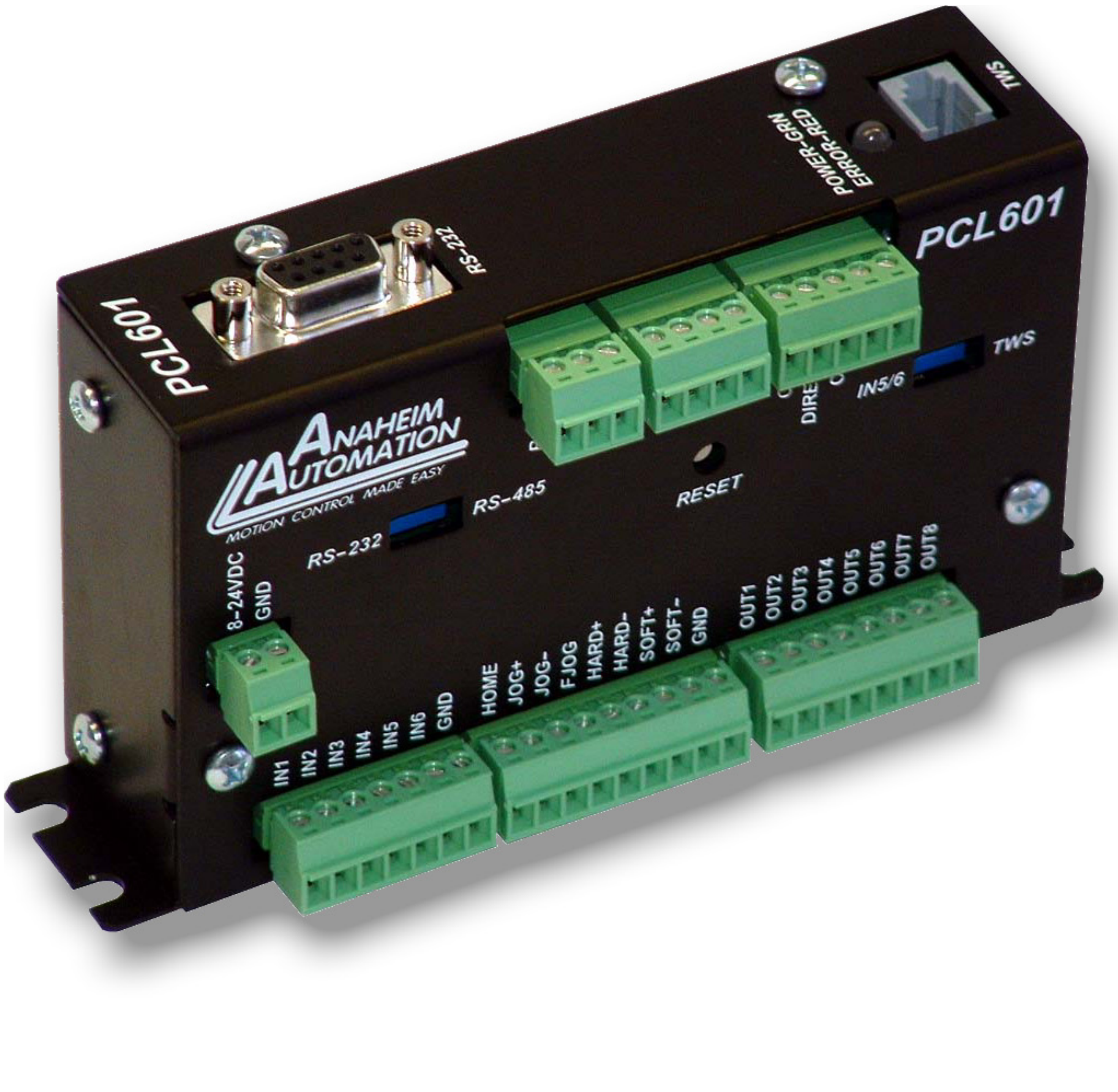
PCL601MOD
Programmable Step Motor Controller

User's Guide



ANAHEIM AUTOMATION, INC.
4985 E. Landon Drive Anaheim, CA 92807 (714) 992-6990 fax: (714) 992-0471
e-mail: info@anaheimautomation.com website: www.anaheimautomation.com

#L011268 July 2018

Table of Contents

Section 1: Introduction 3
Description........ 3
Methods of Communication........ 4
Baud Rate........ 4
RS232 Protocol - SW1 in RS232 position 4
RS485 Protocol - SW1 in RS485 position 4
Terminating Resistor........ 5
Axis Selection........ 5
Status LED 5
Interfacing With A Driver........ 5
Technical Support........ 5
Electrical Specifications........ 6
Ordering Information........ 6
Dimensions/Switch Locations........ 7
Wiring Diagrams........ 7
Terminal Descriptions 8
Connector Descriptions 8
Slide Switch Descriptions 8
Section 2: Functions 9
Section 3: SMC60WIN Software 13
File Menu........ 14
Setup Menu 14
Setup - Axis Menu 14
Program Menu........ 15
Program - Autostart Program Menu........ 15
Edit Menu 15
Help Menu 16
“The Unit is Connected” / “The Unit is NOT Connected” 16
Toolbar........ 17
Tab Sheets........ 17
Add/Change/Insert Commands 22
Calculator 27
Section 4: Communication Protocols........ 28
Modbus RTU 28
Anaheim Automation ASCII 32
Section 5: Troubleshooting 42
Error Codes 43
Section 6: Tutorials 44
Appendix 1: PCL601’s Memory Map 62
Appendix 2: CRC Generation Algorithm........ 64
Appendix 3: Supported Modbus Function Codes........ 65
Appendix 4: Switching to AA ASCII Protocol 68
Appendix 5: ASCII Table for Direct Mode 70
Appendix 6: Firmware Revisions........ 70

Section 1: Introduction

The PCL601 is a Modbus RTU slave single-axis stepper motor controller containing 2 Kbytes of nonvolatile stored programming space and quadrature encoder feedback. The user can control the device with a Human Machine Interface as well as write a program and then have the PCL601 autostart the program on power up. It provides flexible, independent control of stepper motors from computers, or any machine controller with a serial port. It is also capable of standalone operation, making it an embedded machine controller. The easy-to-use Windows software, SMC60WIN, can be used to directly control motion and to program the PCL601. The PCL601 also has the ability for real time functions. “Direct Mode” is used to directly control motion for Real Time Movements through serial communication.

The PCL601 has a comprehensive range of commands, which are easy to remember for direct movement of a stepper motor. The PCL601 communicates via either an RS232 or RS485 bidirectional serial data bus. Up to 99 PCL601’s can be networked from one communication port on your PC or PLC, utilizing the RS485 communication protocol. Special functions of the PCL601 include 8 programmable open collector outputs and 6 TTL, CMOS and 24V compatible inputs, a quadrature encoder input with the ability to autocorrect, an analog input to control either maximum speed or absolute position, registration mark indexing during a slew command, an output that will trigger during an index command at an absolute position, and a thumbwheel input for indexing a motor. The PCL601 can be powered with a DC (5VDC or 8 -24VDC) voltage and uses only 2 watts at 24VDC and only a 1/2 watt at 5VDC (with no connections to terminal blocks).

Description

The PCL601 stepper motor controller provides independent programming of acceleration/deceleration, base speed (start up speed), max speed (running speed), jog speed, and the number of steps to be taken in both relative and absolute positioning modes. On absolute positioning moves, the PCL601 automatically determines the proper direction to go and the number of steps to take. The relative positioning will move a number of steps in the direction that the user defines. The PCL601 also has specific functions such as *index-on-the-fly,* which during a slew move will move a predefined number of steps after an input has been triggered. *Output-on-the-fly,* which will trigger an output on for 50uS during an indexing move at an absolute position and repeat triggering the output on after a given number of steps. An analog input can be used to set either the maximum speed or goto an absolute position based between the upper and lower programmable limits. A seven decade thumbwheel switch can be read for relative indexing. The PCL601 also has a high level programming command set that includes: branching, looping, conditional statements, time delays, text strings, and I/O which the user can use in the programming mode to fully control all machine functionality.

A home input, a set of bidirectional hard and soft limit switch inputs and bidirectional jog inputs are provided for each axis. These features are generally required in most machine control designs. 6 testable TTL, CMOS and 24V compatible inputs and 8 programmable open-collector outputs are provided per axis. The I/O may be used for monitoring and controlling machine operation and/or interaxis coordination. The I/O are accessible independent of the busy state of the axis controls.

The PCL601 has a built-in programmable reset circuit. Reset is automatic on power-up, or by pressing the external reset button. A free software download can be obtained at anaheimautomation.com, and it contains this user’s manual, along with the SMC60WIN software and PCL601 program examples. The software allows you to write and change programs that are to be stored in the PCL601 for autostart use, and also upload the program that is stored in the PCL601 itself for editing and viewing. The software also allows you to save the programs onto your computer hard drive, and easily retrieve them when needed.

Methods of Communication

There are a few methods for communicating with the PCL601. The first is to use a Human Machine Interface that supports the Modbus RTU protocol. With this method, the user can create a custom graphical interface and I/O controlled process to be run on the HMI and control the PCL601. Through the HMI, the user can control all the features of the controller as well as monitor its operation. It is important to note that this method can only be used in the Modbus RTU protocol using either an RS232 or RS485 connection. The second mehod is to directly send commands to the device via a terminal program and a serial connection, either RS232 or RS485. In order to do this, the user must first switch the device to use the AA ASCII communication protocol by writing the command code 11 to the command register (see page 29). Once in AA ASCII mode, the user can use AA ASCII commands described in this manual in order to control the features of the PCL601.
Another way to give commands to the device is to use the software program SMC60WIN to either manually control or to write and send programs. This method is mainly used for programming the device, so that it can automatically execute the given program. For example, this device can replace simple motion control and/or I/O functions when minimal quantities of I/O are required to control specific machinery. Simple motion profiles that can operate with 6 or less inputs and 8 or less outputs can utilize this controller. In addition, once the controller has been programmed, it can be interfaced to an HMI, so that its operation can be monitored and/or stopped at any given time if necessary.

Baud Rate

A term used frequently in serial data communications, a “baud” is defined as the reciprocal of the shortest pulse duration in a data word signal, including start, stop, and parity bits. This is often taken to mean the same as “bits per second”, a term that expresses only the number of “data” bits per second. Very often, the parity bit is included as an information or data bit. **The PCL601MOD accepts a baud rate of 38400 only.**

RS232 Protocol - SW1 in RS232 position

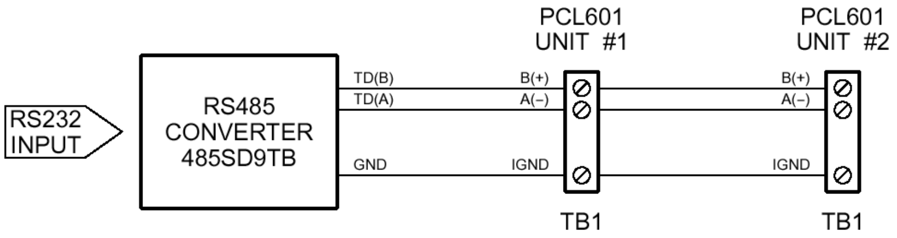
The PCL601 is a DCE device, therefore it will transmit on pin 2 and receive on pin3 of the DB9 RS-232 connector . The RS232 serial communication mode is single ended. This means that for each signal there is one wire, and a common ground reference used by all the signals. The PCL601 does not use handshaking, thus the CTS and RTS lines are internally connected, and the CD, DTR and DSR lines are internally connected inside the PCL601. The signal line maintains levels of +5VDC to +15VDC and -5VDC to -15VDC. For a valid logic level in the controller, the voltage must be at least +/-3 volts. RS232 works at distances of up to 50 feet maximum. **RS232 is susceptible to electrical noise, and should not be used in noisy areas. Always use the shortest cable connection possible. NOTE: Keep controller wiring separated from motor cable/wiring.**

RS485 Protocol - SW1 in RS485 position

The RS485 protocol mode is as follows; On board receivers will remain in active mode indefinitely. Transmitters must be turned off when the unit is not sending data, to prevent the line from sending and receiving data at the same time. Therefore when the PC is transmitting data its driver will be turned on and each of the units connected will have their drivers off. If they are requested to send data back to the PC, the selected unit will turn it’s driver on to send the data and then turn it off after it has completed transmission. Note: The above protocol is done internally between the converter and the PCL601. The RS485 method of communication allows increased noise immunity and increased communication distance of up to 4000 feet without repeaters. RS485 repeaters allow an additional 4000 feet per repeater. The PCL601 is designed for two wire configuration. The 2 wire configuration makes use of the tristate capabilities of RS485 to allow a single pair of wires to share transmit and receive signals for half duplex communications. This “two wire” configuration (note that an additional ground conductor must be used) reduces cabling cost. **NOTE: Keep control wiring separated from motor cable/wiring.**

RS232 to RS485 for multiple units or cables longer than 50ft

The PCL601 can be connected to your PC serial port via a RS485 converter (*model number:* 485SD9TB *sold separately*). This converter will convert the RS232 voltage signals to the compatible RS485 differential signals. Only one converter box is needed per serial port. Contact the factory or use the website www.anaheimautomation.com for RS485 converter information and sales.



