .....	32
Retract.....	34
Move To .....	36
Go At Speed.....	38
Stop.....	40
E-Stop.....	41
Jump N Times .....	42
Goto .....	43
Goto If.....	44
Return.....	45
Return To.....	46
Goto Sub.....	47

Wait .....	48
Wait For Move .....	49
Int on Pos .....	50
Int on Input.....	51
Encoder.....	53
Set Outputs.....	54
Set Position .....	55
Reset.....	56
Abort.....	56
Comment.....	57
<b>Programming Examples.....</b>	<b>58</b>
<b>Example One .....</b>	<b>58</b>
<b>Example Two.....</b>	<b>62</b>
<b>Example Three .....</b>	<b>64</b>
<b>Example Four .....</b>	<b>68</b>
<b>Example Five.....</b>	<b>71</b>
<b>Example Six.....</b>	<b>76</b>
<b>The IDEA Drive Menu Items.....</b>	<b>80</b>
<b>File.....</b>	<b>80</b>
<b>Edit.....</b>	<b>80</b>
<b>Mode .....</b>	<b>80</b>
<b>Drive Commands.....</b>	<b>81</b>
<b>Communications Mode.....</b>	<b>81</b>
<b>Programs on Drive.....</b>	<b>81</b>
<b>Help.....</b>	<b>82</b>
<b>Glossary .....</b>	<b>83</b>

## Revision History

<b>Date</b>	<b>Description</b>
January 2010	Initial version
February 2010	Save debug output Goto if updated for outputs Interrupt and encoder configuration update
January 2011	Added detail to the behavior of interrupts Added detail to the IO wiring diagram Stand alone IO wiring diagram RS-485 wiring Communications features
May 2011	Added RS-485 Pin descriptions Added minimum time between resets
July 2011	Removed model specific information
August 2011	Added information about command strings
September 2011	Updated for 2.0 firmware and software changes Added introduction and part number information
December 2011	Clarified Goto if explanation Updated
March 2013	Clarified part number entry
January 2015	Revised manual for brushless IDEA drive

## **Introduction**

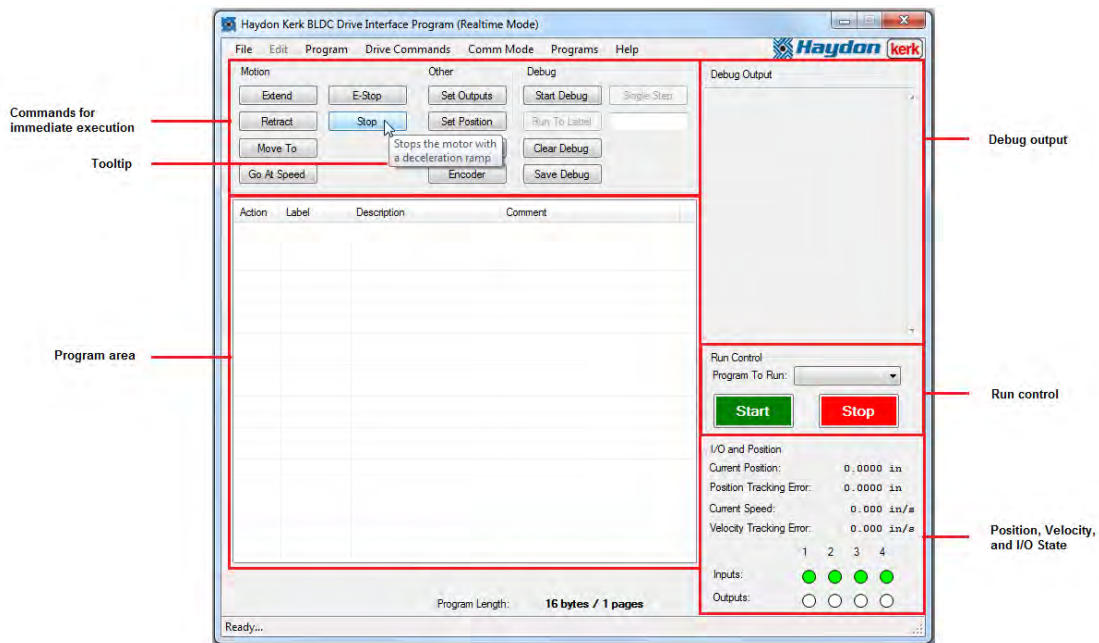
This manual is intended to provide information on using the Brushless IDEA drive from Haydon Kerk Motion Solutions. For information pertaining to a specific product, see the appropriate hardware manual, available at [haydonkerk.com](http://haydonkerk.com)

# IDEA Drive Software Basics

The Haydon Kerk Motion Solutions brushless IDEA drive and associated software are a complete package for the easy control of brushless motors within linear and rotary systems. This solution provides advanced features for both immediate execution as well as user written programs in an extremely user friendly way. All basic commands are used through intuitively named buttons, and each button and input field is further clarified through tooltips.

## Realtime Mode

Below is a screenshot of the User interface in the Realtime mode. This mode is only available when a drive is connected and communicating. A tooltip is demonstrated over the stop button.



In the Realtime mode, each command executed from the “Commands for immediate execution” section elicits an immediate action from the drive.

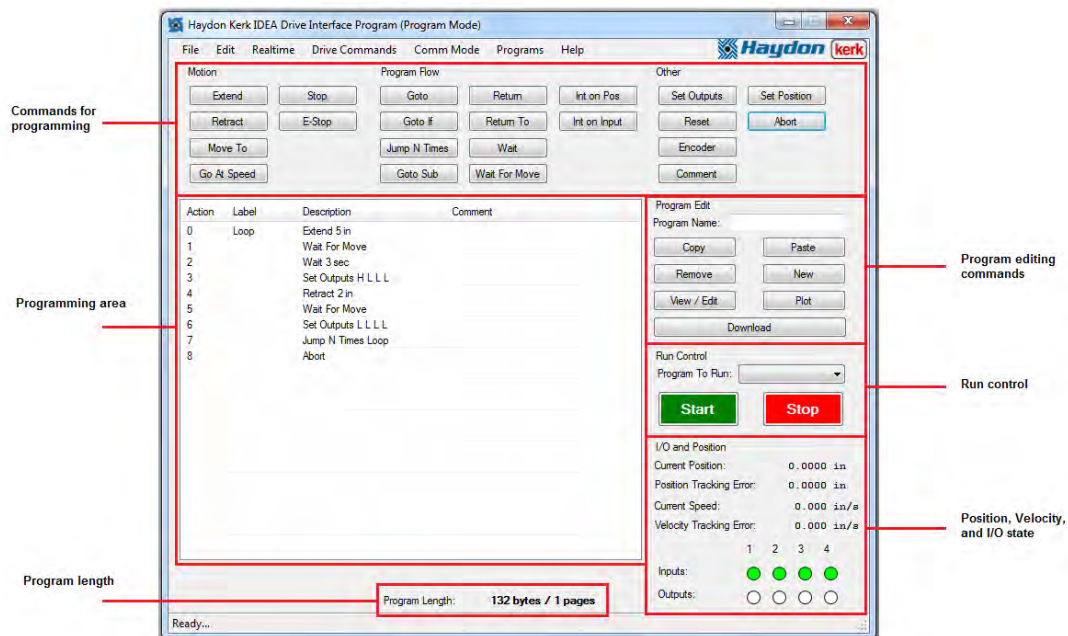
The program area can display any program currently on the drive. These programs can be run or aborted using the “Run Control” section.

The I/O and position section provides constant feedback from the drive pertaining to the current state of the drive. This area can also be used to control the state of the outputs, as well as the inputs if in simulation mode.

The IDEA software provides a powerful debugging feature, which allows the user to enter debug mode, which causes every command in the program to be displayed in the “Debug Output” section as is it executed. This debugger can be run in two ways, manually telling the drive to execute one command at a time (“Single-stepping”), or executing multiple commands in a row until reaching one of the user programmed labels (“Run to Label”). Through this feature the ease of debugging complex programs is greatly increased.

### Program Mode

Below is a screenshot of the user interface in Program Mode. This mode is available even without a drive attached.



The I/O and position section provides constant feedback from the drive pertaining to the current state of the drive. This area can also be used to control the state of the outputs, as well as the inputs if in simulation mode.

The “Run Control” area can be used to execute or stop any program on the drive.

The “Program Edit” area contains buttons used to edit the program as well as gather information about individual commands. The download button is also in this area.

The “Program Area” contains the command that make up the program currently being viewed or edited. Commands in the program are executed sequentially, starting at action 1 and moving onto action 2 and so on, unless a branching command is used, which would send execution instead to a specified label. The different commands available are covered in depth below.

The size of the current program is displayed in both bytes and pages in the “Program Length” area. Each page is 1024 bytes long, and the number of pages used is always rounded up.

## **Startup**

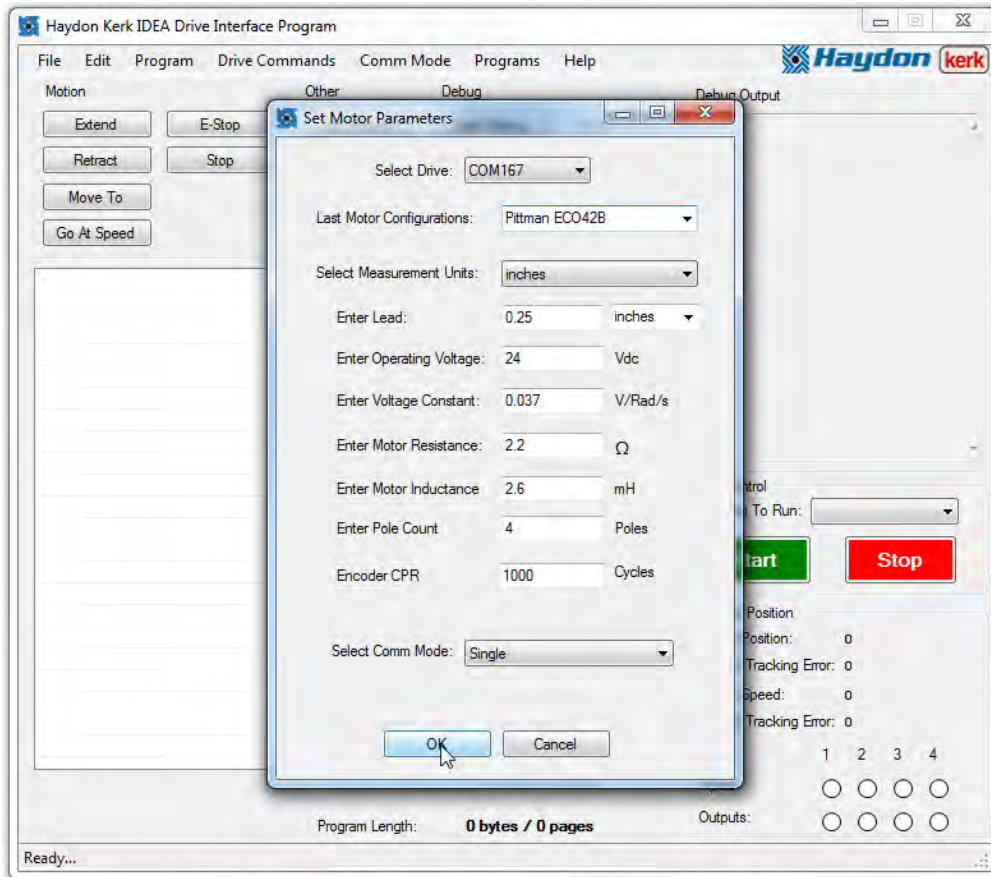
**Installing the Application:** (This is a one-time activity): The IDEA drive software can be downloaded from the Haydon Kerk website. Please perform the following instructions: ( Note, this is to be performed without the actuator being attached via the USB cable )

- Download the IDEA graphic user interface software from [www.HaydonKerk.com](http://www.HaydonKerk.com) under the IDEA drive stepper motor drive section. Double click the executable file to start the installation.
- “Welcome to the IDEA Software setup wizard” will appear on screen.
- “Select installation folder” will appear on screen. You may change the location of the installation if you wish. Click next.
- “Confirm installation” will appear on screen. Click next.
- “Installing IDEA Software” will appear on screen. This may take a few moments.
- “Installation complete” will appear on screen. A black function window will also appear behind the “Installation complete” dialogue box. This is normal. Do not close this window, it will close automatically. After a few moments the “Installation complete” dialogue box will show a “close” button. Click close.
- Installation is now complete and the IDEA Icon resides on your desktop and in your start menu under All Programs > Haydon Kerk.



**Getting Started:** Let us assume that the GUI software and drivers have been loaded into your PC and the drive is connected and powered up. Please proceed as follows:

- Start your PC.
- Double-Click the IDEA Icon on your desktop. This brings you to the initial screen. Screen shot is below:



- Select measurement units of inches or mm when working with a motor which is part of a linear system.
  1. Enter the lead of the screw of the linear system and select the native units of the screw
- Select measurement units of revolutions per second or revolutions per minute when working with a motor which is part of a rotary system.
  1. The lead input parameters will be eliminated from the screen.

- Enter the motor characteristics:
  1. Motor operating voltage
  2. Motor voltage constant in units of volts/radians/second
  3. Motor resistance per phase in units ohms
  4. Motor inductance per phase in units of millihenries
  5. Motor magnetic pole counts: 4,6,8, & 10 pole motors are supported
  6. Encoder cycles per revolution (CPR)
- Enter the name of the motor in the “Last motor configuration” input box
  1. After clicking OK many of these parameters will be placed into non-volatile memory for this particular motor.
  2. Additionally, the motor parameters will be saved to the windows user account. The software will save up to 5 user motor configurations and load the last configuration on next program startup.
- Select the Communication mode
  1. This will most likely be “Single”. The communications options are explained in the “Communications Modes” section of “Features and Concepts”.
- Click OK

Note: The com number will most likely be different from that shown above.

- The “Realtime” display now appears. This mode allows the program to execute immediate actions such as extend, retract, go at speed, etc.

If the drive is not connected, or not turned on, you will not be able to select a drive, and when the program is entered, you will be forced into the program mode. If this happens when the drive is connected and powered up, try disconnecting and reconnecting the usb cable.

### **Safety Margin Algorithm:**

The software uses the motors' characteristics to set safe operating limits during operation. Current and speed limitations are implemented in order to operate the motor at its continuous operation range. These limitations can be viewed using the "Motor Characteristics" function in the file menu.

Motor manufacturers may vary in the methods of testing for and providing motor specifications, therefore the software algorithm may under drive motor for some manufacturers. Additionally, some user may want to overdrive motors in low duty cycle application or when additional motor cooling methods are implemented.

Rather than completely limiting our users, the software instead provides a warning when the limits are exceeded. IT IS UP TO THE USER TO VERIFY CURRENT LIMITATION AGAINST THE MOTOR SPECIFICATIONS FROM THE MANUFACTURE AND ENSURE THE MOTOR IS BEING USED UNDER SAFE OPERATING CONDITIONS.