Terminating Resistor

To eliminate noise on the transmission lines or when using a 4000 ft. or longer cable, a terminating resistor is suggested. If used, the termination resistor need only be added to the last (furthest from the converter box) PCL601 in the network between pins A(-) and B(+) on the RS485 Terminal Block. The value of this resistor should be 120 ohms.

Axis Selection

Each PCL601 is addressed using a programmable register allowing the PC to address up to 99 PCL601's from one port. The PCL601MOD's default axis address is "1". To change the axis, use the SMC60WIN software or the "~" command in Direct Talk Mode. To verify or check the axis, use the SMC60WIN software of the "%" command. The axis designation is nonvolatile and will remain the same until changed by the user.

Status LED

When powered and operated properly, the status LED will be green. When an error occurs, the LED will change to RED, and an error code will be generated in the error code register. To read and clear the error with the software, click on the "Verify Parameters" button located in the "Motion Tab". To read and clear the error while in "Direct Mode", use the error code "!" command. Once the error has been read and cleared, the LED will return to green and the error code register will be cleared to 0. Refer to the table on page 43 for a complete list of the error codes.

Interfacing With A Driver

The PCL601 controller was designed to control a step motor driver. For drivers with "*Opto-isolated*" inputs, the PCL601 needs to sink current through the driver's input LED. To do this, connect the +5VDC to the positive terminals of the inputs and connect the clock, direction and on/off outputs of the PCL601 to the negative terminals of the corresponding inputs on the driver. For a driver with "*TTL/CMOS"* inputs, the PCL601 has a negative going clock that will sink the current from the driver's inputs. Wire the clock, direction and on/off outputs and 0VDC reference of the PCL601 to the corresponding inputs of the driver respectively. For a PCL601 controller purchased with an integrated driver and power supply (Driver Pack), these connections are already internally made.

Technical Support

Everyone needs assistance on occasion. If you have problems using any of the equipment covered by this manual, please read the manual to see if it will answer your questions. Be sure to look in the Troubleshooting Section located near the back of this manual. If you need assistance beyond what this manual can provide, you may call the factory direct for application assistance. If possible, have this manual in hand. It is often helpful to have the controller connected to a computer with the software installed.

Electrical Specifications

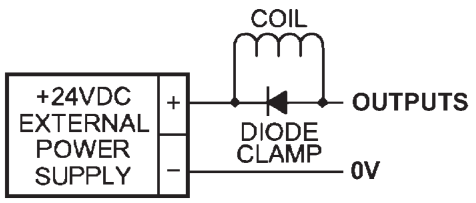
Power Requirements With No Connections:
5VDC @ 100mA or
8VDC to 24VDC @ 90mA

Operating Temperature:
0 to 60 degrees C

Pulse Output Range:
1 to 50,000 Hz
10uS negative going pulse width

Inputs (TTL-CMOS):
Logic “0”: 0 to 0.8VDC
Logic “1”: 3.5 to 24VDC
Analog input 1: 0 to 5VDC

Outputs (CLK,DIR,ON/OFF):
Open Drain Type
40VDC, 75mA



Baud Rate:
38400 Baud, Fixed

Data Format:

Modbus RTU:	Half-Duplex, no start bit, 8 data bits, even parity, no stop bit
AA ASCII:	Half-Duplex, 1 start bit, 8 data bits, no parity, 1 stop bit

Outputs (8 programmable):
Open Drain Type
40V, 100mA

Output1 active low time for output on the fly:
50uS

Encoder and Driver Output:
+5VDC Output, 50mA

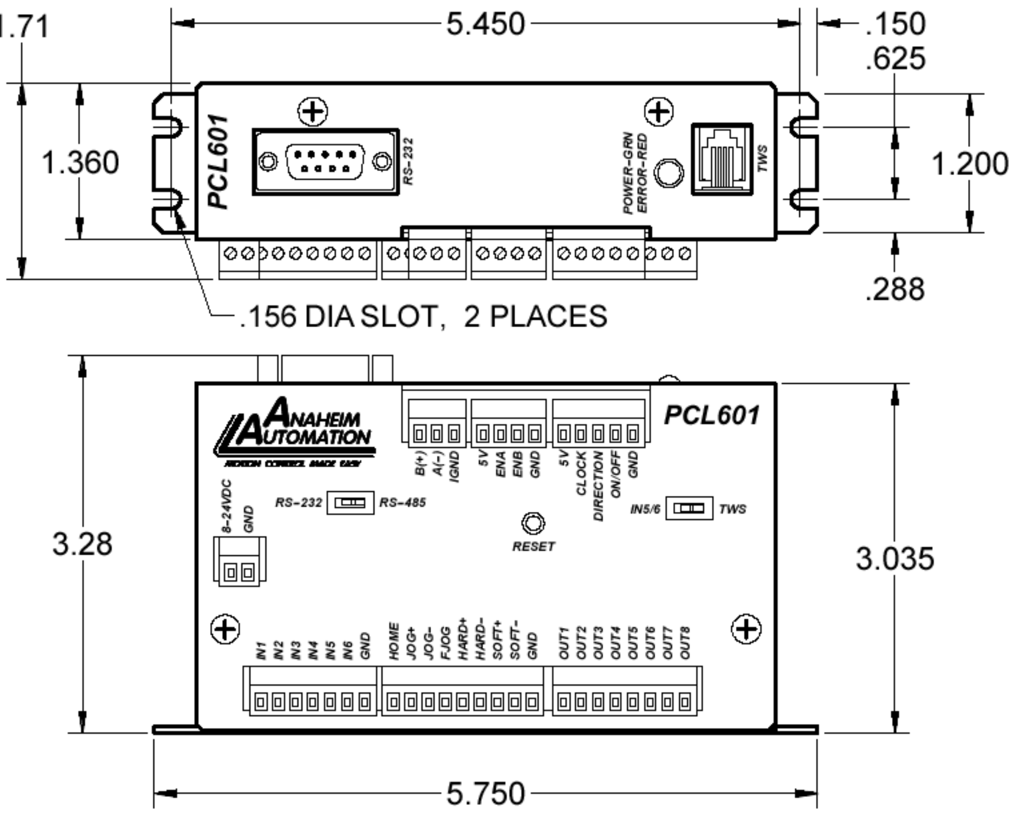
Note: For inductive loads, customers must connect a clamping diode to protect from flyback voltage spikes.

Ordering Information

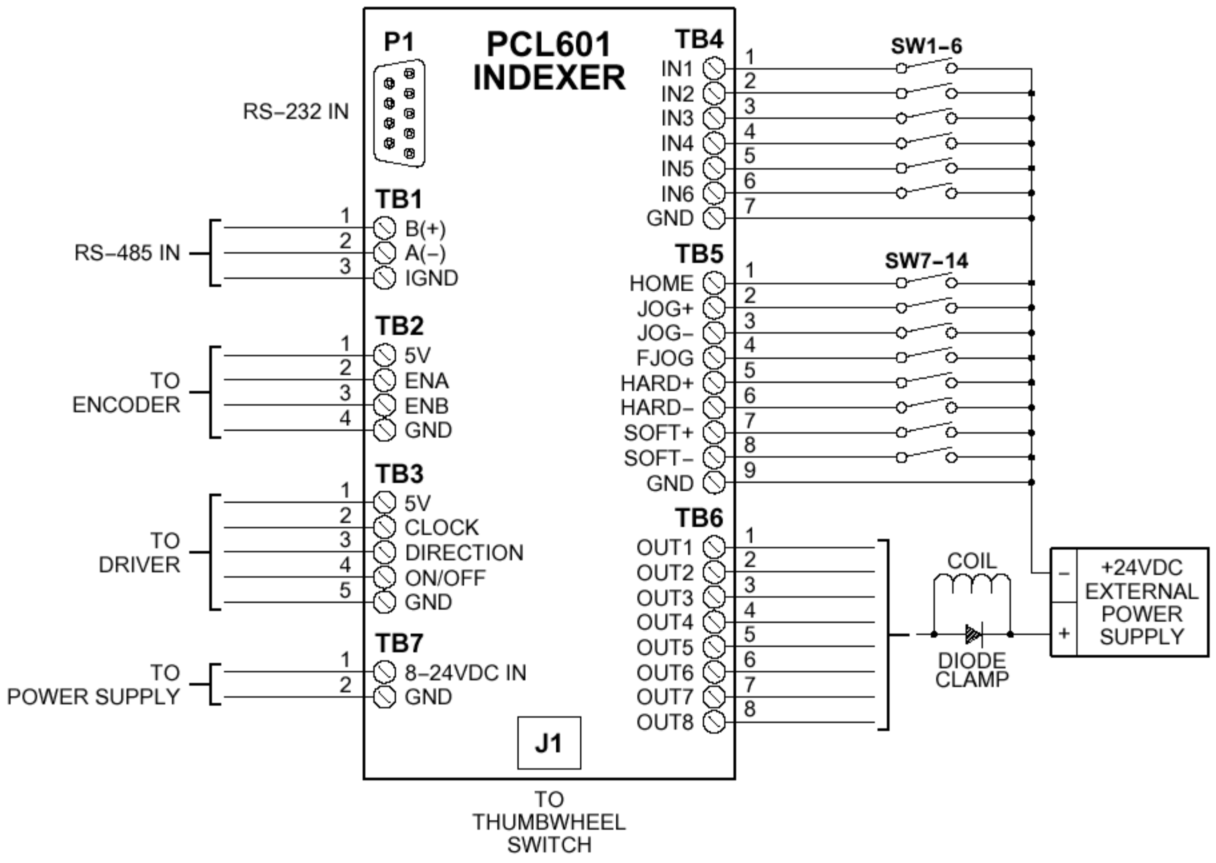
The table below lists a variety of products available from Anaheim Automation, Inc. These products include those covered by this manual, along with supporting cables and devices. We are continually adding new products to our line, so please consult Anaheim Automation, Inc. or its representatives for information on the latest releases.

Part Number	Description
PCL601MOD	Featured step motor controller with encoder feedback.
DPD75601	Controller/Drive Pack - Features a 7 amp unipolar drive and power supply.
DPN10601	Controller/Drive Pack - Features a 10 amp bipolar drive and power supply.
485SD9TB	RS232 to RS485 converter .
TWS7	Seven position thumbwheel switch compatible with any SMC60 series controller.
AA9MFC-6	6 foot straight through serial cable with one DB9 male and one DB9 female connector.
PSAM24V1.2A-5V3.5A	Power supply for PCL601. (24V@1.2A, 5V@3.5A)

Dimensions/Switch Locations



Wiring Diagrams



Terminal Descriptions

Position	Description - Power
1	8-24VDC Power Input
2	Ground Power Return

Position	Description - Inputs
1	Input 1 - Analog input
2	Input 2 - Index on the fly input
3	Input 3
4	Input 4
5	Input 5 - SW2 in position IN5/6
6	Input 6 - SW2 in position IN5/6
7	Ground

Position	Description - RS485
1	A(-)
2	B(+)
3	IGND - This is an isolated ground for RS485 only

Position	Description - Outputs
1	Output 1 - Output on the fly output
2	Output 2
3	Output 3
4	Output 4
5	Output 5
6	Output 6
7	Output 7
8	Output 8 - Encoder retries error output

Position	Description - Encoder
1	+5VDC supply for encoder
2	A channel for encoder
3	B channel for encoder
4	Ground return for encoder

Position	Description - Limit Switch Inputs
1	Home Limit
2	Jog +
3	Jog -
4	Fast Jog
5	Hard Limit +
6	Hard Limit -
7	Soft Limit +
8	Soft Limit -
9	Ground

Position	Description - Driver Outputs
1	+5VDC Supply for opto-isolated driver inputs
2	Clock output
3	Direction output
4	Motor current on/off output
5	Ground reference for non opto-isolated driver inputs

Connector Descriptions

Switch Number	Description
P1	This connector is for the RS-232 communication and is labeled RS-232.
J1	This connector is for the thumbwheel module and is labeled TWS.

Slide Switch Descriptions

Switch Number	Description
SW1	This switch is used to select either RS232 or RS485.
SW2	This switch is used to select either the thumbwheel or inputs 5 and 6.