### **Initial Motor Setup:**

A motor configuration is necessary when connecting a new motor to the drive. The motor characteristics are saved onto the drives' non-volatile memory and subsequently reloaded the next time the drive is powered up. The drive uses the motor characteristics to properly commutate the motor. Aside from the motor parameter entry, there are additional functions which need to be set in order to accomplish this. If these settings are invalid then motor performance will be compromised and in most instances the motor will not run. These functions are listed below and explained in further detail under their self-titled headings under the "Features and Concepts" section.

- Motor Feedback Configuration
- Control Loop Gain (with optional Advanced Gains setting)
- Profile Select

## **Drive Startup**

When the drive first starts up, either by turning the power on, or using the “Reset” command on a drive that was already powered, the following occurs:

1. Input simulation is turned off.
2. All outputs are set low.
3. The position counter is set to zero.
4. The hold current is set to zero.
5. All interrupts are disabled
6. Motor parameters previously saved to drive are loaded.
7. If a startup program is selected, that program is now begun.

When a program is started, the following occurs:

1. All outputs are set low.
2. The position counter is set to zero.
3. All interrupts are disabled

## **Features and Concepts**

### **Auto filling of Inputs**

The software continually saves user inputs and automatically populates fields whenever possible. This is used to aid the user and speed up the programming process. Initially, on the motor parameter entry screen, the software will save the motor configuration to the windows user settings. The last 5 entered configurations will be saved and the latest configuration will be loaded automatically the next time the software is opened. Additionally, command windows are initially prefilled with recommended values for safe operating conditions of the motor. After a user updates a particular command, their parameters are saved and subsequently preloaded the next time the command is used.

### **Tooltips**

Tooltips provide the user with additional information about the controls within the software. A tooltip window will be displayed temporarily when the mouse is hovered over controls but not clicked. Tooltips will provide parameter limits for commands when the mouse is hovered over input boxes and functional insight for commands when hovered over buttons.

### **Unit conversion**

When starting the program, motor parameters and measurement units must be selected. From this information, the number of encoder counts for each command is calculated. This allows the user to write programs in their preferred units rather than having to calculate the number of counts to perform a move. As with any conversion, some rounding error may be present. A lead value is necessary when working with linear units of inches or mm.

## View Command String

The IDEA drive can be used without the GUI, by sending serial commands to it directly from your own programs or devices. In order to help in the development of these applications, a feature has been created that will show you the string that will be sent for a given command.

To access this feature, select the “File” menu, then the “Preferences” item, then “Enable Command Strings”. Once this is enabled, when you are editing parameters for commands, the string that would be sent to the drive will appear near the bottom of the window.

For some commands, such as “Goto” and “Goto Sub”, there will be values that cannot be displayed, because the location of the program on the drive is not yet known. These values will appear as [Address] and will need to be calculated based upon where in the final program the destination will reside, as well as the start page of the program.

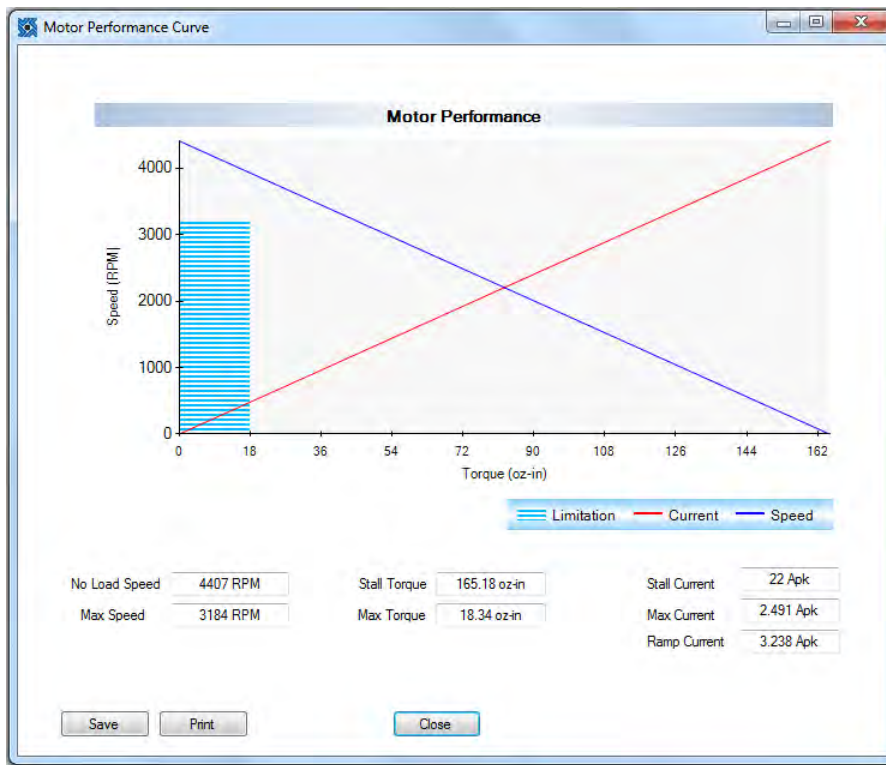
## Communications Modes

There are two different modes that the user interface can be in for communication between it and any attached drives, offline or single. You can change from one of these modes to any other through the “Comm Mode” menu item in either the “Realtime” or “Program” screens.

- Offline: In this mode, no communication is present, so the “Realtime” screen is unavailable, as well as the drive commands and ability to program.
- Single: This is the simplest communication mode. In this mode, only 1 drive should be attached per USB port. All functions are available.

## Motor Characteristics

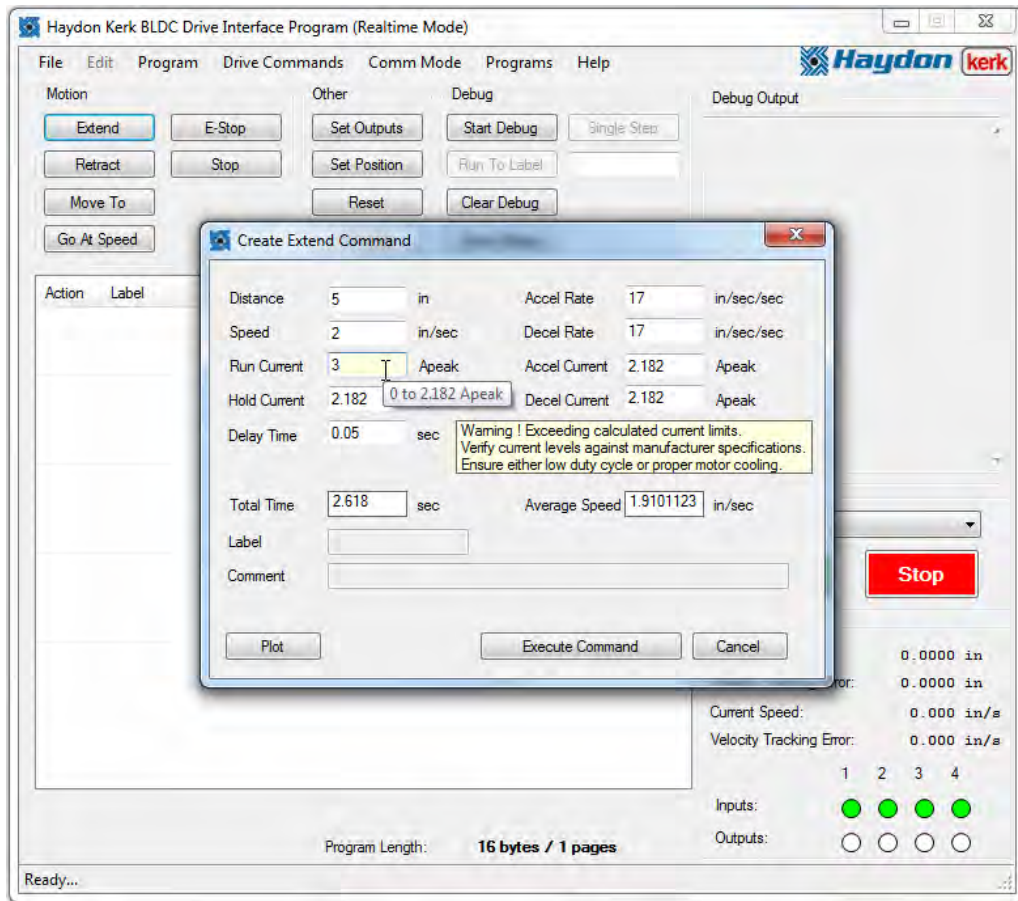
A motor performance curve can be viewed by selecting the “Motor Characteristics” function under the file menu. The graph is generated by the software using the motor specifications entered by the user at startup. A motor performance curve depicts the limitations of the motor by plotting the generated motor torque versus the speed and current ranges of the motor.



In addition, this graph will also display the motor limitations set by the software. Software limitations are designed with the intention to allow the user to drive the motor at its peak efficiency. Generally, a brushless motors' peak efficiency point falls between 15- 30% of the rated torque and slopes down beyond that. This is also within the area where a motor can operate continuously and is ideal for maximum motor life.

Users are allowed to exceed the limitations set by the software although it is recommended for users who are experienced with using brushless motors. When exceeding limitations, adequate cooling of the motor or a low duty cycle is

necessary to prevent damage to the motor. Users will be notified when exceeding limits by a warning message.



## Maximum Speed

The maximum speed of the drive is limited to the pole count of the motor.

4-pole: 10,000 RPM

6-pole: 6,600 RPM

8-pole: 5,000 RPM

10-pole: 4,000 RPM

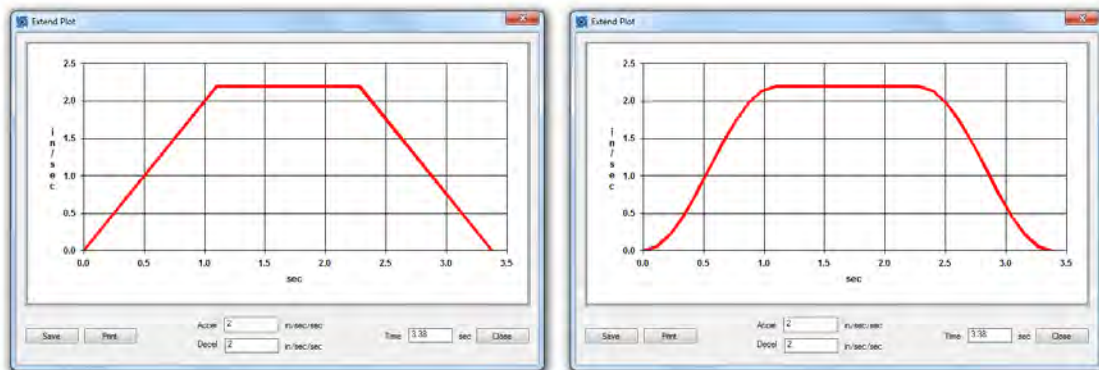
Exceeding these RPM values will result in rapid degradation of motor performance due to poor sinusoidal waveform fidelity.



## Motion Profiles

The Idea drive is capable of performing two types of motion profiles; trapezoidal or s-curve. The move profile can be toggled within the file menu under “Drive Settings”. Once selected, the profile is saved onto the drives’ non-volatile memory and used to perform all moves.

A trapezoidal move profile is most common and will work for majority of applications. Users may switch to an s-curve profile in an effort to smooth out movement by eliminating jerk conditions. A jerk condition occurs during a rate of change; this will occur at the beginning and end of a move, as well as during transition points. These points occur at the transition at the end of the acceleration point to maximum velocity as well as the transition point from maximum velocity to the deceleration portion of the move. S-curve ramping may take slightly longer than the equivalent trapezoidal ramp due to the addition settling time.



## Ramping

The IDEA Drive provides easy to use acceleration and deceleration ramps. A motion profile will be created by entering a speed along with the acceleration and deceleration rates. The versatility of the IDEA Drive allows users to create motion profiles in any of the units chosen; inches, millimeters, or revolutions. This simplifies generating motion profiles for linear and rotary applications alike. The

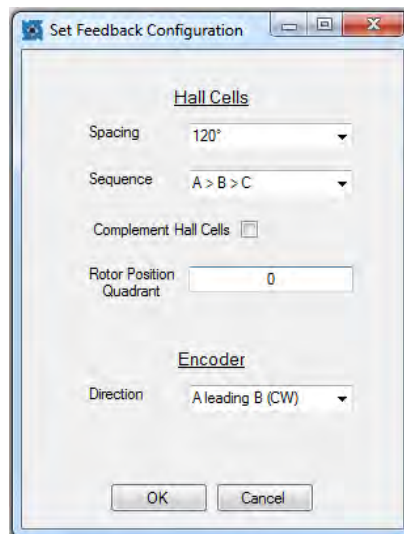
drive automatically calculates a move profile to approximate the desired parameters. For a picture of what the calculated ramp will look like, use the plot function on any “Extend”, “Retract”, “Move To”, or “Go At Speed” command.

## Encoder

The IDEA drive performs sinusoidal commutation for smooth motor performance and relies on the encoder for position feedback. The drive performs X4 decoding and will therefore quadruple the resolution of the encoder’s cycle per revolution value. Encoders with a higher CPR will allow finer resolutions and tighter control loops.

## Feedback Configuration

The feedback configuration allows the user to toggle several features that will modify how the drive reacts to the hardware attached to the motor.



The drive is capable of driving a motor with 60° or 120° hall cell spacing. Although the drive uses sinusoidal commutation to drive the motor, the hall cells are used for phase initialization; to start motor movement and determine direction.

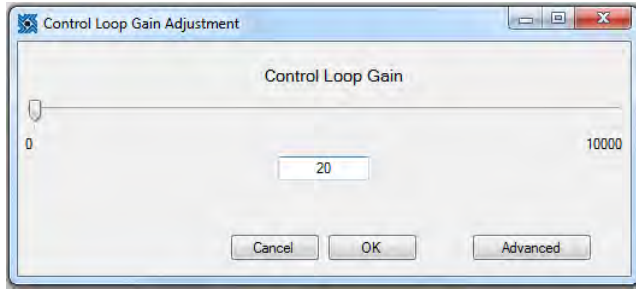
The user has the option to alter the commutation sequence if there was an error in wiring the motor to the drive. The standard, with a proper wiring connection, performs the following sequence  $A \rightarrow B \rightarrow C$  to commutate the motor. The user can swap phase A and B through the software by setting the sequence to  $B \rightarrow C \rightarrow A$ , and essentially reverse the commutation direction. Additionally, the complement function can be used to invert the logic of the hall cells to account for variances in motor construction.

The rotor position quadrant feature is an output which senses the 60 degree commutation quadrants during a motor rotation. This feature can be used to determine if the hall cells are correctly wired. Values of 0 to 5 depict a valid hall code, value of 6 depicts a braking situation where the phases are shorted, and a value of 7 is invalid.

The user may also swap the channels of the encoder if there is an error in wiring as well. A clockwise move is considered when the position counter increments as the shaft of the motor is turned clockwise when looking down on the front face of the motor: Channel A leads channel B and the encoder counts increase. Selecting “B leading A CCW” will reverse this operation without the need to rewire the encoder to the drive.

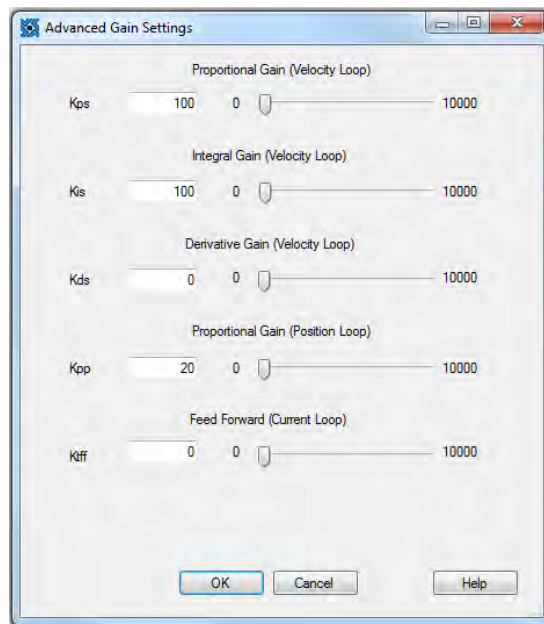
## **Control Loop Gain**

The control loop gain can be adjusted under “Drive Commands” then “Adjust Control Loop Gain”. Once set, this value will be saved to the non-volatile memory of the drive and will be utilized for subsequent moves. This gain will allow the user to adjust how closely the position of the motor is monitored. Increasing the gain will improve dynamic response time but may decrease stability. Effects of a setting which is higher may produce an increased overshoot, oscillations, and longer settling time to complete the move. Decreasing the gain will increase stability but may result in a more sluggish response. Decreasing this value will eliminate overshoot, oscillations, or minimize settling time.



## Advanced Gains Setting

For more advanced users which are familiar with brushless motor controls, the advanced gain settings window will allow for finer control of the drive.



This window will allow users to adjust the PID control loop of the brushless IDEA drive. Users can adjust the proportional, integral, and derivative gains of the velocity loop, as well as the proportional gain of the position loop and the feed forward of the current loop. Users should only adjust these gains if they are familiar with their functionality. Future software revisions will expand on the Help function within this form to aid users in fine adjustments of the control loop.

## **Saving Programs to the Drive**

After a program has been written, saving the program to the drive is simple. The program needs to be named, using the Program Name textbox. The program name may be up to 10 characters long, and cannot contain the “;” character. Any spaces at the end of the program name will not be saved, so “Program1” is the same as “Program1 ”. No two programs may have the same name, so attempting to download a program with the same name as an existing program will result in a warning, and continuing will cause the existing program to be overwritten.

When the download button is pressed, the user interface automatically finds the first area on the drive that can fit the program. If there is no place on the drive where the program can fit, the user is responsible for either moving or removing some programs. In order to move a program to free up larger blocks of free space, simply save the program to the drive again under the same name, and the program will be moved to the first page where it can be fit.

## **Removing Programs**

Programs can be removed from the IDEA drive in one of two ways. Either use the “Display Table of Contents” option under “Drive Commands”, then select the program to be removed in the list, then press “Remove” in the bottom left corner; or use the “Delete Program From Drive” option under “Drive Commands”, then select the program to be removed in the drop down menu and press “Ok”.

## **Table of Contents**

The IDEA Drive user interface provides a list of all programs on the drive, the pages being used by said programs, and a graphical representation of the contents and free space of the drive. This feature is accessed through the “Display Table of Contents” option under “Drive Commands”. This feature can be helpful in freeing up space for additional programs, or seeing which programs could be moved to decrease fragmenting free space.

## **Startup Program**

In real world applications, the controller will need to start execution once power is applied, as opposed to being started through the user interface. In order to set a program to start on power up, the “set Startup Program” option under “Drive Commands” is used. In the associated window, the current startup program is displayed in bold, and a new startup program can be selected using the drop down menu of all programs currently on the drive. If it is desired to not have a startup program, “No Startup Program” should be selected.

## **Saving/Loading/Combining Programs**

The IDEA user interface allows for programs to be saved to your computer or other drives. To do this, once the program is complete, go to “Save” in the “File” menu. This will open a save file dialog box. It is recommended that programs be backed up using the save function, to protect your work in case the program on the drive is overwritten.

To open a saved file, ensure you are in program mode and go to “Open” in the “File” menu. This will open an open file dialog box.

The IDEA user interface also provides the ability to combine two or more different programs. With a program open, go to “Add File” in the “File” menu. This will open an open file dialog. The file you select will be added to the end of the current program. This can be done multiple times to combine multiple programs.

## **Autosave**

The IDEA user interface provides an autosave feature to protect against losing all of your work. Once every minute, if the program is non-empty, the user interface automatically saves the file. If the program crashes, or is accidentally erased, the program can be recovered through the “Recover Autosave” option under the “File” menu. This will restore the most recently autosaved program.

Note: If you lose a program and wish to recover it through this feature, be sure not to open any other programs first. If this is done, the autosave feature may overwrite the file you wish to recover.

## **Over Current Protection**

The IDEA drive and user interface provide protection against overdriving the current in the motor. When the user interface is opened, the parameters entered for the motor provides the necessary information to calculate the maximum current of the motor. The user interface uses this to prevent the user from entering values that would be damaging to the motor.

## **Accel/Decel Current Boost**

The IDEA Drive provides the ability to boost the current for the acceleration ramp and deceleration ramp independently. This may be necessary when implementing sharp ramping rates.

Note: It is important for the user to ensure that the current boost feature is not over used. Repeated long ramp without rest can damage the motor if the boosted current is above the rated current of the motor.

## **Password Protection**

The IDEA drive has a password protection feature. When enabled, this feature prevents any program from being read back without the correct password. Passwords can be up to 10 characters in length and cannot contain the “;” character. Spaces at the end of the password are ignored, so “Password1” is the same as “Password1 ”.

Password protection does not prevent programs from being erased from or written to the drive. This feature is only meant to ensure that programs on password protected drives cannot be copied by a third party.

Note: If a drive has been password protected and the password has been lost, there are two options. The first is to send the drive back to Haydon Kerk Motion Solutions, where the password can be recovered for a fee. The second option is to use the “Restore Factory Defaults” option under the “Drive Commands” menu. This will remove the password protection, but all programs on the drive will also be lost.

## **Inputs and Outputs**

The IDEA drive has four optically isolated inputs and four optically isolated open-collector outputs. The voltage range for these is 5-24VDC. As the outputs are open-collector, they will need a pull-up resistor tied to the Opto-supply. The outputs are capable of sinking up to 200mA each.

Note: When an input is not connected to anything, it is seen as logic high. This allows connecting two IDEA drives without the use of pull-up resistors.

Note: The inputs can be used in two ways. They can be connected to logic levels that swing between opto ground and opto supply, or they can be attached to a switch connected to opto ground. In the second configuration, when the switch is open, the drive will see this as a logic high, when the switch is closed, and the input is connected to opto ground, the drive will see this as a logic low.

Note: When an input is connected to a mechanical switch or relay, a phenomenon called “bounce” can occur. When the switch contact is almost closed, several electrical arcs can form. If an input is being used as an interrupt, each arc will be seen as a rising and falling edge, causing several false interrupts to trigger. Any input being used as an interrupt source should only be attached to solid state devices or a switch with de-bounce circuitry.

In many cases, it is desirable to test a program that uses the inputs and outputs without actually connecting the hardware. In the bottom right hand corner of the user interface, there are eight circles, each representing one of the I/O. These



show the current state of all the inputs and outputs, with a green filled circle representing logic high, and an empty circle representing logic low. While a program is not running, or while a program is being used in the debug mode, these output circles can be clicked to toggle the state of the outputs.

### **Simulating Inputs**

Since the inputs are controlled externally, under normal circumstances, the user does not control them through the user interface. If it is desired to control the inputs through the user interface, the “simulate Inputs” feature can be accessed through the “Drive Commands” menu. When this feature is turned on, the actual states of the inputs are ignored and the user interface tells the drive what state to consider the inputs to be.

Note: The simulate inputs feature cannot be turned on or off while a program is running.

### **Debugger**

The IDEA user interface has a debugger to help in troubleshooting programs. The debugger is available in the Realtime mode. Once in the Realtime mode, the program to be debugged is selected either by the “Program to Run” drop down or through the “Programs on Drive” menu. Once the program is selected, the debug feature is started by pressing the “Start Debug” button. So long as the drive is in debug mode, each program line executed by the drive is displayed in the debug window. This window can be cleared at any time by using the “Clear Debug” button. The most recently executed command is also highlighted in the program area. Debug mode is turned off by either pressing the red “Stop” button, or when the program ends.

There are two ways of advancing through programs in debug mode, single step and running to a label. Each time the “Single Step” button is pressed, the drive executes one command. If the button is pressed multiple times while the drive is on a “Wait” or “Wait For Move” command, the drive will not execute multiple commands after the completion of the wait.

When the “Run To Label” button is pressed, the drive begins to execute the program normally, until the label in the textbox to the right of the “Run To Label” button is reached. If this label does not exist in the program, execution will continue until the red “Stop” button is pressed or execution ends on its own. Note: When the drive is executing many commands in a row without any “Wait” or “Wait For Move” commands, the user interface may be slowed due to the constant communication between the computer and the drive.

In order to aide in readability, as well as documentation, the debug output can be saved as a text file using the “Save Debug” button.

## **Subroutines**

A subroutine is a sequence of commands that can be used from anywhere in the program. When a subroutine is called, the address of the next command that was going to be executed is stored on what is called a stack. When the subroutine is exited using the “Return” command, execution of the program resumes at the address that was stored on the stack. A maximum of 10 addresses can be held in the stack. If there are 10 addresses on the stack, and another subroutine is called without a subroutine completing, a “Stack Overflow” error will be asserted and program execution will abort. It is up to the user to ensure that stack overflows do not occur.

Subroutines are exited using one of two commands, “Return” or “Return to”. When “Return” is used, program execution resumes at the address stored on the stack. When “Return To” is used, a destination address is specified. Execution now resumes at the label specified, and the stack is emptied. The use of “Return to” exits all subroutines at once.

If a “Return” or “Return To” command is used when the stack is empty, a “Stack Underflow” error will be asserted and execution of the program will halt. It is the responsibility of the user to ensure that stack underflows do not occur.

There are two ways in which a subroutine can be called; the simplest is the “Goto Sub” command. A “Goto Sub” command is used to call a subroutine from within the program, usually to complete a sequence of commands which is used repeatedly in the program. The address stored on the stack when a “Goto Sub” command is used is the address immediately after the “Goto Sub”.

The second way to call a subroutine is through interrupts. Interrupts can occur at any time, and are explained in the interrupts section, page 27. When a subroutine is called by an interrupt, the address stored in the stack is the address of the command which would have next been executed. If the interrupt is triggered during a “Wait” or “Wait For Move” command, the address of the “Wait” or “Wait For Move” command is stored, and if the subroutine is completed by the “Return” command, the wait is resumed until it’s completion. Note: If a “Wait” command is interrupted, the time spent executing the interrupt or interrupts counts toward the “Wait” command’s delay time.

## **Interrupts**

The IDEA drive has three different types of interrupts, input triggered, position triggered and encoder triggered. All three operate in the same fashion, the difference being only how they are triggered.

When an interrupt is triggered, it branches to a subroutine with a specific priority. All interrupts need two parameters, the label of the subroutine to be run when triggered and the priority. For more information on subroutines, see Subroutines, page 28.

The priority of the interrupt determines the order in which any pending interrupts are serviced. If an interrupt is triggered while a lower priority interrupt is being serviced, the program immediately begins servicing the new interrupt. If an interrupt is triggered while an interrupt of equal or higher priority is being serviced, the new interrupt is put into a queue of pending interrupts and will be serviced when all higher priority interrupts have been serviced and any interrupts of the same priority which were triggered first have been serviced.

There are 5 interrupt queues, one for each priority level, each being 10 interrupts long at most. If there are 10 interrupts in one queue, and another interrupt of that priority is triggered, the new interrupt will be ignored and an "Interrupt queue full" error will be asserted.

If an interrupt is reconfigured to trigger a new subroutine when there is an instance of that interrupt in a queue, the interrupt in the queue will still execute, but will execute the new subroutine, not the subroutine that the interrupt was originally configured to jump to. Care should be taken when programming ensure this does not occur.

Input Triggered: Each input can be configured to trigger an interrupt through the "Int on Input" button. The "INT" radio button should be selected for any input which is to be used as an interrupt source.

The trigger type should also be selected. The types are:

Rising edge: Occurs when the input level goes from low to high.

Falling edge: Occurs when the input level goes from high to low.

Both edges: Occurs anytime the input changes state.

Position triggered: Using the "Int On Position" command in a program, an interrupt can be set to trigger when the motor reaches a specific position. Note the only one interrupt on position can be initialized at once. If it is desired to set

an interrupt on multiple positions, the position of the interrupt needs to be continually reset.

## **Errors**

During operation, the user interface may report an error, below is an explanation of each of these errors.

**IO Error: An error occurred when attempting to update the IO:** This error occurs when communications between the drive and user interface is interrupted. This may happen on occasion when the drive is busy with a task and temporarily does not respond to the user interface. Press “Retry”. If the message continues to appear, check all connections.

**Stack Underflow:** This error occurs when a running program runs to a “Return” command while not within a subroutine. Check the running program for ways that the program could get to a “Return” without having used a “Goto Sub” or an interrupt.

**Stack Overflow:** This error occurs when 10 or more subroutines are called without returning. Check the running program to ensure that all subroutines end with either a “Return” or “Return To”, and that you are not nesting too many subroutines.

**Driver Overtemp:** This error occurs when the internal temperature of the drive exceeds a safe level. Consider moving the drive to a cooler location, reducing the hold and/or run currents, adding additional heat sinking, or adding active cooling.

**Encoder Error:** This error occurs when the encoder encounters a problem.

**Interrupt Queue Full:** This error occurs when a running program encounters more than 10 interrupts of the same priority without having serviced any. This

could occur due to an interrupt source causing multiple extraneous interrupts, or by one interrupt subroutine not returning, thus preventing execution of any other interrupts. Check your interrupt sources, and ensure that all interrupt subroutines end in a “Return” or “Return To” command.

**Loop Overflow:** This error occurs when a running program is in more than 10 “Jump N Times” commands at once. Ensure that the running program does not nest more than 10 “Jump N Times” commands.

**Drive Current Limit:** This error occurs when the drive is given a command with a current setting higher than its capability.