Section 2: Functions

Move Number of Steps: This command causes the motion to start in the direction last specified. This command will move the motor the number of steps given. (Range: 1 to 8388607)

Move to Position: The move to position command specifies the next absolute position to go to. The PCL601 controller automatically sets the direction and number of steps needed to go to that position. (Range: -8388607 to +8388607)

Slew: The slew command will accelerate the motor up to maximum speed and continue to run at that speed until reaching a registration mark, hard limit switch, soft limit switch, receiving a "." (stop hard) or "," (stop soft) command.

Set Position: The set position command sets the position register to a designated value. The number will be the new absolute position of the motor. The default value is 0. (Range: -8388607 to +8388607)

Limit Switch Inputs: The limit switch inputs are internally pulled up by a resistor making them normally +5 volts. To activate the input, the pin must be grounded to (0VDC). All limit switch inputs are internally clamped to +5V, thus allowing voltages of upto +24VDC to be used.

Hard Limit Inputs: When a hard limit switch is encountered, the motion will stop immediately. The position counter will also cease counting. *Hard limits are intended as an emergency stop for your system. It should not be used to do any positioning type functions*. These limits are directional.

Soft Limit Inputs: These switches should be used exclusively for homing. Once positioned properly with the appropriate parameters, it causes the motor to ramp down to the base speed before encountering the home limit switch. However, the soft limit switch will work for any type of motion command. These limits are directional.
NOTE: Whenever a soft limit switch is activated, the motor will decelerate and run at base speed during an indexing move, or stop during a slewing move. Be sure to come back past the soft limit switch to set any origins, otherwise the motor will decelerate as it goes past the soft limit switch during normal operation.

Home Limit Input: This switch is used to establish a position designated "home" or datum position using the following: home to *soft and home limit*, or home to *home limit*. This limit is not directional.

Home to Soft, Home Limit (2 Switch Operation): This type of homing routine requires two grounding type limit switches called home and soft. The first limit switch seen is the soft limit. This will decelerate the motor down to base speed. The motor will then continue to run at base speed until it contacts the home limit switch input causing the motor to stop. The home limit switch activates as a hard limit if a soft limit is not sensed. The soft limit is directional, meaning that it will work in only one direction as specified. The soft limit switch will work for any type of motion command. The home limit switch will work only for the two home motion commands.
NOTE: There should be sufficient distance between the two limit switches, as to let the motor reach base speed.

Home to Home Limit (1 Switch Operation): This type of homing differs in that only one limit switch is needed. In this homing routine the motor moves toward the home limit switch. When the home limit switch is contacted the motor will ramp down to base speed, reverse direction and continue at base speed until the limit switch is released. This is a good way to compensate for any backlash in a system. It is also useful for minimizing the number of limit switches needed for homing.
NOTE: The home switch needs to be low during the entire deceleration and reversing time.

Jog Inputs: The jog switch inputs are internally pulled up by a resistor making them normally +5 volts. To activate the input, the pin must be grounded to (0VDC). All jog switch inputs are internally clamped to +5V, thus allowing voltages of upto +24VDC to be used. Jog is a manual function. The user can select the direction and speed (fast or slow) by grounding the appropriate combinations of inputs. To jog a motor, it is necessary to ground the jog input for the direction desired. For fast jog, both the fast input and jog input for the appropriate direction must be low at the same time. By grounding one of the jog inputs, the user causes the motor to run at base speed. When the fast input is grounded, the motor will then accelerate to the programmed jog speed. The position register will keep track of the number of steps that are taken during jogging. Once a +jog or a -jog function has been performed, the direction register will retain the last direction of movement; that is, a subsequent go command will be in the same direction as the last jog command.

Inputs: All inputs (except input 1) are internally pulled up by a resistor making them normally +5 volts. To activate the input, the pin must be grounded to (0VDC). All inputs are internally clamped to +5V, thus allowing voltages of upto +24VDC to be used. Six inputs are provided per axis. The inputs are TTL, CMOS and 24V compatible. The inputs may be used to initiate a machine cycle, for inter-axis coordination (in stored program mode), for operator intervention, for sensing a machine condition such as out of stock or wait for temperature to be reached, etc. A grounded input will read a "0" and an open or high input will read as a "1". Input 1 is a special input that is capable of reading an analog voltage between 0 and +5VDC. Since this input does not have a pull-up resistor, biasing of this input is needed if it is not used as an analog input. Inputs 5 and 6 are used together with the thumbwheel switch. To use inputs 5 and 6, SW2 must be in the IN5/6 position. If SW2 is in the TWS position, then these two inputs are not connected to the processor.

Analog Input: Input 1 can be configured to read an analog voltage to either set the absolute position of the motor or to set the maximum speed of the motor.
To set the position, when told via the *goto analog position* command, the input will read a voltage between 0 and +5VDC and based on the" upper and lower" limits of the function, a move will occur to a calculated position between the two limits. The motor must finish the move before it can be told to read the input again for the next position. For example, if the lower limit is set to 0 and the upper limit is set to 5000 and the analog position is set at +2.0VDC, then the motor will move to position 2000. Changing the lower limit to 1000 and the voltage to +3.2VDC, the motor will move to position 3560. See examples below for calculations of the analog inputs. (Range of limits: 0 to 65535 and the lower limit < upper limit)
To set the max speed, when told via the *set analog speed* command the input will read a voltage between 0 and +5VDC, and based on the "upper and lower" limits of the function, a max speed can be obtained based on a calculated frequency between the two points. The speed however can not be changed when the DPY50601 is busy (moving). See examples below for calculations of the analog inputs. (Range of limits: 0 to 50000 and the lower limit < upper limit)

Analog calculations.	**Example1:**	**Example2:**
(Upper-Lower) * (Voltage/5) = X	(5000 - 0) * (2 / 5) = 2000	(5000 - 1000) * (3.2 / 5) = 2560
Lower + X = Position or Frequency	0 + 2000 = 2000	1000 + 2560 = 3560

Outputs: Eight outputs are provided per axis. Outputs may be used to operate relays, coolant valves, air cylinders, or, with the correct interfacing, any electronically controlled device. The outputs can drive all types of common peripheral power loads, including lamps, relays, solenoids, LED's, printer heads, and heaters. For inductive loads, it will be necessary to connect a clamping diode (refer to specification section) from the output to the power source in order to provide adequate fly-back protection. The outputs are current sinking, open collector darlingtons. They are capable of sinking up to 100mA per output with voltages up to 40VDC. Turning an output on will pull the output pin to ground and turning an output off will make the output pin open. Output 1 has a special function *(output on the fly*) that will enable it to be triggered at a certain absolute position during a move. Output 8 has a special function that will trigger when the encoder retries function fails.

Output on the fly: This special function enables output 1 to turn on during a relative index or absolute move. There are three critical portions of information needed to make this function work correctly. First, output 1 will turn on (0VDC) for a preset delay of 50uS at a specific absolute position set by the *1st output position* command. Second, the output can then repeat this after a preset amount of steps set by the *number of steps between outputs* command, and third a predetermined amount of times to set the output is required by the *number of outputs* command which determines the preset amount of times to trigger the output. So if you start at position 0 and want to move to an absolute position of 10,000, you can set output 1 to turn on at position 2000, and every 1000 steps after that 5 times. So at position 2000, 3000, 4000, 5000, and 6000 output 1 will turn on for 50uS. To only have the output turn on at one position set both the "*number of steps between outputs*" and the "*number of outputs*" commands to 0. This function must be enabled, and will only work during a relative index or absolute position move. The output will trigger while going in either direction. If you do not want the output to trigger in the negative direction, the function must be turned off before the index move is started.

Index on the Fly: This special function uses Input 2 when a motor is slewing to move a predetermined amount of steps, set with the registration index command, before stopping. This function must be enabled, and will only work during a slew move. The registration index must be set before movement begins. (Range: 1 to 8388607)

End of Program: The end of program command, used within a stored program, stops execution of the program. This command must be used at the end of all programs.

Wait: In stored program mode, the wait command pauses the program for the specified number of milliseconds. (Range: 1 to 65535)

If/Then Statements: The if/then statements are conditional based on the values preset in the program. The user can either test each individual input or all inputs at once. If the input or input register matches the given value or values, then the program will execute the next line. If the input or input register does not match the given value, the program will skip the next line and execute the following line. An open input is read as a 1, and a grounded input is read as a 0.

Branching or Goto Statements: The goto instruction will have the program jump to the given label. If no label is in the program, it will error when trying to send.

Return from Subroutine: This function can be placed anywhere in the program as long as a goto statement has been already executed. The program will jump back to the last goto statement encountered and execute the next line in the program.

Inner and Outer Loop: The loop instructions allow the user to loop a program a variable number of times. The program will loop to the designated label location of the program. However , the label must always be at a lower line number than the loop instruction itself. You can only nest inner loops inside an outer loop. You may not nest an inner loop inside an inner loop, or an outer loop inside an outer loop. Multiple nested inner loops are allowed in one outer loop.

Finish Move: When writing a program, the finish move command is used directly after every motion command. When using this command, the PCL601 internally generates a busy signal and will wait until the move is complete before executing any further commands. Unless the finish move command is used, the PCL601 will continue to execute the program. If it encounters a command that cannot be used when the motor is moving, the PCL601 will error and stop the program prematurely.

Repeat Last Move: This command will move the motor the number of steps given in the last indexing move. This command will not work correctly if the encoder autocorrect function is enabled.

Encoder Commands: The PCL601 controller is capable of using a quadrature incremental encoder with A and B channels.

Encoder Auto Correct: This command will enable or disable the encoder feature of the PCL601. When enabled, the encoder function will compare the desired position with the actual encoder position. If it is not in the correct position a correction move will be made.

Encoder Delay: This sets the wait time, which is a specified number of milliseconds after a relative index or absolute move is finished, prior to reading the encoder. This is used to remove the ringing that might be associated with the mechanics of the system. (Range: 0 to 65535)

Encoder Motor Ratio: This represents the ratio for the number of encoder pulses to one motor step. This ratio must be a whole number. For example, given a 1000 line quadrature encoder and a 400 step/revolution motor, the motor ratio is (1000 * 4) / 400 = 10 (Range: 1 to 255 and must be a whole number)

Encoder Retries: This is the number of times the PCL601 will try to autocorrect the motor shaft position before producing an error. When the error is produced, Output 8 is triggered. (Range: 0 to 255)

Encoder Window: This is the allowable error in encoder pulses (either plus or minus) from the desired position that is allowed before the motor autocorrects. (Range: 0 to 255)

Thumbwheel Index: This special function allows a thumbwheel with up to 7 decades to be used with the PCL601 to set a relative index. To use the thumbwheel, SW2 must be in the TWS position or the thumbwheel will be disabled.

Acceleration/Deceleration: The acceleration and deceleration are the same value. The acceleration is entered directly as steps/sec^2 and controls the time that the motor will take to move from base speed to max speed, and from max speed to base speed. The higher the value, the faster the motor will accelerate. The same principal applies for the deceleration which is controlling the time it takes to go from the max speed to base speed. (Range: 100 to 9,999,999)

Base Speed: The base speed is the speed at which motion starts and stops. It is entered directly as the number of steps per second. This speed must always be less than the max speed and jog speed. (Range: 1 to 5000)

Max Speed: The max speed is the top speed the user wants the motor to run. This speed must always be greater than the base speed. It is entered directly as the number of steps per second.
(Range: 1 to 50000)

Jog Speed: The jog speed sets the fast jog rate. Jog (+/-) is used to run at base speed. The FJOG pin, when grounded, will ramp the motor to the set jog speed. This speed must always be greater than the base speed. It is entered directly as the number of steps per second. (Range: 1 to 50000)

Motor Current: This command will control the on/off output which is designed to connect to the on/off input of Anaheim Automation's step motor drivers. To energize and allow current to flow through the coil of the motor, set the value to on. To de-energize and turn the current off to the motor, set the value to off. This is a dedicated output and not controlled with the output register.

Verify: The verify command causes the PCL601 controller to send data back to the PC or PLC. The data is sent as an ASCII decimal string followed by a carriage return and a line feed. The verify commands are shown in the table on page 38.

Section 3: SMC60WIN Software

The SMC60WIN software is a handy utility that supports Anaheim Automation's line of PCL601's step motor controllers. Connecting your PC to the PCL601, via a serial cable, the SMC60WIN software can easily perform the following tasks:

- Exercise and monitor the PCL601 controller

- Write and edit stored programs for standalone operation

- Directly communicate with the PCL601 controller

Note: The SMC60WIN Software communicates with the PCL601 using the AA ASCII protocol. The device is set by default to use the Modbus RTU protocol, so, in order to interface it with the software, the device must be switched to use the AA ASCII protocol; This can be achieved in a couple of ways: the user can use an HMI to write the command code 11 to the command register of the device, or a terminal program can be used to send the command below. See Appendix 4 for more details.