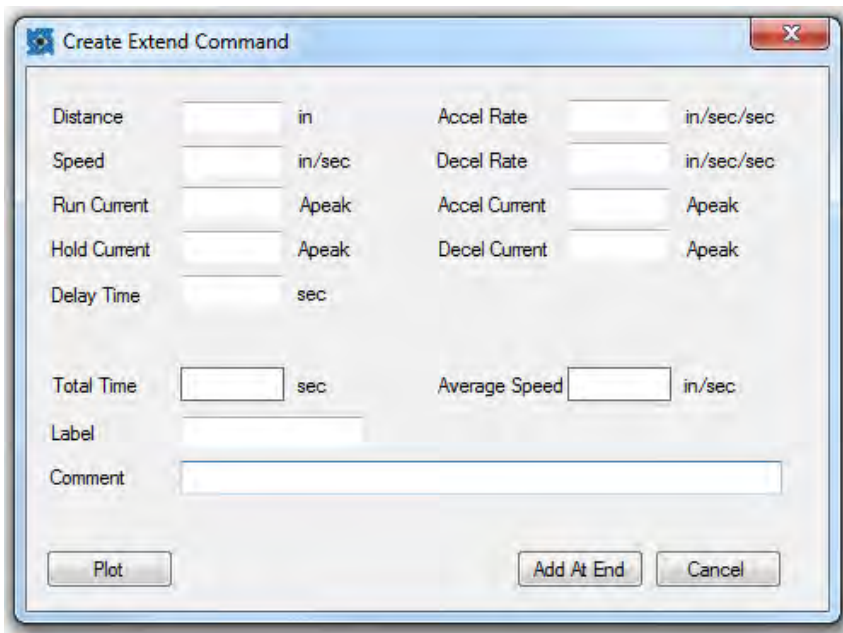
**Bad Checksum, update aborted:** This error is most likely caused by communications being interrupted during a firmware update. Hit “OK” and attempt to update the firmware again. If the problem persists, contact Haydon Kerk.

**Unknown Error:** If you ever receive this error, make a note of what you were doing at the time, and contact Haydon Kerk.

## Explanation of Commands

### **Extend/Index CW**

The extend command moves the motor a specified distance forwards from its current location. When using a rotary motor, this command rotates the motor clockwise, as viewed from the output shaft.



The screenshot shows a dialog box titled "Create Extend Command". It contains the following fields and labels:

Distance	<input type="text"/>	in	Accel Rate	<input type="text"/>	in/sec/sec
Speed	<input type="text"/>	in/sec	Decel Rate	<input type="text"/>	in/sec/sec
Run Current	<input type="text"/>	Apeak	Accel Current	<input type="text"/>	Apeak
Hold Current	<input type="text"/>	Apeak	Decel Current	<input type="text"/>	Apeak
Delay Time	<input type="text"/>	sec			
Total Time	<input type="text"/>	sec	Average Speed	<input type="text"/>	in/sec
Label	<input type="text"/>				
Comment	<input type="text"/>				

Buttons at the bottom: Plot, Add At End, Cancel.

### **Parameters**

Distance: This is the distance that the motor will extend or rotate to.

Speed: This is the top speed at which the motor will extend or rotate at.

Run Current: This is the maximum peak current per phase that may be applied to the windings while the motor moves at the top speed.

Hold Current: This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

Accel Rate: This is the rate at which the motor will be ramped up from standstill to the run speed.

Decel Rate: This is the rate at which the motor will be ramped down from the run speed to standstill.

Accel Current: During acceleration, the maximum peak current per phase that may be applied to the windings used to bring the motor up to the run speed from a standstill.

Decel Current: During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

Delay Time: The dwell time between switching from the decel current to the hold current.

The following program extends the motor in a linear system 1”, and waits for the move to complete before executing any commands which may follow:

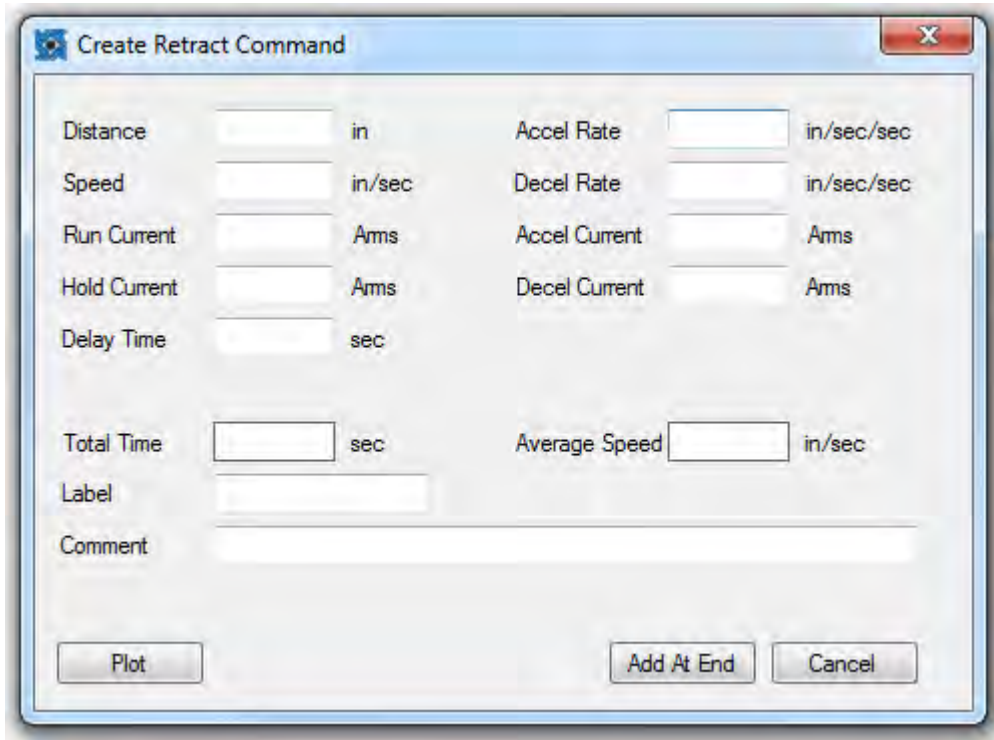
Action	Label	Description	Comment
0		Extend 1 in	
1		Wait For Move	

This command is used in example 1.



## Retract/Index CCW

The retract command moves the motor a specified distance reverse from its current location. When using a rotary motor, this command rotates the motor counterclockwise, as viewed from the output shaft.



The screenshot shows a software dialog box titled "Create Retract Command". It features a standard Windows-style title bar with a close button (X) in the top right corner. The main area of the dialog is organized into two columns of input fields. The left column includes: "Distance" (input field) with "in" to its right; "Speed" (input field) with "in/sec" to its right; "Run Current" (input field) with "Ams" to its right; "Hold Current" (input field) with "Ams" to its right; "Delay Time" (input field) with "sec" to its right; "Total Time" (input field) with "sec" to its right; "Label" (text input field); and "Comment" (text input field). The right column includes: "Accel Rate" (input field) with "in/sec/sec" to its right; "Decel Rate" (input field) with "in/sec/sec" to its right; "Accel Current" (input field) with "Ams" to its right; "Decel Current" (input field) with "Ams" to its right; and "Average Speed" (input field) with "in/sec" to its right. At the bottom of the dialog, there are three buttons: "Plot" on the left, "Add At End" in the center, and "Cancel" on the right.

### Parameters

Distance: This is the distance that the motor will retract or rotate to.

Speed: This is the top speed at which the motor will extend or rotate at.

Run Current: This is the maximum peak current per phase that may be applied to the windings while the motor moves at the top speed.

Hold Current: This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

Accel Rate: This is the rate at which the motor will be ramped up from standstill to the run speed.

Decel Rate: This is the rate at which the motor will be ramped down from the run speed to standstill.

Accel Current: During acceleration, the maximum peak current per phase that may be applied to the windings used to bring the motor up to the run speed from a standstill.

Decel Current: During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

Delay Time: The dwell time between switching from the decel current to the hold current.

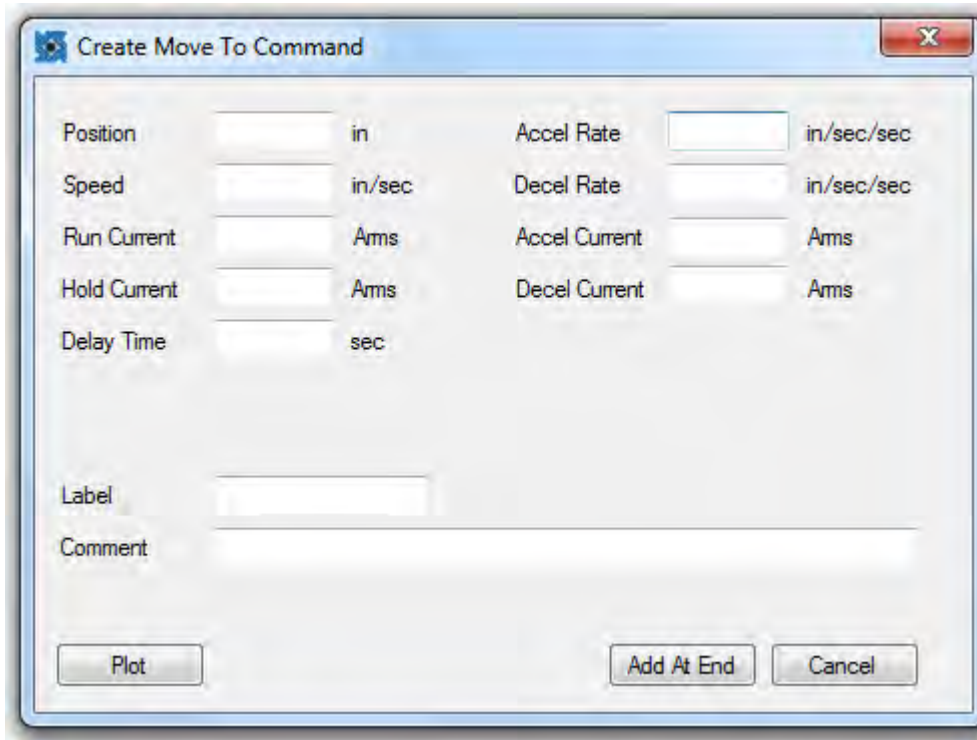
The following program retracts the motor in a linear system 1”, and waits for the move to complete before executing any commands which may follow:

Action	Label	Description	Comment
0		Retract 1 in	
1		Wait For Move	

This command is used in examples 1, 2, 3, 4, 5 and 6.

## Move To

The move to command moves the motor to a specific location, based upon the encoder.



The image shows a software dialog box titled "Create Move To Command". It features a standard Windows-style title bar with a close button (X) in the top right corner. The main area of the dialog is divided into two columns of input fields. The left column includes: "Position" (input field) with the unit "in", "Speed" (input field) with the unit "in/sec", "Run Current" (input field) with the unit "Ams", "Hold Current" (input field) with the unit "Ams", and "Delay Time" (input field) with the unit "sec". The right column includes: "Accel Rate" (input field) with the unit "in/sec/sec", "Decel Rate" (input field) with the unit "in/sec/sec", "Accel Current" (input field) with the unit "Ams", and "Decel Current" (input field) with the unit "Ams". Below these fields are two more input areas: "Label" (a single-line text box) and "Comment" (a multi-line text area). At the bottom of the dialog, there are three buttons: "Plot" on the left, "Add At End" in the center, and "Cancel" on the right.

### Parameters

Position: This is the position to which the motor will move to.

Speed: This is the top speed at which the motor will extend or rotate at.

Run Current: This is the maximum peak current per phase that may be applied to the windings while the motor moves at the top speed.

Hold Current: This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

Accel Rate: This is the rate at which the motor will be ramped up from standstill to the run speed.

Decel Rate: This is the rate at which the motor will be ramped down from the run speed to standstill.

Accel Current: During acceleration, the maximum peak current per phase that may be applied to the windings used to bring the motor up to the run speed from a standstill.

Decel Current: During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

Delay Time: The dwell time between switching from the decel current to the hold current.

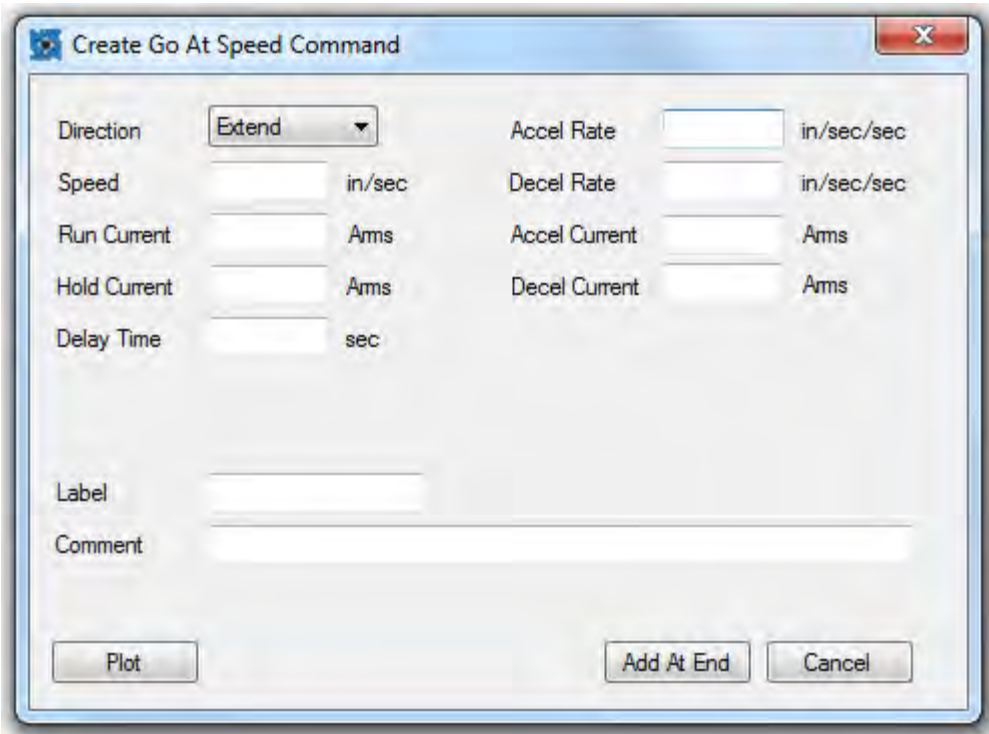
The following program moves the motor to a point 1" forward from where the motor was at the beginning of the program, and waits for the move to complete before executing any commands which may follow:

Action	Label	Description	Comment
0		Move To 1 in	
1		Wait For Move	

This command is used in examples 4, 5, and 6.

## Go At Speed

The Go At Speed command moves the motor in a specified direction at a specified speed.



### Parameters

Direction: This is the direction in which the motor will move.

Speed: This is the top speed at which the motor will extend or rotate at.

Run Current: This is the maximum peak current per phase that may be applied to the windings while the motor moves at the top speed.

Hold Current: This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

Accel Rate: This is the rate at which the motor will be ramped up from standstill to the run speed.

Decel Rate: This is the rate at which the motor will be ramped down from the run speed to standstill.

Accel Current: During acceleration, the maximum peak current per phase that may be applied to the windings used to bring the motor up to the run speed from a standstill.

Decel Current: During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

Delay Time: The dwell time between switching from the decel current to the hold current.

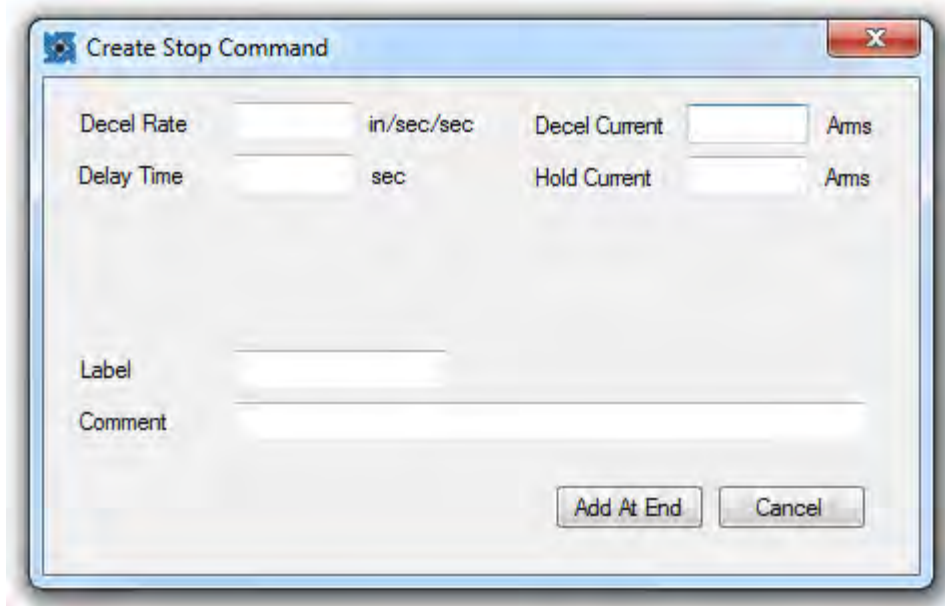
The following program moves the motor for 1 second.

Action	Label	Description	Comment
0		Go At Speed 1 in/sec	
1		Wait 1 sec	

This command is used in example 6.

## Stop

The Stop command brings the motor to a stop with an optional deceleration ramp. This does not halt program execution.



### Parameters

**Decel Rate:** This is the rate at which the motor will be ramped down from the run speed to standstill.

**Decel Current:** During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

**Hold Current:** This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

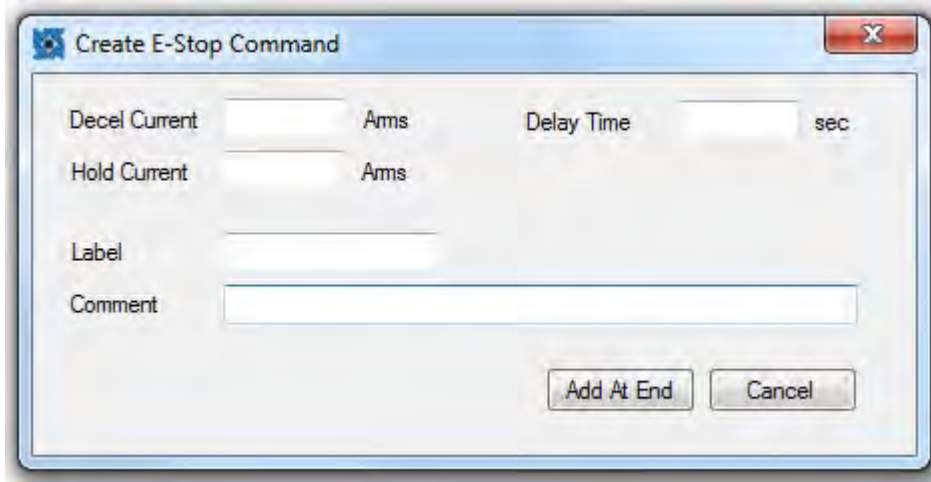
The following program starts a move at 1" per second, waits 0.5 seconds, and the stops the actuator.

Action	Label	Description	Comment
0		Go At Speed 1 in/sec	
1		Wait .5 sec	
2		Stop	

This command is used in example 5.

## E-Stop

The E-Stop command brings the motor to an immediate stop. This does not halt program execution.



### Parameters

**Decel Current:** During deceleration, the maximum peak current per phase that may be applied to the windings used to stop the motor from the run speed to a standstill.

**Hold Current:** This is the maximum peak current per phase that may be applied to the windings when the motor is at standstill.

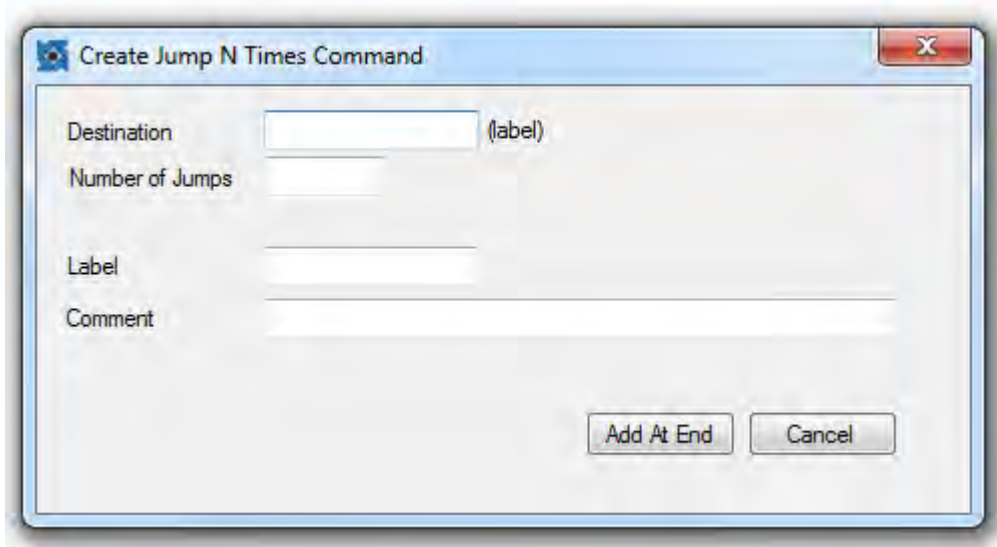
The following program starts a move at 1" per second, waits 0.5 seconds, and then stops the actuator.

Action	Label	Description	Comment
0		Go At Speed 1 in/sec	
1		Wait 0.5 sec	
2		E-Stop	



## Jump N Times

The Jump N Times command goes to a specified label a specified number of times. Once the number of jumps has been completed, execution continues at the next line in the program, if any exist.



### Parameters

Destination: This is the label of the command that should be jumped to.

Number of Jumps: This is how many times the command should jump.

The following program extends .25", waits for the move to stop, then repeats 3 times.

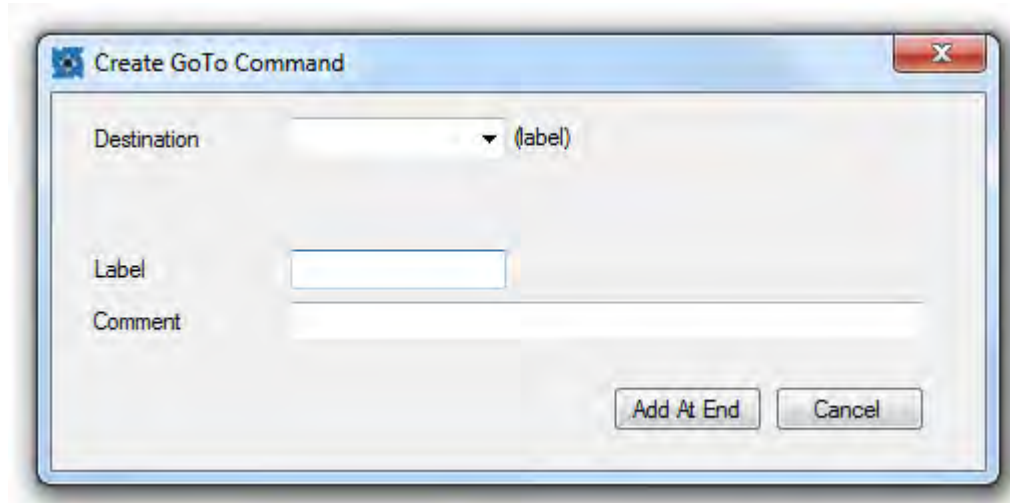
Note: Because the command goes to the extend command 3 times from the jump n times command, the extend is performed a total of 4 times.

Action	Label	Description	Comment
0	extend	Extend 0.25 in	
1		Wait For Move	
2		Jump N Times extend	Number of Jumps: 3

This command is used in example 2.

## Goto

The Goto command goes to a specified label.



### Parameters

Destination: This is the label of the command to which the program will go.

The following program extends .25", waits for the move to stop, then repeats until the program is aborted.

Action	Label	Description	Comment
0	extend	Extend 0.25 in	
1		Wait For Move	
2		Goto extend	

This command is used in examples 1, 5 and 6.

## Goto If

The Goto If command goes to a specified label if the current states of the inputs match the specified conditions, and goes to the next line in the program otherwise.



## Parameters

**Destination:** This is the label of the command that should be jumped to.

**Inputs/Outputs:** Each I/O can be specified as High, Low, or not tested. Any I/O set to "Not Tested" will be ignored; the state of the I/O when the command is executed must match all settings of high or low in order to go to the specified label, otherwise, the next line of the program is executed.

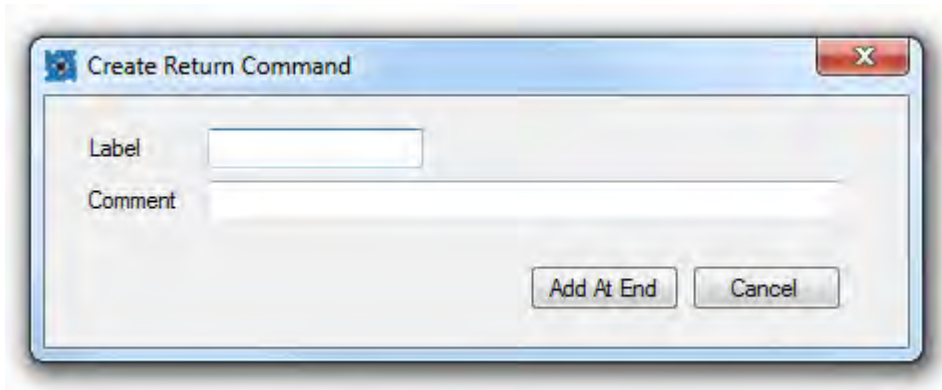
The following program extends .25", waits for the move to stop, then repeats so long as input 1 is high and input 2 is low.

Action	Label	Description	Comment
0	extend	Extend 0.25 in	
1		Wait For Move	
2		Goto If extend	1 set to high, 2 set to low

This command is used in example 5.

## Return

The return command ends a subroutine and returns execution to the location from which the subroutine was called. For further explanation of subroutines, see Subroutines, page 26.



## Parameters

This command has no parameters.

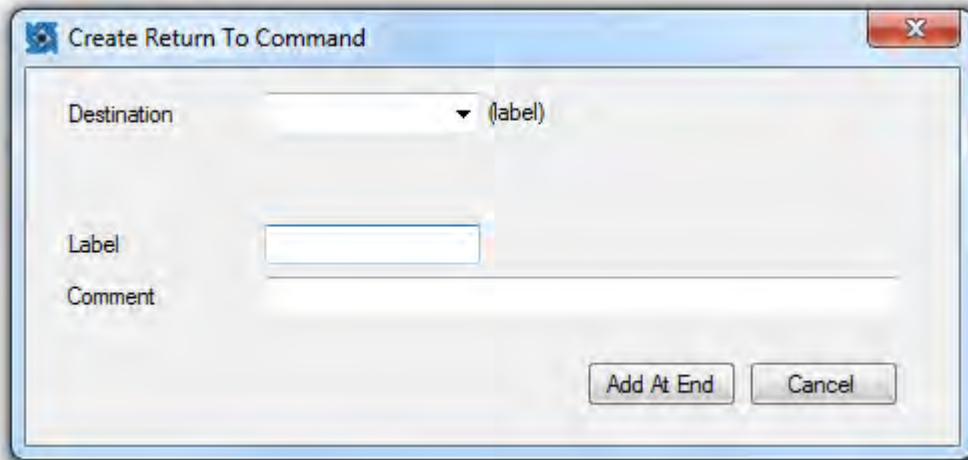
The following program runs a subroutine starting at label extend, which extends the motor 0.25", waits for the move to complete, then returns, then repeats.

Action	Label	Description	Comment
0	start	Goto Sub extend	
1		Goto start	
2			
3	extend	Extend 0.25 in	
4		Wait For Move	
5		Return	

This command is used in example 3.

## Return To

The return to command ends a subroutine, clears the stack, clears any pending interrupts, clears any “Jump N Times” commands, and goes to a specified label. For further explanation of subroutines, see Subroutines, page 26.



### Parameters

Destination: The label of the command that should be executed next.

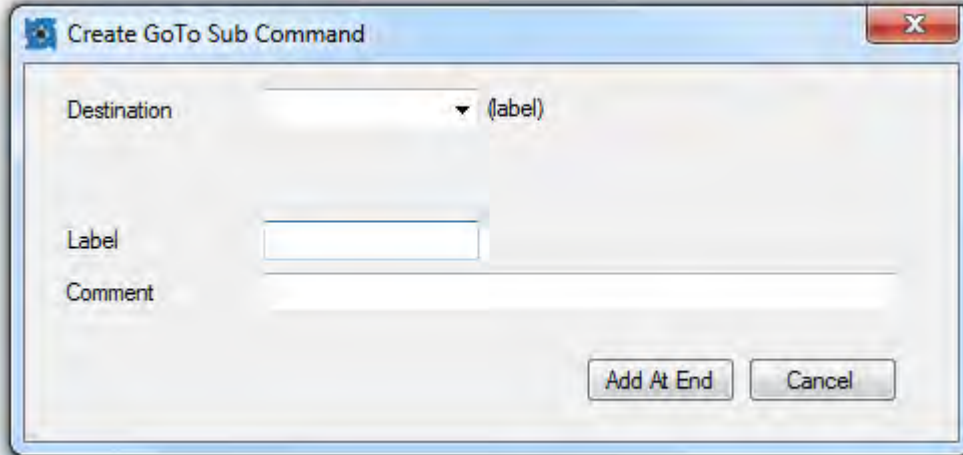
The following program runs a subroutine starting at label extend, which extends the motor 0.25”, waits for the move to complete, then exits the subroutine, goes to the abort command, and aborts the program.

Action	Label	Description	Comment
0	start	Goto Sub extend	
1		Goto start	
2			
3	extend	Extend 0.25 in	
4		Wait For Move	
5		Return To abort	
6			
7	abort	Abort	

This command is used in example 6.

## Goto Sub

The Goto Sub command goes to a subroutine starting with the specified label. For further explanation of subroutines, see Subroutines, page 26.



### Parameters

Destination: This is the label of the command that is the start of the subroutine.