Broadcast/ Address	Write Function	Command Register Address		Command Code: Switch to AA ASCII		CRC Low Byte	CRC High Byte
00h	06h	00h	00h	00h	0Bh	C8h	0Dh

Installation

Software

- The SMC60WIN software can be downloaded at anaheimautomation.com, and it contains the setup program and the SMC60WIN software, PCL601 manual and sample programs.
- SMC60WIN is compatible with all versions of Windows including Windows XP, Windows Vista, and Windows 7.

WindowsXP/Vista/7 Installation

1) Download software from the Anaheim Automation website.
2) Unzip the file.
3) Double click on the **setup** application file.

Getting Started

1) Double click on the SMC60WIN icon to run the SMC60WIN software.

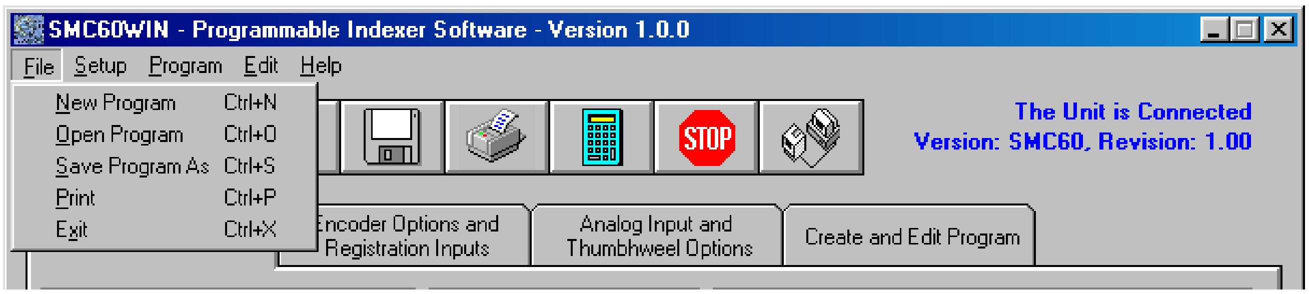
2) Apply power to the PCL601 controller.

3) Set the appropriate com port setting by selecting Setup | Com Port Settings from the menu bar. (Ctrl+M is a shortcut)

4) Set the appropriate axis setting by selecting Setup | Axis from the menu bar. (Ctrl+A is a shortcut)

5) Establish communications with the PCL601 by clicking on the Connect Icon, or select Setup | Connect from the menu bar. If the unit is connected properly, the program will notify you when communications has been established. (Ctrl+C is a shortcut)

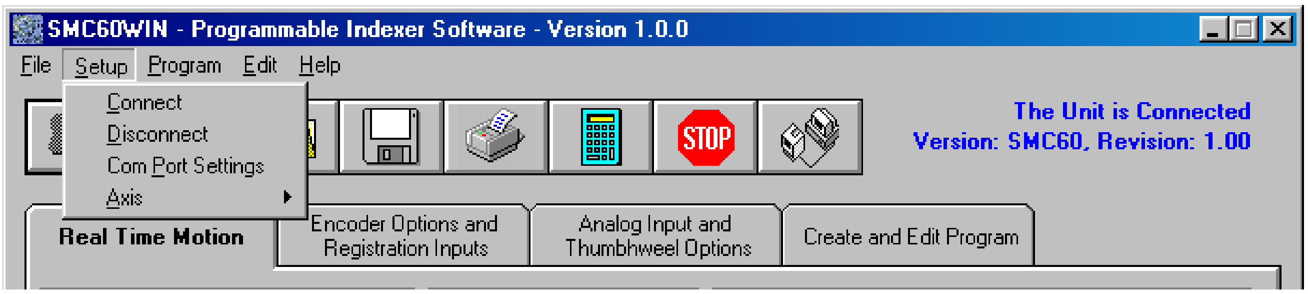
File Menu



The Unit is Connected
Version: SMC60, Revision: 1.00

New Program	Start editing a new program.
Open Program	Open an existing program from disk.
Save Program As	Save the current program to disk.
Print...	Print the current program.
Exit	Exit the SMC60WIN software.

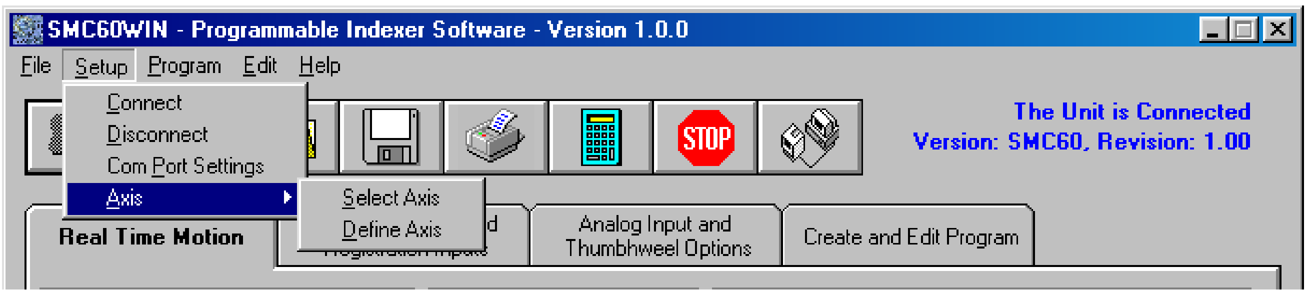
Setup Menu


SMC60WIN - Programmable Indexer Software - Version 1.0.0
File
Setup
Program
Edit
Help
Connect
Disconnect
Com Port Settings
Axis
STOP
Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbhweel Options
Create and Edit Program

The Unit is Connected
Version: SMC60, Revision: 1.00

Connect	Establish communications with the controller.
Disconnect	Release the COM port for other devices to use.
Com Port Settings	Select COM port.
Axis	Set axis selection and stored axis of the controller.

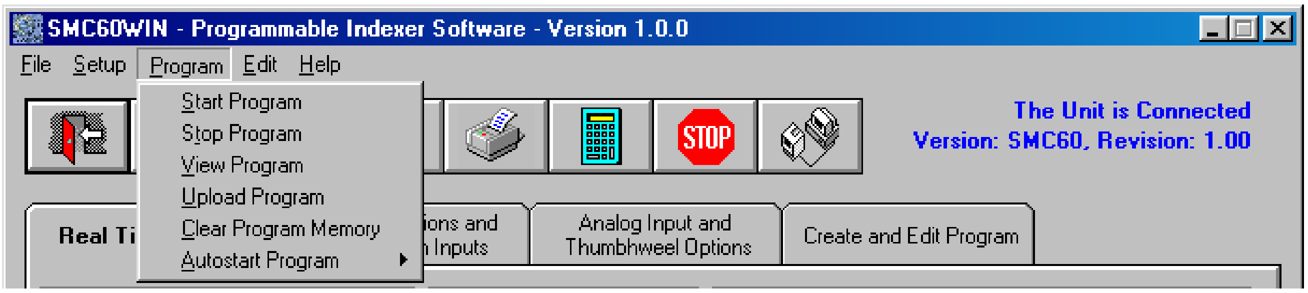
Setup - Axis Menu



The Unit is Connected
Version: SMC60, Revision: 1.00

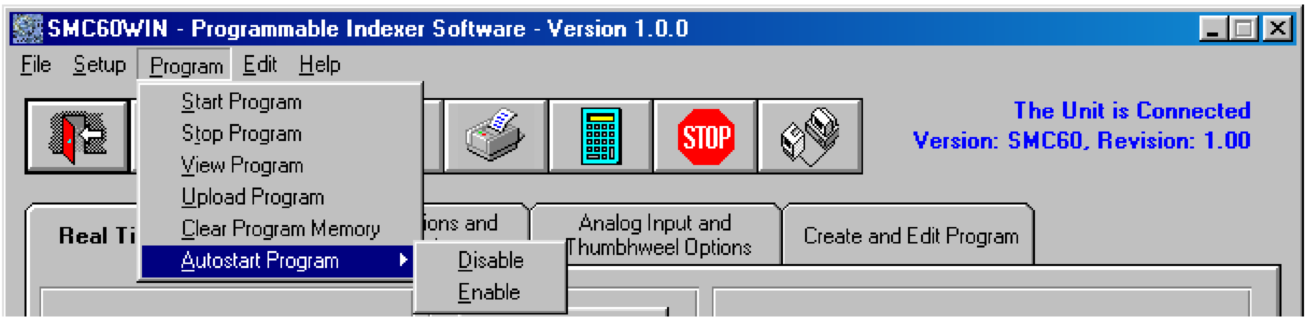
Select Axis	Sets the axis select parameter in the SMC60WIN software. (1-99)
Define Axis	Sets the programmable address in the controller. (1-99)

Program Menu



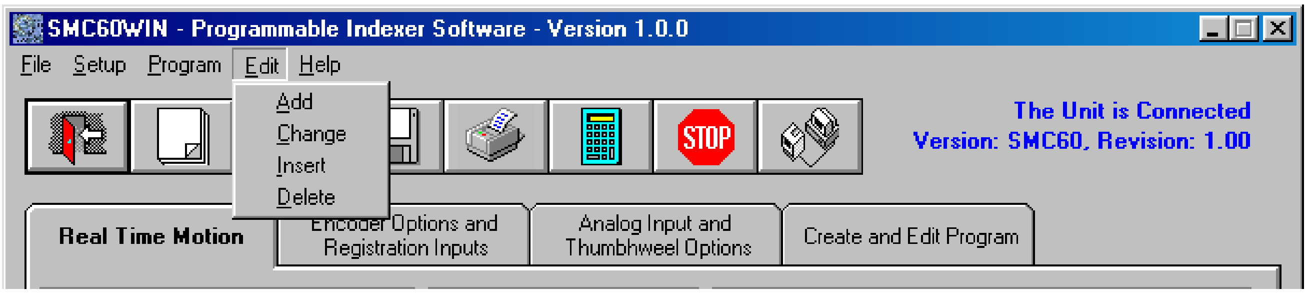
Start Program	Start the execution of the program in the controller memory.
Stop Program	Stop the execution of the program in the controller memory.
View Program	View the program stored in the controller memory.
Clear Program Memory	Clear the program memory in the controller.
Autostart Program	Turn the autostart function on or off.

Program - Autostart Program Menu



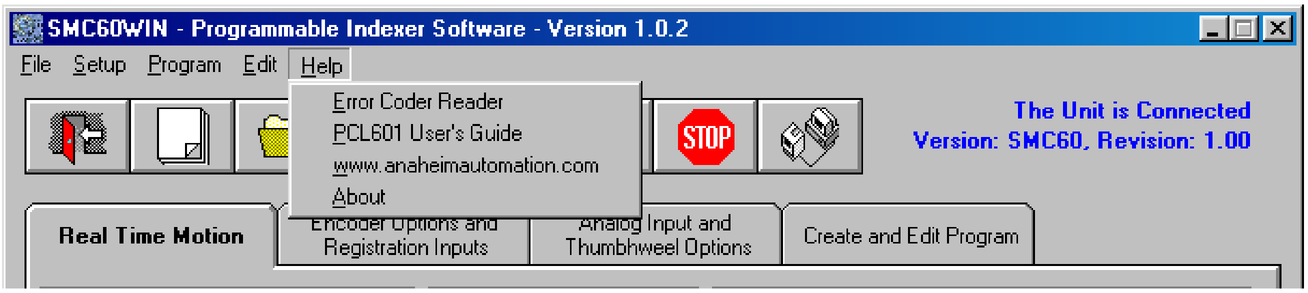
Disable	Program will disable the execution of a stored program at power up.
Enable	Program will start execution when controller is powered up.

Edit Menu


SMC60WIN - Programmable Indexer Software - Version 1.0.0
File Setup Program Edit Help
Add
Change
Insert
Delete
STOP
The Unit is Connected
Version: SMC60, Revision: 1.00
Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbhweel Options
Create and Edit Program

Add	Adds a new line of code to the end of the progam.
Change	Edits the currently selected line of code.
Insert	Insert a new line of code before the currently selected line of code.
Delete	Deletes the currently selected line of code.