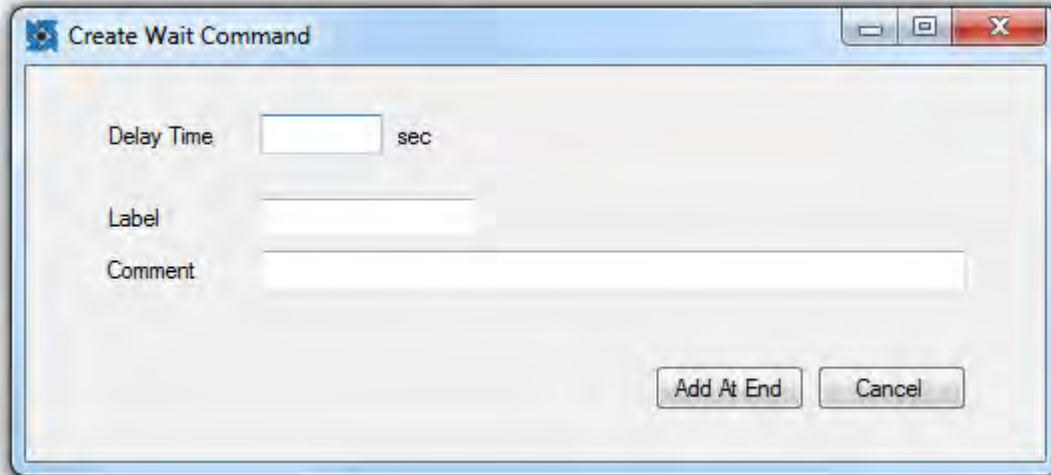
The following program runs a subroutine starting at label extend, which extends the motor 0.25", waits for the move to complete, then returns, then repeats.

Action	Label	Description	Comment
0	start	Goto Sub extend	
1		Goto start	
2			
3	extend	Extend 0.25 in	
4		Wait For Move	
5		Return	

This command is used in example 3.

## Wait

The wait command delays execution of the next command in the program for a specified time.



## Parameters

Delay Time: The amount of time that the program should wait.

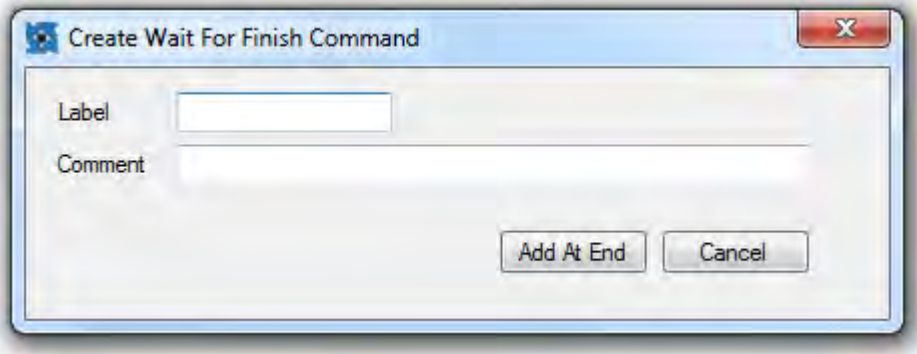
The following program begins moving the actuator at 1" per second, waits 0.5 seconds, then stops the motor.

Action	Label	Description	Comment
0		Go At Speed 1 in/sec	
1		Wait 0.5 sec	
2		Stop	

This command is used in examples 1, 3, and 4.

## Wait For Move

The Wait For Move command delays execution of the next command in the program until a move has completed. This command is automatically added after every Extend, Retract, and Move To, but can be removed if necessary.



### Parameters

This command has no parameters.

The following program extends the motor 1", and waits for the move to complete before executing any commands which may follow:

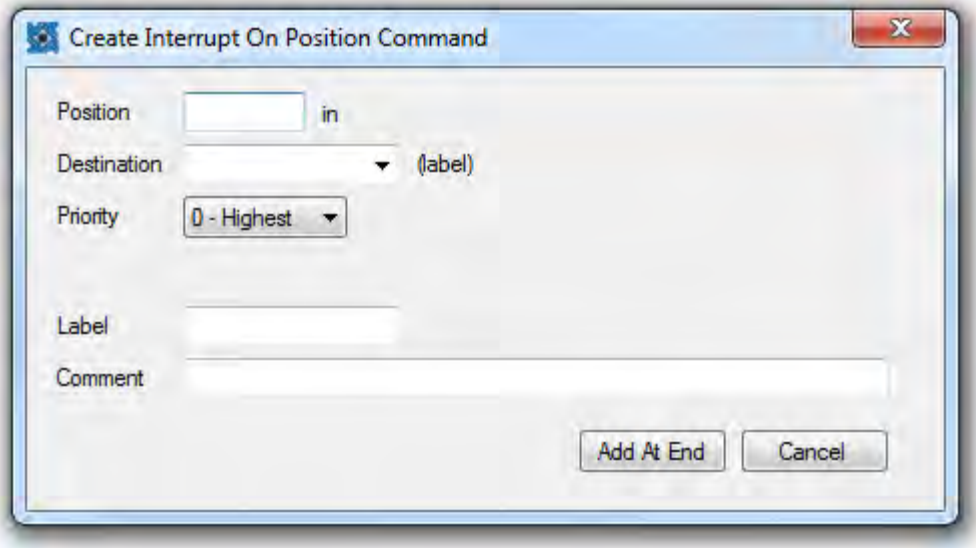
Action	Label	Description	Comment
0		Extend 1 in	
1		Wait For Move	

This command is used in examples 1, 2, 3, 4, 5, and 6.



## Int on Pos

The int on pos command sets an interrupt to be triggered when the motor reaches a specified position. For further explanation of interrupts, see Interrupts, page 27.



### Parameters

**Position:** The position, based upon the position counter of the drive, at which the interrupt should be triggered.

**Destination:** The label of the subroutine that should be executed when the position is reached.

**Priority:** The priority of the interrupts.

**Note:** If an interrupt is set for a position, and the position counter is adjusted through the use of the “Set Position” command, the interrupt will still occur at the same point.

**Example:** The drive is turned on, and an interrupt is set on position 0.5”. The “set Position” command is then used, changing what was the 0” position to the 1” position. The interrupt will now trigger when the drive’s position counter reaches 1.5”.

The following program sets an interrupt at position 1”, and then begins extending. When the actuator reaches the 1” position, the interrupt is triggered and the program aborts.

Action	Label	Description	Comment
0		Int On Position 1 in	Destination: abort
1		Go At Speed 1 in/sec	
2		Wait For Move	
3			
4	abort	Abort	

This command is used in example 6.

## Int on Input

The Interrupt on Input command allows an interrupt to be triggered when any of the inputs are changed. For further explanation of interrupts, see Interrupts, page 24.

**Create Interrupt on Input Command**

Input 1 Interrupt  
 Disabled  Enabled  Destination:  Trigger Type: Falling Edge Priority: 0 - Highest

Input 2 Interrupt  
 Disabled  Enabled  Destination:  Trigger Type: Falling Edge Priority: 0 - Highest

Input 3 Interrupt  
 Disabled  Enabled  Destination:  Trigger Type: Falling Edge Priority: 0 - Highest

Input 4 Interrupt  
 Disabled  Enabled  Destination:  Trigger Type: Falling Edge Priority: 0 - Highest

Label:   
 Comment:

Add At End Cancel

## Parameters

Each input has the same parameters.

Disabled/Enabled: Select “Enabled” if the input should cause an interrupt, select “Disabled” otherwise.

Destination: This is the label for the subroutine associated with the interrupt.

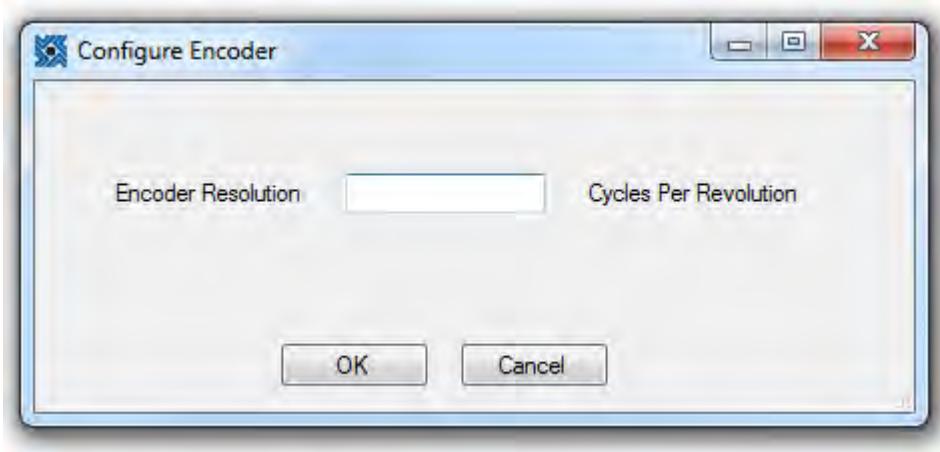
Trigger Type: If “Falling Edge” is selected, the interrupt will be triggered when the input goes from a logic high to a logic low. If “Rising Edge” is selected, the interrupt will trigger when the input goes from logic low to logic high. If “Both Edges” is selected, the interrupt will trigger any time the state of the input changes.

The following program sets an interrupt for the rising edge of input 1, and then loops continuously. When input one changes to logic high, output 1 is set to high.

Action	Label	Description	Comment
0		Int on Input INT, GP, GP, GP	
1	goto	Goto goto	
2	set Output	Set Outputs H X X X	
3		Return	

## Encoder

The Encoder command is used to verify the encoder resolution only. Please use the motor parameter entry to reset encoder counts per revolution. For more information on the encoder feature, see Encoder, page 21.

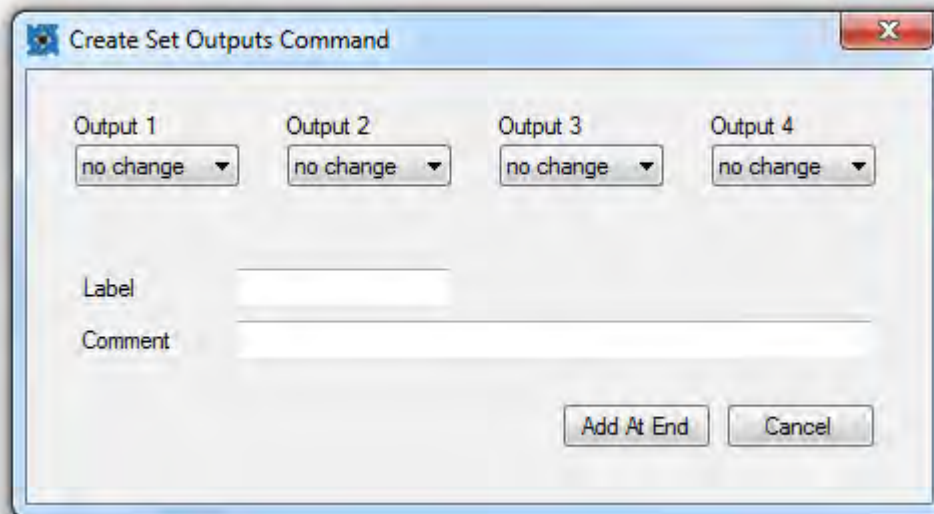


### Parameters

Encoder Resolution: This is the resolution of the encoder to be used, in cycles per revolution, which is equivalent to optical lines when using an optical encoder.

## Set Outputs

The Set Outputs command sets the logic state of the outputs.



### Parameters

Outputs: Each output is individually set as either high, which will bring the outputs to the opto-supply voltage, low, which brings the output to the opto-ground voltage, or no change, which will leave the output in its current state.

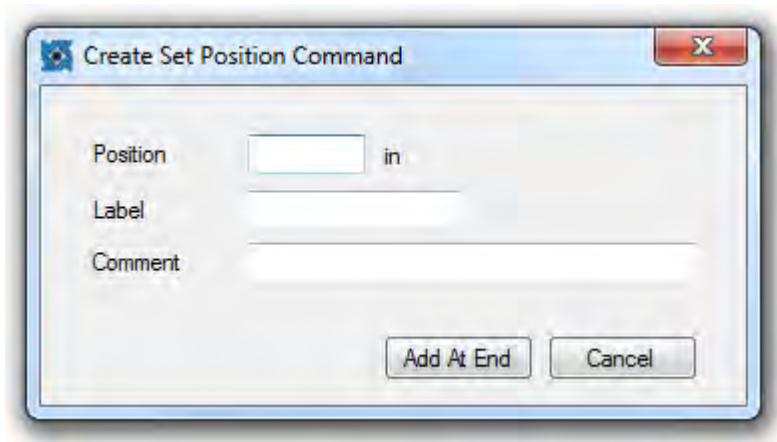
The following program sets output 1 high, waits 0.5 seconds, and then sets output 1 low. During this outputs 2 3 and 4 are unchanged.

Action	Label	Description	Comment
0		Set Outputs H X X X	
1		Wait 0.5 sec	
2		Set Outputs L X X X	

This command is used in example 3.

## Set Position

The Set Position command sets the position counter of the drive to the specified value.



## Parameters

Position: The value to which the current position of the drive should be set.

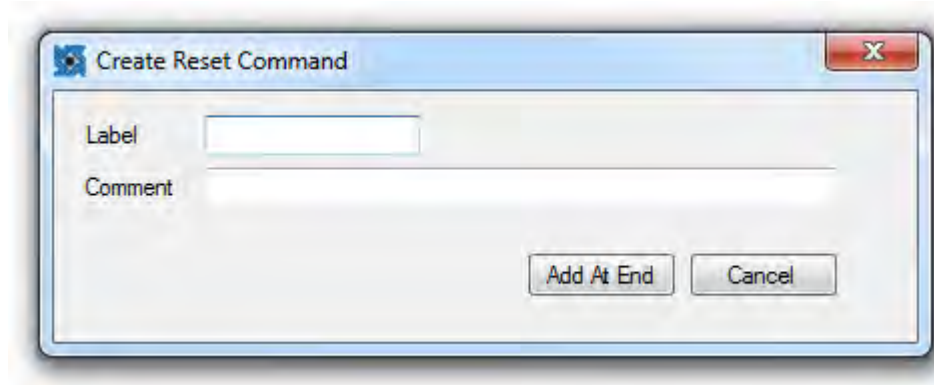
The following program retracts the motor 1", sets the position to 0", and then moves to the 0.5" position, which causes a 0.5" extend.

Action	Label	Description	Comment
0		Retract 1 in	
1		Wait For Move	
2		Set Position 0 in	
3		Move To 0.5 in	
4		Wait For Move	

This command is used in examples 4, 5, and 6.

## Reset

The Reset command restarts the drive, similar to turning the drive off and on. If used in a program, this command will not be executed less than 1/10<sup>th</sup> of a second after the drive has seen a reset, in order to prevent an uninterruptible cycle of resets.



## Parameters

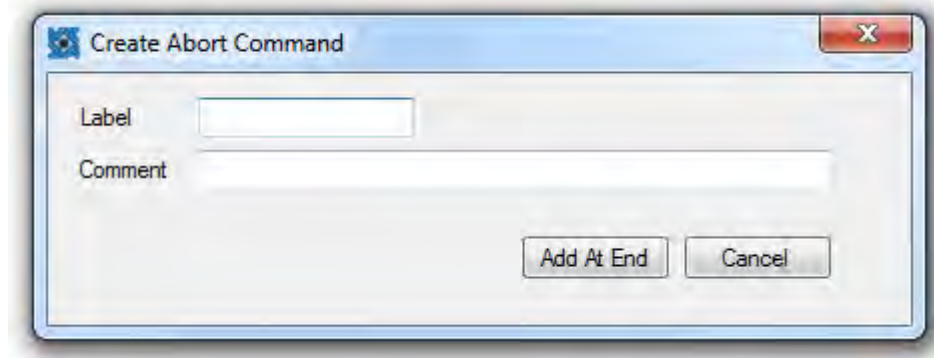
This command has no parameters.

The following program resets the drive.

Action	Label	Description	Comment
0		Reset	Resets the device

## Abort

The abort command immediately stops any moves without deceleration, applies the last specified holding current and halts execution of any running program.



## Parameters

This command has no parameters.

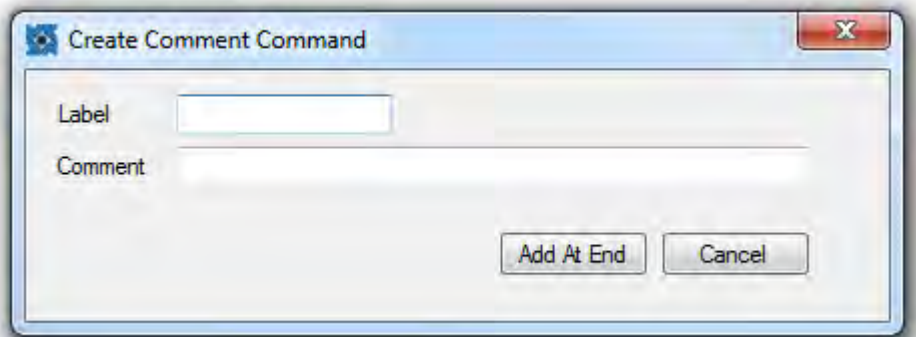
The following program aborts the program

Action	Label	Description	Comment
0		Abort	Ends the program

This command is used in examples 3 and 6.

## Comment

The comment command performs no action. This command is used to add extra information into a program for better documentation. Each comment line affords an additional 30 characters of comments. This command can also be used to add breaks between sections of code, making it more readable.



## Parameters

This command has no parameters.

The following program performs no actions.

Action	Label	Description	Comment
0			
1			
2			
3			
4			
5			
6			
7			

Comment  
This program performs no function, but does help to demonstrate the usefulness of the comment command

By adding in a comment command without a comment, I have made a line break for readability.



## Programming Examples

Because of the variety of motors that this product can be used with, examples for every product, or a general example for all products cannot be achieved.

Speeds, distances and other parameters will need to be changed for your particular actuator.

**Example One:** Extend the motor within a linear application 0.4 inches, wait one second, retract the motor 0.2 inches, wait one second, extend the motor 0.6 inches, wait 1 second, retract the motor 0.8 inches, wait 1 second, and repeat from the first extend indefinitely. Let the linear speed for each move be 1 inch per second.

We first want to extend 0.4 inches.

- Click the “Extend” button.
- Input the distance as 0.4 inches.
- Input the speed as 1 inch per second.
- All other parameters can be left as the defaults.
- Enter “Start” as the label.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.  
Note: You will notice that when the extend command populates in the program field on the screen, that it is followed by a second command that states “Wait For Move“. This is also true when using the Retract and Move To commands. This is a command to allow the move to finish before execution of the next command.

We now want to wait 1 second before moving again.

- Click the “Wait” button.
- Input the delay time as 1 second. No label or comment is necessary.
- Click “Add At End”. This places this command into the program.

We now want to retract 0.2 inches.

- Click the “Retract” button.
- Input the distance as 0.2 inches
- Input the speed as 1 inch per second
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to wait another second.

- Click the “Wait” button.
- The delay time will be 1 second, as previously entered. No label or comment is necessary.
- Click “Add At End”. This places this command into the program.

We now want to extend 0.6 inches.

- Click the “Extend” button. As before, many items are already populated. This time the distance and speed are also populated.
- Input the distance as 0.6 inches
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to wait one second.

- Click the “Wait” button.
- The delay time will be 1 second as previously entered. No label or comment is necessary.
- Click “Add At End”. This places this command into the program.

We now want to make the final move of retracting 0.8 inches.

- Click the “Retract” button.
- Input the distance as 0.8 inches
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to wait one second.

- Click the “Wait” button.
- The delay time will be 1 second as previously entered. No label or comment is necessary.
- Click “Add At End”. This places this command into the program.

We now want to repeat.

- Click the “Goto” button.
- Enter the destination as “Start”.
- Click “Add At End”. This places this command into the program.

The completed program looks as follows:

Haydon Kerk IDEA Drive Interface Program (Program Mode)

File Edit Realtime Input Configuration Drive Commands Programs On Drive Encoder Help **Haydon kerk**

Motion: Extend Retract Move To Go At Speed Stop E-Stop

Program Flow: Jump N Times Goto Goto If Return Return To Goto Sub Wait Wait For Move Int on Pos

Other: Set Outputs Set Position Reset Abort Comment

Action	Label	Description	Comment
0	Start	Extend 0.4 in	
1		Wait For Move	
2		Wait 1 sec	
3		Retract 0.2 in	
4		Wait For Move	
5		Wait 1 sec	
6		Extend 0.6 in	
7		Wait For Move	
8		Wait 1 sec	
9		Retract 0.8 in	
10		Wait For Move	
11		Wait 1 sec	
12		Goto Start	

Program Edit: Program Name: [ ] Copy Paste Remove New View / Edit Plot Download

Run Control: Program To Run: [ ] Start Stop

I/O and Position: Current Position: 0.000 in  
 Inputs: 1 2 3 4 (all green)  
 Outputs: [ ] [ ] [ ] [ ]

Program Length: **240 bytes / 1 pages**

Ready...

**Example Two:** This example will extend the motor within a linear application .5”, retract the motor .5”, then repeat those two moves 4 more times. Once the repetitions are complete, the motor will extend 1”. Let the linear speed for each move be 1 inch per second.

We first want to extend 0.5”.

- Click the “Extend” button.
- Input the distance as 0.5 inches.
- Input the speed as 1 inch per second.
- All other parameters can be left as the defaults.
- Enter “Start” as the label.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Note: You will notice that when the extend command populates in the program field on the screen, that it is followed by a second command that states “Wait For Move“. This is also true when using the Retract and Move To commands. This is a command to allow the move to finish before execution of the next command.

We now want to retract 0.5”.

- Click the “Retract” button.
- Input the distance as 0.5 inches
- Input the speed as 1 inch per second
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to repeat the first two moves four times.

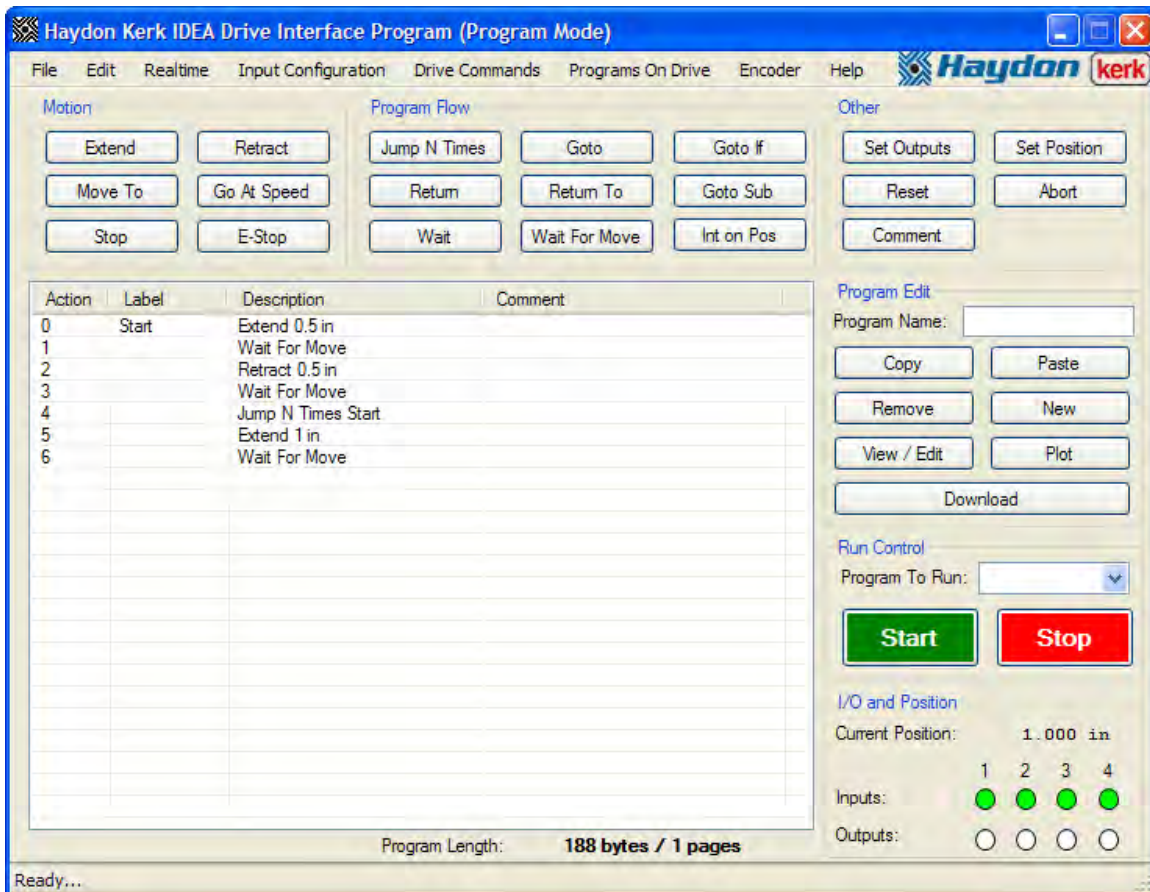
- Click the “Jump N Times” button.

- Enter the destination as “Start”
- Enter the number of jumps as 4.
- Click “Add At End”. This places this command into the program.

We now want to extend 1”.

- Click the “Extend” button.
- Input the distance as 1”.
- Input the speed as 1 inch per second.
- All other parameters can be left as the defaults.
- No Label is required
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

The completed program looks as follows:



**Example Three:** This example will extend the motor within a linear application .5", turn all four outputs high, wait .5 seconds, turn all outputs low, retract the motor .5", turn all four outputs high, wait .5 seconds, turn all outputs low, then end the program. We will accomplish this by using a subroutine.

We first want to extend 0.5".

- Click the "Extend" button.
- Input the distance as 0.5 inches.
- Input the speed as 1 inch per second.
- All other parameters can be left as the defaults.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the "Add At End" button. This places the command in program.

Note: You will notice that when the extend command populates in the program field on the screen, that it is followed by a second command that states "Wait For Move". This is also true when using the Retract and Move To commands. This is a command to allow the move to finish before execution of the next command.

We now want to use a subroutine to toggle the outputs on and off.

- Click the "Goto Sub" button.
- Enter the destination as "Toggle"
- No label is required.
- You can add a comment in the comment line if you wish.
- Click the "Add At End" button. This places the command in program.

We now want to retract 0.5".

- Click the "Retract" button.
- Input the distance as 0.5 inches
- Input the speed as 1 inch per second

- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to use a subroutine to toggle the outputs on and off.

- Click the “Goto Sub” button.
- Enter the destination as “Toggle”
- No label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to stop the program.

- Click the “Abort” button.
- Input the distance as 1”.
- Input the speed as 1 inch per second.
- All other parameters can be left as the defaults.
- No Label is required
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Now we have the main body of the program, but we still need the “Toggle” subroutine.

- Click the “Set Outputs” button.
- Select “High” for each of the four outputs.
- Enter “Toggle” as the label
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Now we need to wait 0.1 seconds.

- Click the “Wait” button.



- Enter 0.5 seconds as the delay time
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Now we need to set the outputs low.

- Click the “Set Outputs” button.
- Select “Low” for each of the four outputs.
- No label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Now we need to return to the main body of the program. To go back to where the subroutine was called from, we use a “Return” command.

- Click the “Return” button.
- No label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

The completed program looks as follows:

Haydon Kerk IDEA Drive Interface Program (Program Mode)

File Edit Realtime Input Configuration Drive Commands Programs On Drive Encoder Help **Haydon kerk**

**Motion**

Extend Retract  
Move To Go At Speed  
Stop E-Stop

**Program Flow**

Jump N Times Goto Goto If  
Return Return To Goto Sub  
Wait Wait For Move Int on Pos

**Other**

Set Outputs Set Position  
Reset Abort  
Comment

Action	Label	Description	Comment
0		Extend 0.5 in	
1		Wait For Move	
2		Goto Sub Toggle	
3		Retract 0.5 in	
4		Wait For Move	
5		Goto Sub Toggle	
6		Abort	
7	Toggle	Set Outputs H H H H	
8		Wait 0.5 sec	
9		Set Outputs L L L L	
10		Return	

**Program Edit**

Program Name: example3

Copy Paste  
Remove New  
View / Edit Plot  
Download

**Run Control**

Program To Run: example3

**Start** **Stop**

**I/O and Position**

Current Position: 0.000 in

Inputs: 1 2 3 4

Outputs:

Program Length: **168 bytes / 1 pages**

Ready...

**Example Four:** This example is different in that the motor within a linear application does not need to start in the fully retracted position. We will perform a homing routine, to find the fully retracted position, and then move to 0.5” from that point, wait 1 second, and then return to the fully retracted position.

We first want to retract the full stroke of the actuator.

- Click the “Retract” button.
- Input the distance as 1 inch.
- Input the speed as 0.5 inch per second.
- Select “1/4” as the step mode.
- All other parameters can be left as the defaults.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Note: You will notice that when the retract command populates in the program field on the screen, that it is followed by a second command that states “Wait For Move“. This is also true when using the Extend and Move To commands. This is a command to allow the move to finish before execution of the next command.

We are now at the fully retracted position; to keep track of this we will use the “Set Position” command.

- Click the “Set Position” button.
- Enter a position of 0”.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

We now want to move to the 0.5” position.

- Click the “Move To” button.
- Enter a position of 0.5”.
- Enter a speed of 1” per second.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to wait 1 second.

- Click the “Wait” button.
- Enter a delay time of 1 second.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

We now want to move back to the 0” position.