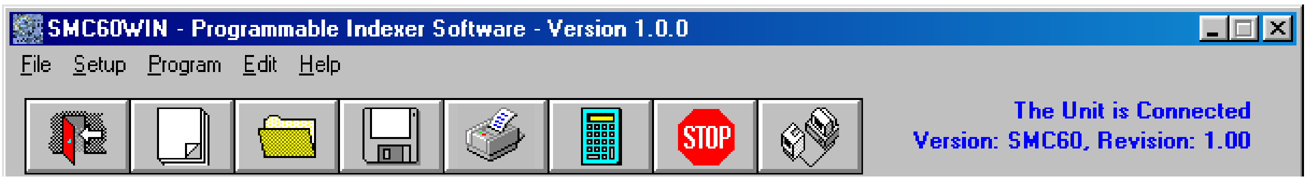
Help Menu

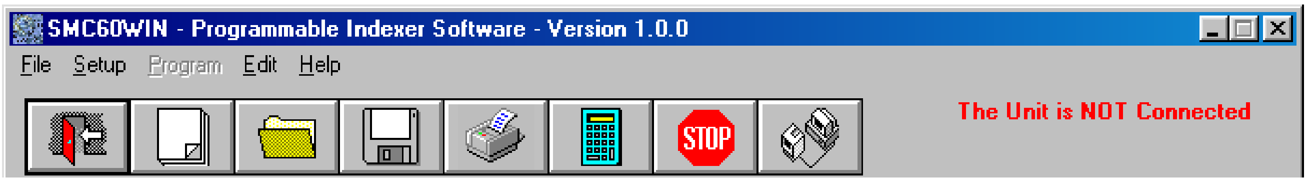

SMC60WIN - Programmable Indexer Software - Version 1.0.2
File Setup Program Edit Help
Error Coder Reader
PCL601 User's Guide
www.anaheimautomation.com
About
STOP
The Unit is Connected
Version: SMC60, Revision: 1.00
Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbhweel Options
Create and Edit Program

Error Coder Reader	Utility to read the error code generated by the PCL601 controller.
PCL601 User's Guide	Opens up the User's Guide in .pdf format.
www.anaheimautomation.com	Opens up the Anaheim Automation Website.
About	Displays the version of the SMC60WIN and contact information.

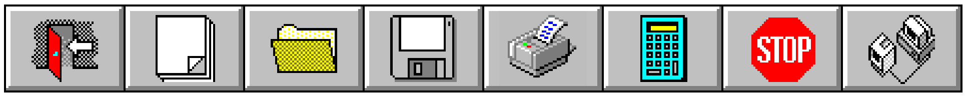
"The Unit is Connected" / "The Unit is NOT Connected"

On the right of the Toolbar, the user will find the communication status of the PCL601 controller. If communications are not established, please refer to the Troubleshooting Section.


SMC60WIN - Programmable Indexer Software - Version 1.0.0
File Setup Program Edit Help
STOP
The Unit is Connected
Version: SMC60, Revision: 1.00

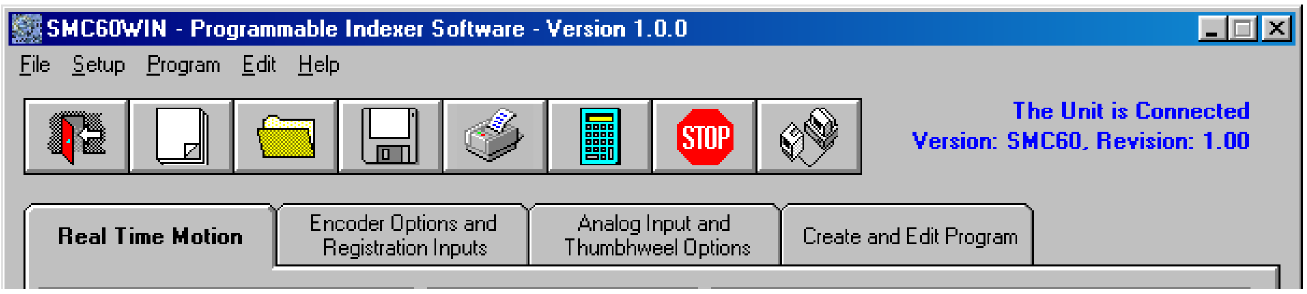

SMC60WIN - Programmable Indexer Software - Version 1.0.0
File Setup Program Edit Help
STOP
The Unit is NOT Connected

Toolbar



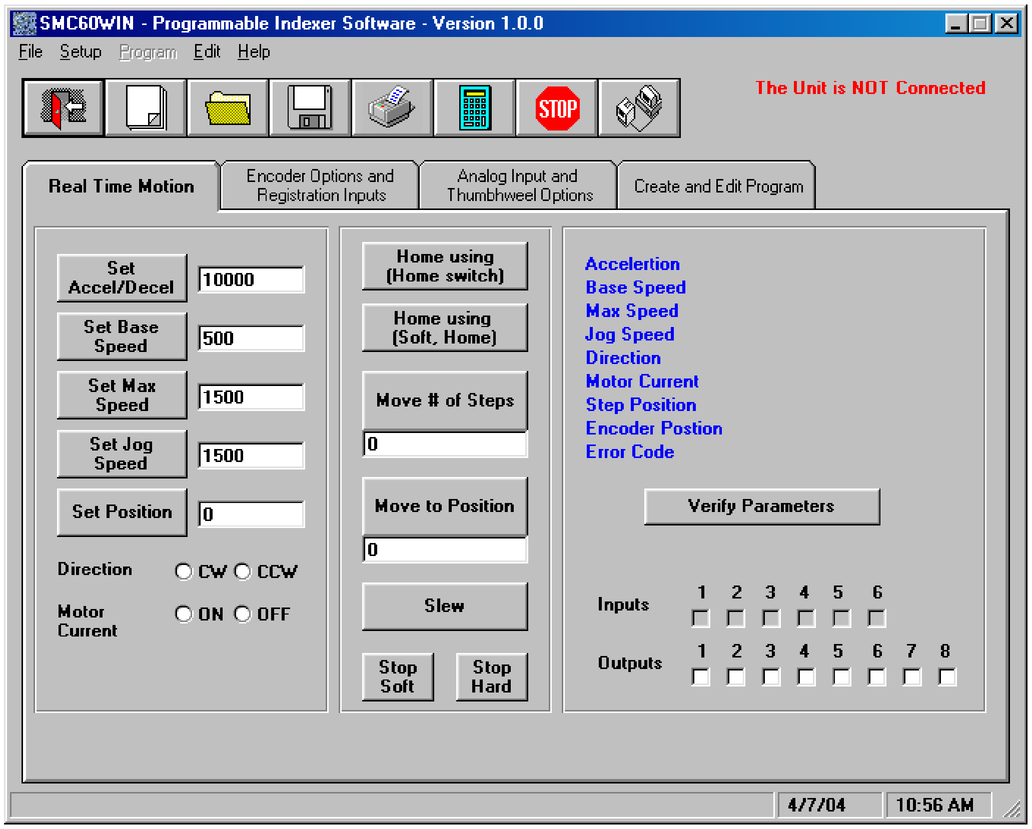
Exit	Exit the SMC60WIN software.
New	Start editing a new program.
Open	Open an existing program from disk or directory.
Save	Save the current program to disk or directory.
Print	Print the current program.
Calculator	Open the desktop calculator.
Stop All	Stop the program and all motion from running.
Connect	Establish communication with the controller.

Tab Sheets


SMC60WIN - Programmable Indexer Software - Version 1.0.0
File Setup Program Edit Help
The Unit is Connected
Version: SMC60, Revision: 1.00
Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbhweel Options
Create and Edit Program

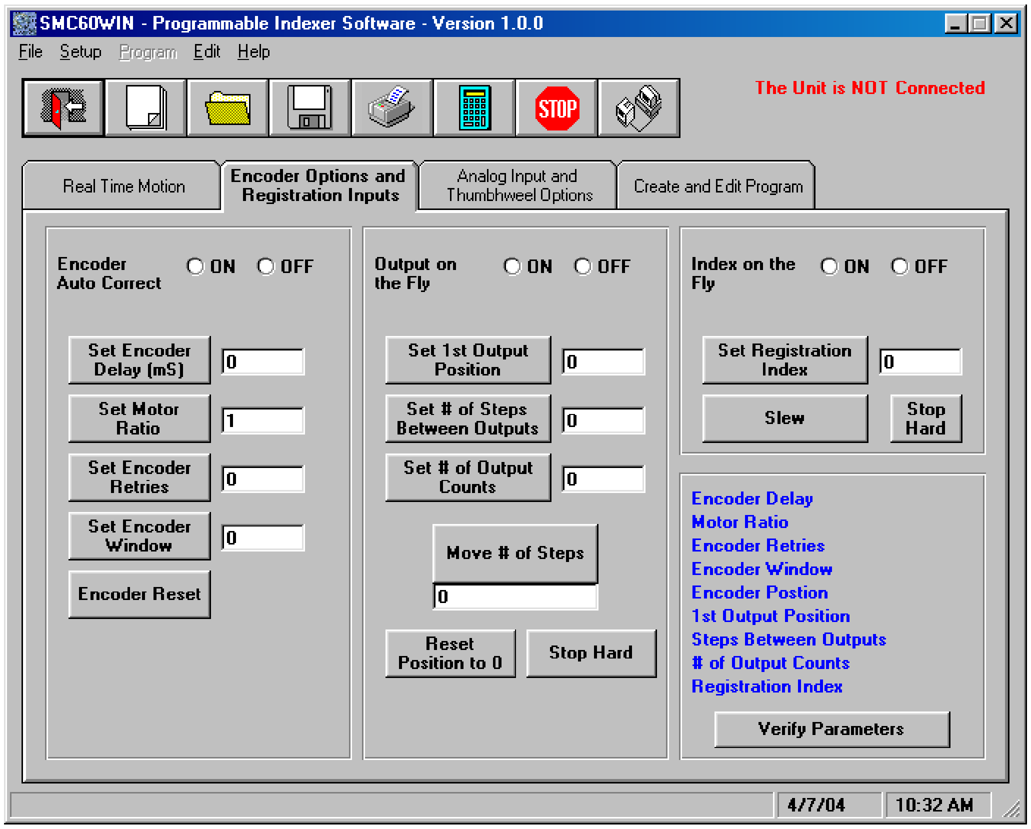
Real Time Motion	Monitor and control motion of the controller.
Encoder Options and Regis-tration Inputs	Monitor and change settings for encoder options, input on the fly and output on the fly.
Analog Input and Thumb-wheel Options	Monitor and change settings to Analog Inputs and the thumbwheel switches.
Create and Edit Programs	Write and edit PCL601 stored programs.

Tab Sheets - Real Time Motion



Set Accel/Decel	Send acceleration & deceleration parameter to controller. (step/sec^2)
Set Base Speed	Send base speed parameter to the controller. (step/sec)
Set Max Speed	Send maximum speed parameter to the controller. (step/sec)
Set Jog Speed	Send fast jog speed parameter to the controller. (step/sec)
Set Position	Set motor position.
Set Direction	Set direction to clockwise or counter-clockwise.
Set Motor Current	Set the current in the motor on or off.
Home using (Home Switch)	Motor will seek the home position by moving towards home switch which will stop the motor, reverse the motor direction and stop when the home limit switch is no longer triggered. (One switch is required to stop anti-backlash)
Home using (Soft and Home Switches)	Motor will seek the home position by moving towards home switch but motor will slow down to base speed when the soft switch is triggered, followed by triggering the home switch to stop motion. (Two switches are required to stop)
Move number of steps	Motor will move number of steps entered.
Move to Position	Motor will move to specified position.
Slew	Motor will ramp up to maximum speed and keep moving until stop motion is triggered.
Stop Soft	Ramp motor down to base speed and stop.
Stop Hard	Stop any motor motion immediately.
Inputs	View inputs. (checked = On, blank = Off)
Outputs	View and trigger outputs. (checked = ON, blank = OFF)
Verify Parameters	Updates and displays controllers parameters sheet and resets the error codes.