- Click the “Move To” button.
- Enter a position of 0”.
- All other parameters are populated.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

The completed program looks as follows:

Haydon Kerk IDEA Drive Interface Program (Program Mode)

File Edit Realtime Input Configuration Drive Commands Programs On Drive Encoder Help **Haydon kerk**

**Motion**

Extend Retract

Move To Go At Speed

Stop E-Stop

**Program Flow**

Jump N Times Goto Goto If

Return Return To Goto Sub

Wait Wait For Move Int on Pos

**Other**

Set Outputs Set Position

Reset Abort

Comment

Action	Label	Description	Comment
0		Retract 1 in	
1		Wait For Move	
2		Set Position 0 in	
3		Move To 0.5 in	
4		Wait For Move	
5		Wait 1 sec	
6		Move To 0 in	
7		Wait For Move	

**Program Edit**

Program Name:

Copy Paste

Remove New

View / Edit Plot

Download

**Run Control**

Program To Run:

**Start** **Stop**

**I/O and Position**

Current Position: 0.000 in

Inputs: 1 2 3 4

Outputs:

Program Length: **184 bytes / 1 pages**

Ready...

**Example Five:** In this example, we will first find the fully retracted position of the motor within a linear application, then the motor will move to the 0" position if input 1 is high and input 2 is low, move to the 1" position if input 1 is low and input 2 is high, or stop if inputs 1 and 2 are both high, or both low.

We first want to retract the full stroke of the actuator.

- Click the "Retract" button.
- Input the distance as 1".
- Input the speed as 0.5 inch per second.
- Select "1/4" as the step mode.
- All other parameters can be left as the defaults.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the "Add At End" button. This places the command in program.

Note: You will notice that when the retract command populates in the program field on the screen, that it is followed by a second command that states "Wait For Move". This is also true when using the Extend and Move To commands. This is a command to allow the move to finish before execution of the next command.

We are now at the fully retracted position; to keep track of this we will use the "Set Position" command.

- Click the "Set Position" button.
- Enter a position of 0".
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click "Add At End". This places this command into the program.

We now move based on the input status. Set up the first "Goto If"

- Click the "Goto If" button.

- Enter “Retract” as the destination.
- Set Input 1 is “High”
- Set Input 2 as “Low”
- Set Inputs 3 and 4 as “Not Tested”
- Enter “Test” as the label.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Set up the second “Goto If”

- Click the “Goto If” button.
- Enter “Extend” as the destination.
- Set Input 1 is “Low”
- Set Input 2 as “High”
- Set Inputs 3 and 4 as “Not Tested”
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

If we reach this line, then either both inputs are high, or both inputs are low. So we want to stop the motor.

- Click the “Stop” button.
- All parameters can be left at the defaults.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to check if the input conditions have changed, so we go back to the Goto Ifs.

- Click the “Goto” button.
- Enter “Test” as the destination.

- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now need the commands that will be used to move the motor. We will start with the “Retract” move.

- Click the “Move To” button.
- Enter a position of 0”.
- Enter “Retract” as the label.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

We now want to check if the input conditions have changed, so we go back to the Goto lfs. We don’t want the move to finish before we check the inputs again, so we will remove the “Wait For Move” command.

- Click the line that holds the “Wait For Move” command
- Click the “Remove” button
- Click “Yes” to confirm the removal of the command.
- Click the “Goto” button.
- Enter “Test” as the destination.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We will next add the “Extend” move.

- Click the “Move To” button.
- Enter a position of 1”.
- Enter “Extend” as the label.
- You can add a comment in the comment line if you wish.



- Click “Add At End”. This places this command into the program.

We now want to check if the input conditions have changed, so we go back to the Goto lfs. We don't want the move to finish before we check the inputs again, so we will remove the “Wait For Move” command.

- Click the line that holds the “Wait For Move” command
- Click the “Remove” button
- Click “Yes” to confirm the removal of the command.
- Click the “Goto” button.
- Enter “Test” as the destination.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

The completed program looks as follows:

Haydon Kerk IDEA Drive Interface Program (Program Mode)

File Edit Realtime Input Configuration Drive Commands Programs On Drive Encoder Help **Haydon kerk**

**Motion**

Extend Retract

Move To Go At Speed

Stop E-Stop

**Program Flow**

Jump N Times Goto Goto If

Return Return To Goto Sub

Wait Wait For Move Int on Pos

**Other**

Set Outputs Set Position

Reset Abort

Comment

Action	Label	Description	Comment
0		Retract 1 in	
1		Wait For Move	
2		Set Position 0 in	
3	Test	Goto If Retract	
4		Goto If Extend	
5		Stop	
6		Goto Test	
7	Retract	Move To 0 in	
8		Goto Test	
9	Extend	Move To 1 in	
10		Goto Test	

**Program Edit**

Program Name:

Copy Paste

Remove New

View / Edit Plot

Download

**Run Control**

Program To Run:

**Start** **Stop**

**I/O and Position**

Current Position: 0.000 in

Inputs: 1 2 3 4

Outputs:

Program Length: **252 bytes / 1 pages**

Download Complete

**Example Six:** In this example, we will first find the fully retracted position of the motor within a linear application. The motor will continuously move in the extend direction, until getting 0.9” from the fully retracted position. When the 0.9” position is reached, the motor will retract back to the 0” position, and resume the extend. If at any time during the program, input 1 changes state, the program will abort.

We first need to set up the interrupt triggered by the input.

- Click the “Int on Input” Button.
- Select “Enabled” for Input 1 Interrupt
- Enter “Abort” as the destination.
- Select “Both Edges” as the trigger type.
- Select “0-Highest” as the priority.
- Click the “Add At End” button. This places the command in program.

We now want to retract the full stroke of the actuator.

- Click the “Retract” button.
- Input the distance as 1”.
- Input the speed as 0.5 inch per second.
- Select “1/4” as the step mode.
- All other parameters can be left as the defaults.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

Note: You will notice that when the retract command populates in the program field on the screen, that it is followed by a second command that states “Wait For Move“. This is also true when using the Extend and Move To commands. This is a command to allow the move to finish before execution of the next command.

We are now at the fully retracted position; to keep track of this we will use the “Set Position” command.

- Click the “Set Position” button.
- Enter a position of 0”.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

In order to force the actuator to go back to the 0” position when the 0.9” position is reached, we will use an interrupt based on position.

- Click the “Int On Pos” button.
- Enter a position of 0.9”.
- Enter “Retract” as the destination.
- Select “1” as priority.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

We now want to begin moving.

- Click the “Go at Speed” button.
- Select “Extend” as the direction
- Enter 1” per second as the speed.
- Enter “Extend” as the label.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now want to continuously loop onto the “Go At Speed” command.

- Click the “Goto” button.

- Enter “Extend” as the destination.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now need the interrupt subroutines. We will start with the interrupt subroutine for the interrupt based on position.

- Click the “Move To” button.
- Enter a position of 0”.
- Enter a speed of 1” per second.
- Enter “Retract” as the label.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now need to exit the subroutine.

- Click the “Return To” button.
- Enter “Extend” as the destination.
- No Label is required.
- You can add a comment in the comment line if you wish.
- Click the “Add At End” button. This places the command in program.

We now need a subroutine for the interrupt based on the input.

- Click the “Abort” button.
- Enter “Abort” as the label.
- You can add a comment in the comment line if you wish.
- Click “Add At End”. This places this command into the program.

The completed program looks as follows:

Haydon Kerk IDEA Drive Interface Program (Program Mode)

File Edit Realtime Drive Commands Programs On Drive Help

**Motion**      **Program Flow**      **Other**

Action	Label	Description	Comment
0		Int on Input INT, GP, GP, GP	
1		Retract 1 in	
2		Wait For Move	
3		Set Position 0 in	
4		Int On Position 0.9 in	
5	Extend	Go At Speed 1 in/sec	
6		Goto Extend	
7	Retract	Move To 0 in	
8		Wait For Move	
9		Return To Extend	
10	Abort	Abort	

**Program Edit**

Program Name:

**Run Control**

Program To Run:

**I/O and Position**

Current Position: **0.000 in**

Inputs:     1     2     3     4  
 Outputs:               

Program Length: **224 bytes / 1 pages**

Ready...

## The IDEA Drive Menu Items:

- **File:** This menu gives access to functions for manipulation of program files and the User interface itself.
  1. **New:** Clears the current program so a new program can be written.
  2. **Open:** Opens a previously saved program file.
  3. **Save:** Saves the current program to a file.
  4. **Add File:** Adds a previously saved program file to the end of the current program.
  5. **Recover Autosave:** Opens the most recently autosaved file.
  6. **Print:** Prints the current program
  7. **Motor Characteristics:** Displays motor performance curve depicting drive limitations based on motor specifications
  8. **Preferences:** Contains options for GUI behavior.
  9. **Restart:** Exits the user interface and opens a new user interface.
  10. **Exit:** Closes the user interface.
- **Edit:** This menu gives access to functions for manipulating the current program.
  1. **Undo:** Restores the program to what it was before the last action.
  2. **Redo:** Restores the program to what it was before an undo.
  3. **Cut:** Removes a selected line or lines from the program and copies them to be pasted later.
  4. **Copy:** Copies a selected line or lines to be pasted later.
  5. **Paste:** Inserts previously copied lines into a program.
  6. **Select all:** Selects all lines of the program.
- **Mode:** The third menu item toggles between the Realtime Mode and Program Mode.

- **Drive Commands:** This menu allows access to various drive functions.
  1. **Display Table of Contents:** Gives a listing of the programs on the drive and their page locations. Also provides a graphical representation of the used space in the drive.
  2. **Set Startup Program:** Used to choose what program, if any, should begin execution when the drive starts up.
  3. **Delete Program:** Used to remove programs from the drive.
  4. **Input Simulation:** This item toggles the drive between using the true input status, or the simulated status through the user interface
  5. **Set Drive Address:** Used to change the address of the current drive.
  6. **Configure Motor Feedback:** Used to configure motor commutation and encoder settings
  7. **Toggle Profile Select:** Used to set motion profile settings; trapezoidal or s-curve
  8. **Adjust Control Loop Gain:** Used to adjust the sensitivity of the encoder feedback loop used for drive commutation
  9. **Set/Change Password:** Used to configure the password of the drive.
  10. **Restore Factory Defaults:** Removes password protection from the drive, and removes all programs on the drive.
  11. **Firmware Version:** Used to find the firmware version of the drive.
  12. **Update Firmware:** Used to reprogram the drive with the firmware version that was most recent when the user interface was installed.
  
- **Communications Mode:** Opens the communications mode dialog box. This allows for switching between the for communications modes.
  
- **Programs on Drive:** Shows a list of the programs that are on the drive. Clicking on a program loads that program to the program area. The startup program, if one exists, appears in bold on this list.



- **Help:** Allows access to information about the user interface and drives.
  1. **About:** Displays a brief description of Haydon Kerk and its products.
  2. **User's Manual:** Displays this manual.
  3. **Communications Manual:** Opens the communications manual.
  4. **Hardware Manuals:** Manuals for individual products.

## **Glossary:**

**Abort:** Stops movement of the actuator with holding current and ends any running program.

**Accel Current:** Allow the user to implement up to a 30% increase in peak current per phase during the beginning of the acceleration ramp.

**Accel Rate:** The acceleration rate to be used with a move.

**Clear:** Clear the entire program from the program screen

**Comment:** Allows the user to insert comments within the program

**Copy:** Allows the user to copy a given program line or lines and insert them elsewhere in the program

**Current position box:** Indicates the current position of the motor

**Decel Current:** Allows the user to implement up to a 30% increase in peak current per phase during the end of the deceleration ramp.

**Decel Rate:** The deceleration rate to be used with a move.

**Delay Time (in reference to a move):** The time between when the last step in a move profile is taken, and when the current is set to the hold current.

**Delay Time (in reference to a “Wait” command):** The amount of time that the wait command should delay execution of the next command.

**Destination:** The address to which the program should branch.

**Distance:** How far a move should go.

**Download:** Allows the program to be downloaded into drive

**E-Stop:** Abruptly stops the motor without any deceleration

**Encoder:** A feedback device that converts position data to an electronic signal that the drive keeps track of.

**Extend:** Extends the motor within a linear system forward, or clockwise. The user inputs the distance and speed. Items such as run current and hold current are auto populated based on the characteristics of the motor. These values can be overridden provided the inserted value does not exceed the devices limitations.

**Go at Speed:** Extends (CW) or retracts (CCW) the motor at a given speed indefinitely.

**Goto:** Branching statement for programming. Branches to a destination label.

**Goto if:** Branching statement for programming. Branches to a destination label if the input conditions are met.

**Goto Sub:** Branching statement for programming that navigates the program to a subroutine.

**Hold Current:** The maximum, peak current limit applied to the phases of the motor which the drive may use to hold the load in position when the motor is at a standstill.

**I/O box:** Displays the state of each general purpose I/O.

**Int on Input:** Interrupt on Input. This is an interrupt that occurs when an input changes state.

**Int on Pos:** Interrupt on Position. This is an interrupt that occurs when the encoder reaches a specific position.

**Interrupt:** An asynchronous event that causes the execution of a subroutine.

**Jump N times:** Allows the program to jump N times to a specified label

**Label:** A string that identifies a command. Used to branch to the command.

**Move to:** Moves the motor to a specified position.

**Paste:** Used in conjunction with the copy or cut functions. After one or more commands are selected, another location in the program is then highlighted by the user. The paste button is then pressed and the line or lines are inserted above the highlighted line.

**Plot:** Any move can be shown as a plot of speed vs time. Highlight the move command of interest then press the plot button.

**Position:** A location based upon the encoder counter.

**Priority:** The determining factor in which interrupts are serviced first.

**Program name box:** This is the location on the screen where the name of the program is inputted by the user. The program is stored in the drive under this name.

**Program to run box:** This is a drop down menu that list the programs stored on the drive. Double clicking on a given name populates the program into the program screen and creates that program as the active program in the drive.

**Remove:** This is used to remove one or more lines from a program. Highlight the lines to be removed. Then click the remove button.

**Reset:** This command simulates turning the drive off and on again.

**Retract:** Retracts the motor within a linear system reverse, or counter clockwise. The user inputs the distance and speed. Items such run current and hold current are auto populated based on the characteristics of the motor. These values can be over ridden provided the inserted value does not exceed the devices limitations.

**Return:** Used in conjunction with the goto sub command or interrupts. At the end of a subroutine the “return” command returns to the program to the very next line after the Goto sub line command, or the command that was going to be executed before the interrupt was triggered.

**Return to:** Used in conjunction with the goto sub command or interrupts. At the end of a subroutine the “return to” command returns to the command at a specified label location.

**Run Current:** The maximum allowable, peak current to be applied to the motor phases during a move.

**Set outputs:** Allow the programmer to set general purpose outputs.

**Set position:** Used to change the current position. Using this command the position counter can be adjusted, usually after a homing routine. Then other commands such as “move to” or “interrupt on position” can be used in relationship with this set position.

**Speed:** The desired top speed for a move.

**Start (Large green button):** Starts running a program.

**Stop:** Stops the movement of the motor with a specified deceleration

**Stop (Large red button):** Immediately stops the motor and program when pressed.

**Subroutine:** A section of code used that is entered by using an interrupts, or the “Goto Sub” command. Subroutines must end with a “Return” or “Return To” command.

**View / Edit:** After highlighting a specific line in a program, the “View / Edit” button can be pressed causing the details for that line to display. These details can then be modified and updated.

**Wait:** Allows the programmer to put in a time specific time delay.

**Wait for move:** Delays execution of the next line of the program until the motor has come to a stop.