Tab Sheets - Encoder Options and Registration Inputs


SMC60WIN - Programmable Indexer Software - Version 1.0.0

File Setup Program Edit Help

The Unit is NOT Connected

Real Time Motion | Encoder Options and Registration Inputs | Analog Input and Thumbwheel Options | Create and Edit Program

Encoder Auto Correct ○ ON ○ OFF

Set Encoder Delay (mS): 0

Set Motor Ratio: 1

Set Encoder Retries: 0

Set Encoder Window: 0

Encoder Reset

Output on the Fly ○ ON ○ OFF

Set 1st Output Position: 0

Set # of Steps Between Outputs: 0

Set # of Output Counts: 0

Move # of Steps: 0

Reset Position to 0 | Stop Hard

Index on the Fly ○ ON ○ OFF

Set Registration Index: 0

Slew | Stop Hard

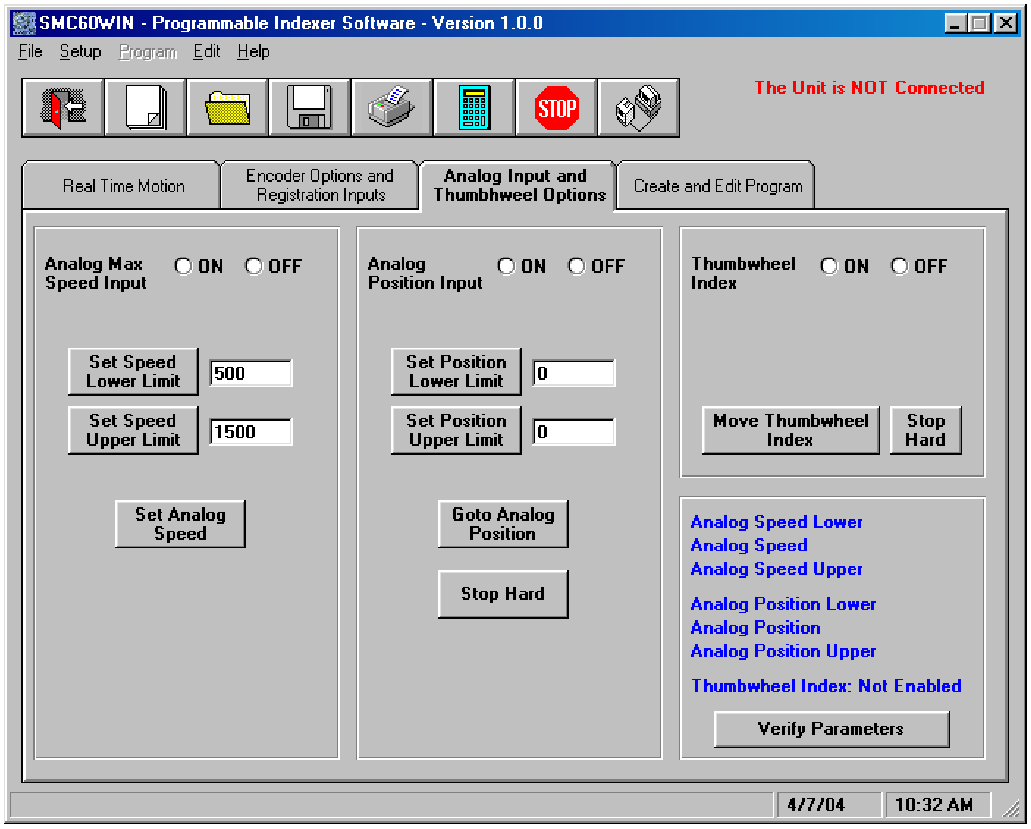
Encoder Delay
Motor Ratio
Encoder Retries
Encoder Window
Encoder Postion
1st Output Position
Steps Between Outputs
# of Output Counts
Registration Index

Verify Parameters

4/7/04 10:32 AM

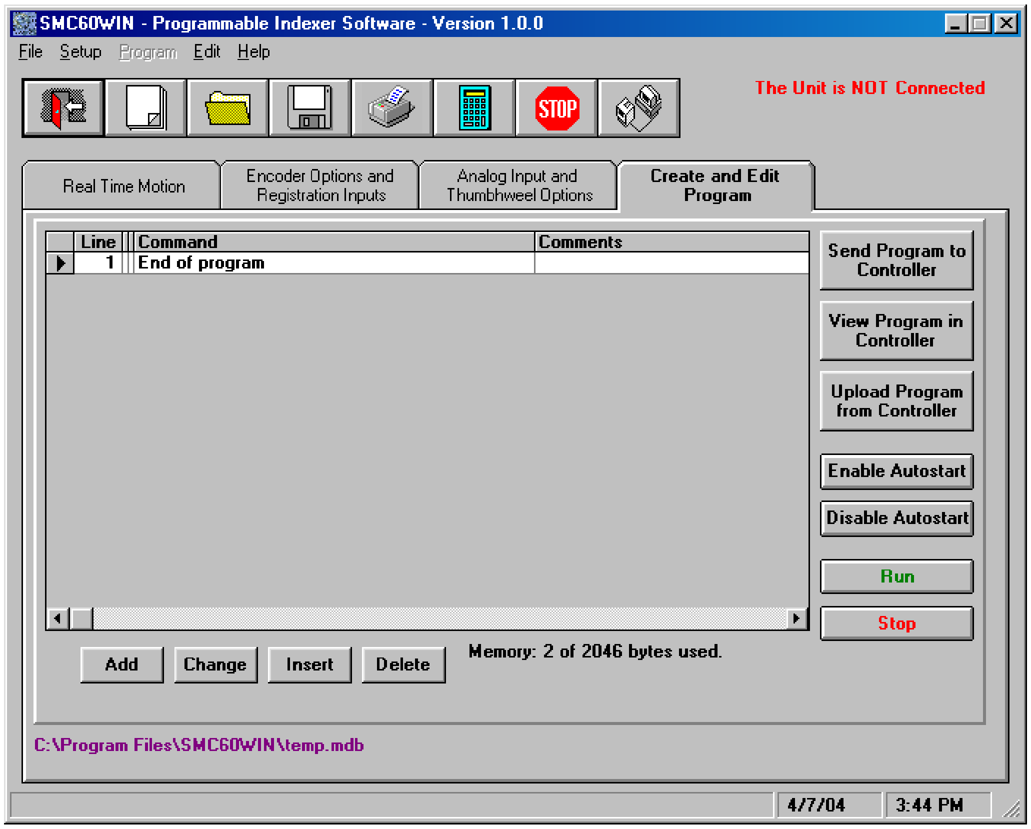
Encoder Auto Correct	Set the encoder autocorrect feature on or off.
Set Encoder Delay	Send the encoder delay parameter to the controller. (ms)
Set Motor Ratio	Send the encoder pulse to motor step ratio to the controller.
Set Encoder Retries	Send the number of encoder autocorrect retries to the controller. When the autocorrect errors, Output 8 will be triggered.
Set Encoder Window	Send the encoder window to the controller.
Encoder Reset	Reset the encoder count to 0 in the controller.
Output on the Fly	Set the output on the fly feature on or off.
Set 1st Output Position	Send the 1st position to set the output to the controller.
Set # of Steps Between Outputs	Send # of steps to take between activated outputs to the controller.
Set # of Output Counts	Send # of output counts to the controller.
Move # of Steps	Motor will move number of steps entered.
Reset Position to 0	Reset the controller position to 0 (zero).
Stop Hard	Stop any motor motion immediately.
Index on the Fly	Set the index on the fly feature on or off.
Set Registration Index	Send registration index to controller.
Slew	Motor will ramp up to maximum speed and keep moving until the index on the fly input is activated or a stop motion is triggered.
Stop Hard	Stop any motor motion immediately.
Verify Parameters	Updates and displays controllers parameters for this tab sheet.

Tab Sheets - Analog Input and Thumbwheel Options



Analog Max Speed Input	Sets the analog speed input feature on or off.
Set Speed Lower Limit	Send the analog speed lower limit to the controller.
Set Speed Upper Limit	Send the analog speed upper limit to the controller.
Set Analog Speed	Sets the max speed based on analog voltage measured at input.
Analog Position Speed Input	Sets the analog position input feature on or off.
Set Position Lower Limit	Send the analog position lower limit to the controller.
Set Position Upper Limit	Send the analog position upper limit to the controller.
Set Analog Position	Motor will move to position based on analog voltage measured at input.
Stop Hard	Stop any motor motion immediately.
Thumbwheel Index	Sets the thumbwheel index feature on or off.
Move Thumbwheel index	Motor will move number of steps set by the thumbwheel switches.
Stop Hard	Stop any motor motion immediately.
Verify Parameters	Updates and displays controllers parameters for this tab sheet.

Tab Sheets - Create and Edit Program


SMC60WIN - Programmable Indexer Software - Version 1.0.0
File
Setup
Program
Edit
Help
STOP
The Unit is NOT Connected
Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbwheel Options
Create and Edit Program
Send Program to Controller
View Program in Controller
Upload Program from Controller
Enable Autostart
Disable Autostart
Run
Stop
Add
Change
Insert
Delete
4/7/04
3:44 PM

Line	Command	Comments
1	End of program	

Memory: 2 of 2046 bytes used.

C:\Program Files\SMC60WIN\temp.mdb

Send Program to Controller	Send current program to the controller.
View Program in Controller	View program in the controller memory.
Upload Program in Controller	Upload the program in the controller for editing and saving.
Enable Autostart	Program will start when controller is powered up.
Disable Autostart	Program will only execute when run is clicked.
Run	Execute the program in the controller memory.
Stop	Abort program execution.
Add	Adds a new line of code to the end of the program.
Change	Edits the currently selected line of code.
Insert	Insert a new line of code before the currently selected line of code.
Delete	Deletes the currently selected line of code.

PCL601 Memory Available

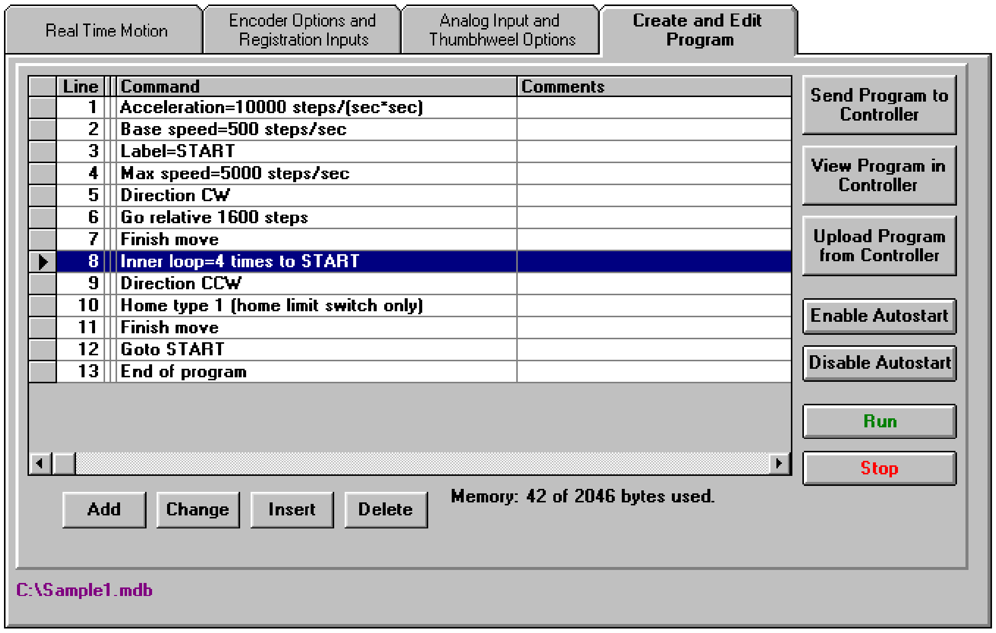
With the create and edit program tab sheet selected, the user can obtain the amount of available memory, located to the right of the Delete command button. The PCL601 has a maximum available memory of 2046 bytes - each instruction can use from 2 to 7 bytes.

Current Program Filename

With the create and edit program tab sheet selected, the user can obtain the current program filename, located in the lower left corner of the SMC60WIN window. All programs created by the SMC60WIN software will have a .mdb and a .bak extension.

Currently Selected Line

The currently selected line is indicated in the program by the right pointing arrow/triangle in the left column. Clicking on any line will select a new currently selected line.


Real Time Motion
Encoder Options and Registration Inputs
Analog Input and Thumbwheel Options
Create and Edit Program
Send Program to Controller
View Program in Controller
Upload Program from Controller
Enable Autostart
Disable Autostart
Run
Stop
Add
Change
Insert
Delete

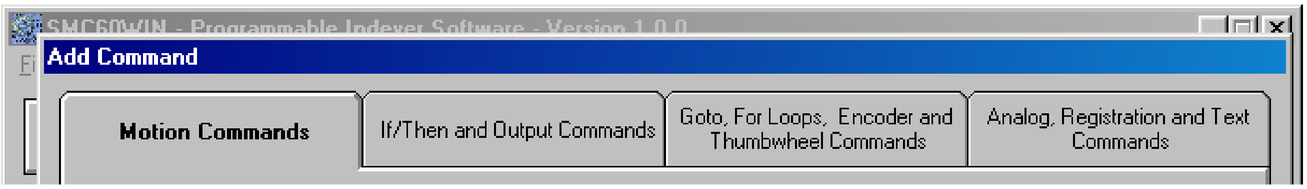
Line	Command	Comments
1	Acceleration=10000 steps/(sec*sec)	
2	Base speed=500 steps/sec	
3	Label=START	
4	Max speed=5000 steps/sec	
5	Direction CW	
6	Go relative 1600 steps	
7	Finish move	
8	Inner loop=4 times to START	
9	Direction CCW	
10	Home type 1 (home limit switch only)	
11	Finish move	
12	Goto START	
13	End of program	

Memory: 42 of 2046 bytes used.

C:\Sample1.mdb

Add/Change/Insert Commands

The Add/Change/Insert commands contain four different tab sheets, which are "Motion Commands", "If/Then and Output Commands", "Goto, For Loops, Encoder and Thumbwheel Commands" and "Analog, Registration and Text Commands".

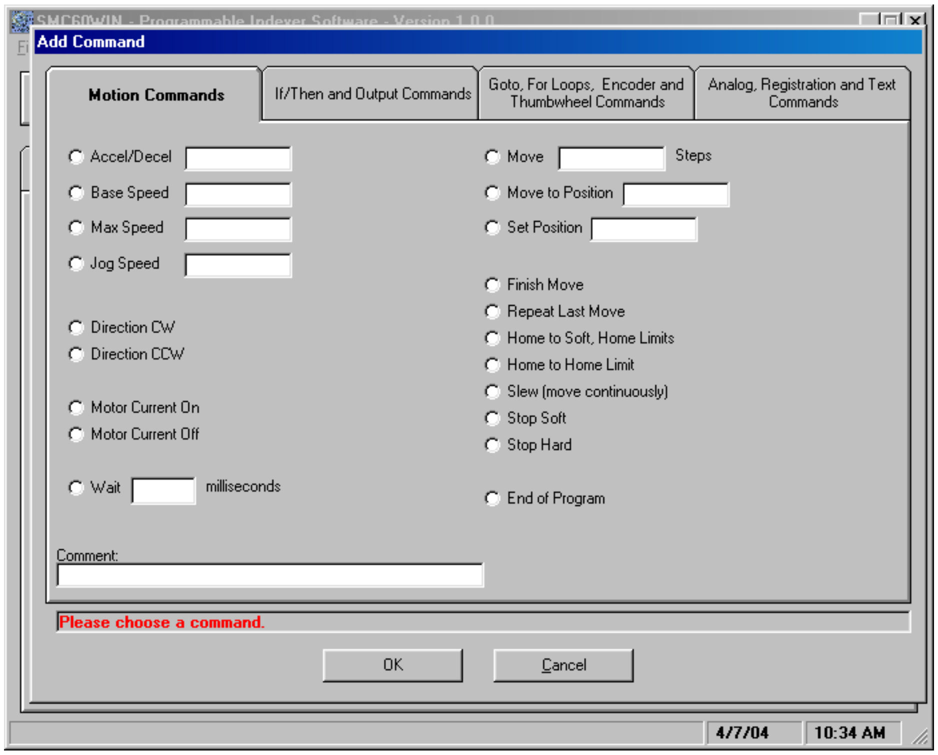

SMC60WIN - Programmable Indexer Software - Version 1.0.0
Add Command
Motion Commands
If/Then and Output Commands
Goto, For Loops, Encoder and Thumbwheel Commands
Analog, Registration and Text Commands

Motion Commands	Software section that allows user to enter speeds, positions, direction, etc.
If/Then and Output Commands	Software section that allows user to manipulate conditional statements and I/0 routines.
Goto, For Loop, Encoder and Thumbwheel Commands	Software section that allows user to manipulate branching and loop routines ,enter encoder parameters and control the thumbwheel switches.
Analog, Regisration and Text Commands	Software section that allow user to enter parameters for analog speed and position limits, index on the fly and output on the fly parameters and text strings to be sent.

These tab sheets is where the program functions are selected to be added to or to change existing lines of programming code in the Create and Edit Program tab.

- To add a line of motion control, select appropriate command, and if required, enter the required value for that particular command. Then, click **OK**.
- Comment is optional, for any lines of code.
- The text box above the **OK** and **Cancel** buttons will display useful information about each command.

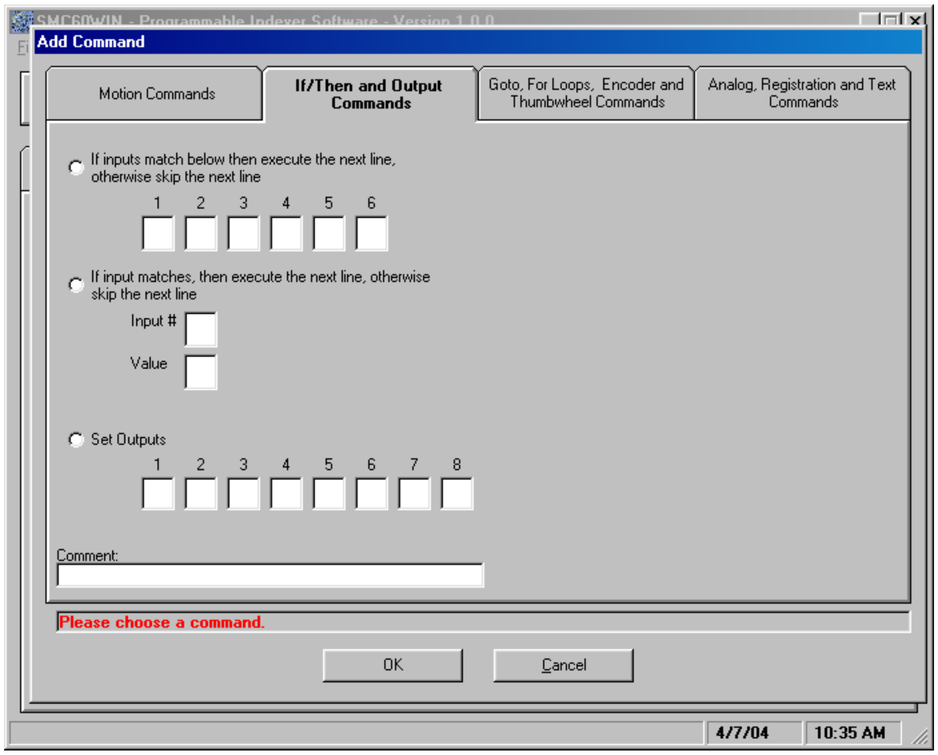
Add Tab Sheets - Motion Commands



Please choose a command.

Accel/Decel	Set program acceleration & deceleration parameter. (step/sec^2)
Base Speed	Set program base (start) speed rate. (step/sec)
Max Speed	Set program maximum (running) speed rate. (step/sec)
Set Jog Speed	Set program jogging speed rate. (step/sec)
Set Position	Set motor position.
Direction CW (CCW)	Set direction to clockwise or counter-clockwise.
Motor Current ON (OFF)	Set the current in the motor on or off.
Wait ___ Milliseconds	This command allows the user to enter a delay in milliseconds.
Move ___ Steps	Relative move command will allow motor to move the defined number of steps entered.
Move to Position	Absolute move command will move motor to the position specified.
Set Position	Allows user to change the position register in the controller.
Finish Move	Command will allow any motion command to be completed before continuing to the next line of code. This command should be used after every motion command.
Repeat Last Move	Command will repeat the previous index move. Do not use with encoder autocorrect enabled.
Home to Soft, Home Limits	Command will begin motion in the direction last entered, seeking the soft input first to slow the motor down to base speed, then to stop when the home limit is triggered.
Home to Home Limit	Command will begin motion seeking the home limit which will slow the motor down to base speed, reverse the direction and stop when the home limit is no longer triggered.
Slew (move continuously)	Command will ramp motor up to max speed and keep moving until triggered to stop.
Stop Soft	Ramp motor down to base speed and stop.
Stop Hard	Stop any motor motion immediately.
End of Program	This command is required as the last command in the program code.

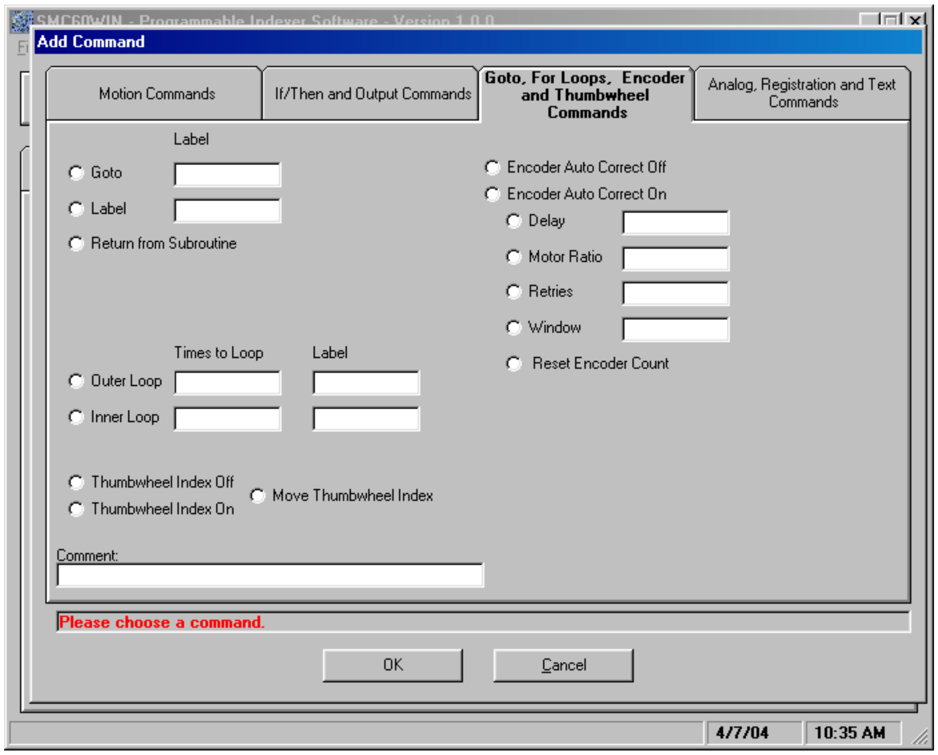
Add Tab Sheets - If/Then and Output Commands


SMC60WIN - Programmable Indexer Software - Version 1.0.0
Add Command
Motion Commands
If/Then and Output Commands
Goto, For Loops, Encoder and Thumbwheel Commands
Analog, Registration and Text Commands
If inputs match below then execute the next line, otherwise skip the next line
1 2 3 4 5 6
If input matches, then execute the next line, otherwise skip the next line
Input #
Value
Set Outputs
1 2 3 4 5 6 7 8
Comment:
OK
Cancel
4/7/04
10:35 AM

Please choose a command.

If inputs match below then execute the next line, otherwise skip the next line	This conditional command allows the user to execute the next line of code if the inputs triggered match the given value. If the inputs do not match, the next line is skipped.
If input matches, then execute the next line, otherwise skip the next line	This conditional command allows the user to execute the next line of code if the specific input triggered matches the given value. If the input does not match, the next line is skipped.
Set Outputs	The outputs can be turned (on=1) or (off=0). These outputs can be used to trigger PLC operations, relays, solenoids, etc.

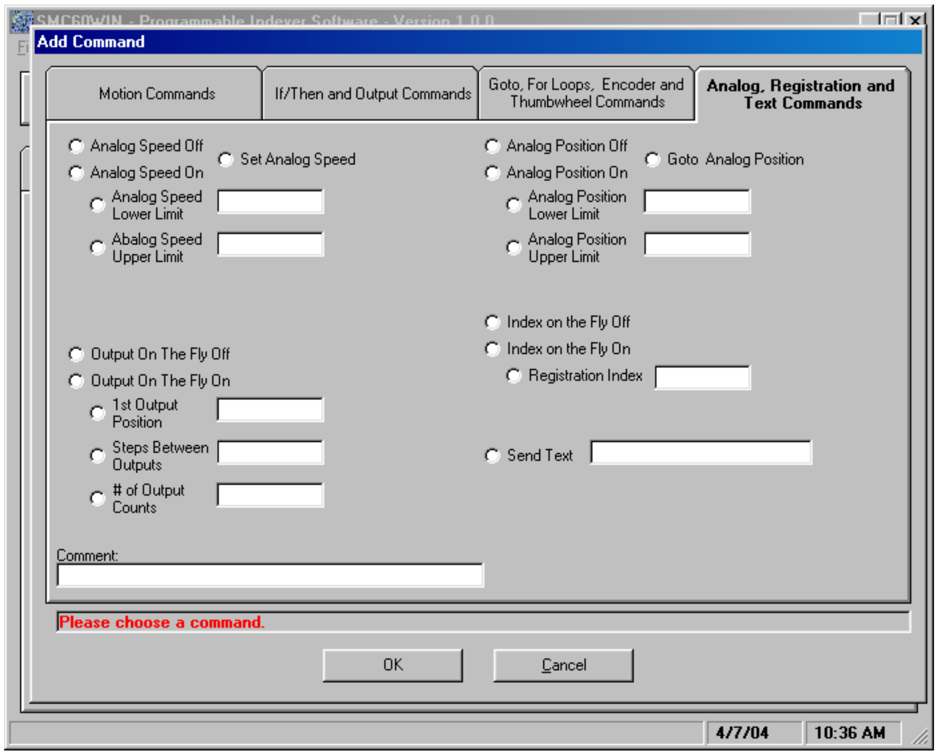
Add Tab Sheets - Goto, For Loops, Encoder and Thumbwheel Commands



Please choose a command.

Goto	Command allows the program to jump to the specified label.
Label	Command inserts a label for goto and loop commands.
Return from Subroutine	Command will return to the last goto and execute the next line of code.
Outer Loop	Command allows a sequence of commands to be looped a specific number of times to a label. This label must be before the outer loop command.This command cannot be used within an inner loop.
Inner Loop	Command allows a sequence of commands to be looped a specific number of times to a label. This label must be before the inner loop command. This command can be used within an outer loop.
Thumbwheel Index Off	Command will disable the ability to use the thumbwheel switches to index.
Thumbwheel Index On	Command allows the user to use the thumbwheel index. Switch 2 must be in the thumb-wheel position for the thumbwheel to be enabled.
Move Thumbwheel Index	Relative move command will allow motor to move the defined number of steps set by the thumbwheel switches.
Encoder Auto Correct Off	Command will disable the autocorrect of the motor. The encoder value can still be read.
Encoder Auto Correct On	Command will enable the use of the encoder and autocorrect the motor if needed, based on the encoder registers.
Delay	Command sets a time delay used for settling time needed after an index is finished: The delay occurs before the encoder count is read.
Motor Ratio	Command sets the number of encoder counts to one motor step.
Retries	Command sets the number of times the motor can autocorrect itself before erroring. When the autocorrect errors, Output 8 will be triggered.
Window	Command sets the number of encoder counts the motor is allowed to be off.
Reset Encoder Count	Command will reset the encoder count to 0 (zero).

Add Tab Sheets - Analog, Registration and Text Commands

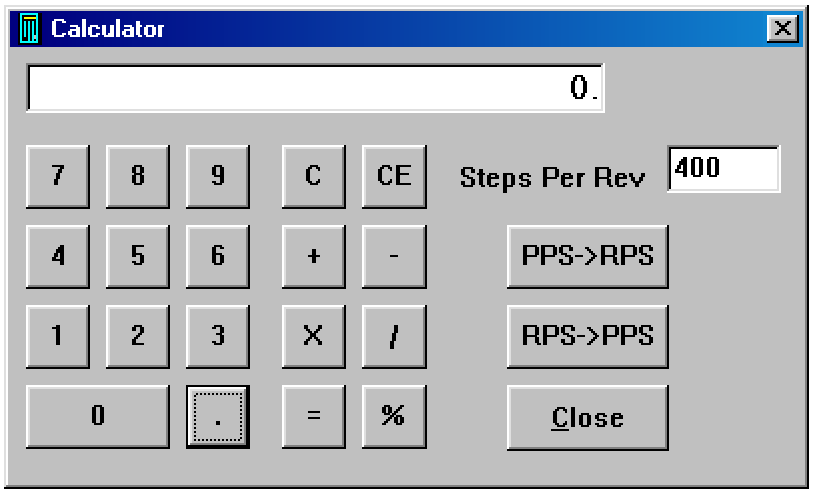


Please choose a command.

Analog Speed On (Off)	Sets the analog speed input feature on or off.
Set Analog Speed	Sets the maximum (running) speed based on the analog input.
Analog Speed Lower Limit	Sets the analog speed lower limit to the value specified.
Analog Speed Upper Limit	Sets the analog speed upper limit to the value specified.
Analog Position On (Off)	Sets the analog position input feature on or off.
Goto Analog Position	Motor will move to the position based on the analog input.
Analog Position Lower Limit	Sets the analog position lower limit to the value specified.
Analog Position Upper Limit	Sets the analog position upper limit to the value specified.
Output on the Fly On (Off)	Sets the output on the fly feature on or off.
1st Output Position	Sets the position that the 1st output will trigger during and index.
Steps Between Outputs	Sets the number of steps between the triggered output.
# of Output Counts	Set the number of times the output will trigger.
Index on the Fly On (Off)	Sets the index on the fly feature on or off.
Registration Index	Sets the value of the index that will occur after the registration input is activated.
Send Text*	Will send the entered text string back to the user. (20 characters maximum)

*Note that this command should only be used when the device is intended to run in the AA ASCII protocol.

Calculator



PPS->RPS	Convert from pulses per second to revolution per second.
RPS->PPS	Convert from revolution per second to pulses per second.
Steps Per Rev	Enter the number of steps per revolution of the step motor. The default is for a 200 step/rev motor in half step, which is equal to 400.
Close	Exit the Calculator.

Section 4: Communication Protocols

Modbus RTU

Modbus is a communication protocol commonly used in industrial applications. This protocol can be implemented in two different formats, which are defined by the way transmissions are framed and the format in which the data is transmitted. The two formats are ASCII and RTU (Remote Terminal Unit). In the former, data is transmitted as a string of ASCII characters with two characters representing a single byte, and the message is framed by a colon at the start and a carriage return/line feed combination at the end. The one implemented on this device, however, is the RTU format. In this format, messages are transmitted as a string of hexadecimal values, and the message is framed on both ends by an idle time-out equal to the time it takes to transmit 3.5 characters at the implemented baud rate. This protocol allows the device to communicate with Human Machine Interfaces (HMIs), so that it can be polled and monitored while running. In addition, this allows for commands to be sent from the HMI itself, giving the user full control over the functions of the controller. Modbus RTU is the default communication protocol of the PCL601, and it can operate as a Modbus Slave only.

COM PORT Settings

Baud Rate:	38400
Start Bits:	None
Data Bits:	8
Parity:	Even
Stop Bits:	None
Flow Control:	None

Message Framing

Start	Address	Function	Data	CRC	End
$\geq 3.5t_{char}$(1)	1 byte	1 byte	up to 252 bytes	2 bytes	$\geq 3.5t_{char}$(1)

Address Field:
The device address of the PCL601 can range from 1 to 99. Address 0 is reserved for broadcasting to all devices. It is important to note that no response is obtained from the device when broadcast is used.

Function Field:
Valid codes are in the range of 1 – 255 decimal. The device only supports the function codes presented in this manual. See Appendix 3 for information on the functions supported by the device.

Data Field:
The data field mainly contains the data being written or read as well as other information needed for the specific Modbus function being used. See Appendix 3 for more details.

Cyclical Redundancy Check:
Cyclical Redundancy Check (CRC) is a 16- bit value that allows the receiver to verify the validity of the message received. The low byte transmitted first, followed by the high byte. The method used by this device for calculating the 16-bit CRC value can be found in Appendix 2**.**

(1) t_{char} is the time it takes to transmit a single character at the implemented baud rate. A character is an individual package including the start bit, data byte, parity bit, and stop bit; in other words, the character is equivalent to 11 bits. In addition, the message must be continuous with the delay between each character in the same message being no longer that $1.5t_{char}$

$$t_{char} = (11bits) / (38400bits/s) \approx 286.5\mu s, \qquad 1.5t_{char} \approx 430\mu s, \qquad 3.5t_{char} \approx 1ms$$

Command Register

The command register in the PCL601 is a Modbus compliant data register (4x reference) that allows the user to send live commands to the device. This register is mapped as register one, and it can be read or written to; however, this register only works a buffer register for the command code being sent to the device. Once the command is being executed, the command register value will read as zero. The command codes that can be sent to the device through the command register range from 1-15, and they are presented below.

1 - Go

Code:	1
Description:	This command is used to send a set number of clocks out of the PCL601 controller. The number of steps or absolute position data register must be written to before this command. The ramp profile is specified by the base speed, max speed, and acceleration/deceleration values.

2 - Home Type 0

Code:	2
Description:	In type 0 homing, the PCL601 will send clocks until a soft limit is reached, then ramp down to base speed. Clocks will continue at base speed until a home limit is reached. The ramp profile is specified by the base speed, max speed, and acceleration/deceleration values.

3 - Home Type 1

Code:	3
Description:	In type 1 homing, the PCL601 will send clocks until a home limit is reached, ramp down to base speed, change directions and run at base speed until the release of the home limit input. The ramp profile is specified by the base speed, max speed, and acceleration/deceleration values.

4 - Go slew

Code:	4
Description:	This command will send clocks out to the PCL601. The only commands that can stop the clocks are stop hard or stop soft. Motion can also be stopped by using the limit switch inputs. The ramp profile is specified by the base speed, max speed, and acceleration/deceleration values.

5 - Stop Motion

Code:	5
Description:	This command will stop all motion. It can also be used to stop the current program that is running.

6 - Stop Soft

Code:	6
Description:	This command will ramp the clocks down to base speed. The move type then determines what will happen. In a relative or absolute type motion the PCL601 controller will continue to the set position and stop. In a slew type motion the PCL601 controller will ramp down and stop. In a home type motion the PCL601 controller will ramp down and run at base speed, until the home limit is activated.

7 - Set Max Speed from Analog Input

Code:	7
Description:	This command uses the voltage on input 1 to calculate and set the max speed. The analog speed/position enable register must be set to 1, and the analog speed upper and lower limit values must be written before the execution of this command.

8 - Set Absolute Position from Analog Input

Code:	8
Description:	This command uses the voltage on input 1 to calculate and set the number of clocks for the PCL601 to send out following a Go command. The analog position must be enabled for this command to work. Motion is not activated by this command, it only sets the number of steps required to go to the absolute position determined from the analog input. The analog speed/position enable register must be set to 1, and the analog speed upper and lower limit values must be written before the execution of this command.

9 - Set Number of Steps from Thumbwheel

Code:	9
Description:	This command reads the thumbwheel switches to set the number of clocks for the PCL601 to send out following a Go command. Motion is not activated by this command, it only sets the index register. The thumbwheel enable register must be set to 1 before the execution of this command.

10 - Clear Errors

Code:	10
Description:	This command clears the contents of the error code register. For a description of the error codes see page 43.

11 - Switch to AA ASCII protocol

Code:	11
Description:	This command switches the device to the AA ASCII protocol.

12 - Run Program

Code:	12
Description:	This command tells the PCL601 to execute the program stored in its internal memory. The device can be programmed via the SMW60WIN software.

13 - Autorun Off

Code:	13
Description:	This command disables the program-autorun feature of the PCL601. This means the device will NOT automatically execute the program stored in its internal memory upon being powered.

14 - Autorun On

Code:	14
Description:	This command enables the program-autorun feature of the PCL601. When given this command, the device will automatically execute the program stored in its internal memory every time it is powered until it receives an Autorun Off command.

15 - Encoder reset

Code:	15
Description:	This command will reset the internal encoder position register to 0.

Anaheim Automation ASCII

The second protocol used by the device is a custom communication protocol used in Anaheim Automation Programmable Controllers. In this protocol, data is transmitted as strings of ASCII characters, with alpha characters representing the function being requested and numerical characters representing the function parameters in decimal form. This protocol also allows the device to communicate and be programmed via the SMC60WIN software.

COM PORT Settings

Baud Rate:	38400
Start Bits:	None
Data Bits:	8
Parity:	None
Stop Bits:	1
Flow Control:	Xon/Xoff

Message Format

Start	Address	Instruction	Parameters(1)	End
‘@’ character	1 - 2 characters	1 - 2 characters	0 - 8 characters	Carriage Return

The message does not have to be sent as a continuous string; this allows user to use a simple terminal program to communicate and control the device via a serial connection. The command must not contain any spaces.

(1) Not all commands require parameters. See Command Summary for details.

Start / Unit Selection:
In order to select a unit, the @ command followed by the address of the unit must be sent.
For example: @13$ (Unit 13 is selected, Command $ is used)

Address Field:
The device address of the PCL601 can range from 1 to 99.

Direct Talk

Direct mode is used to directly control motion for real time movements through serial communication. The PCL601 controller has 40 commands, which are easy to remember for direct movement of a step motor.

Command Summary:

A - Acceleration/Deceleration
B - Base speed
C - Steps between outputs
D - 1st output on the fly position
EA - Encoder autocorrect enabled
ED - Encoder delay
EM - Encoder motor ratio
ER - Encoder retries
ET - Encoder reset
EW - Encoder window
G - Go number of steps
H - Home
I - Read inputs
J- Fast jog speed
M - Max speed
N - Number of steps
O - Set outputs
P - Absolute position
S - Go slew
T - Motor current enabled
V - Verify
Z - Position
! - Error codes register
$ - Version number
% - Verify axis number
‘ - Index on the fly enabled
(- Output on the fly enabled
+ - Clockwise direction
, - Stop soft
- - Counterclockwise direction
. - Stop hard
/ - Thumbwheel enabled
: - Analog position enabled
; - Analog speed enabled
[- Analog speed lower limit
] - Analog speed upper limit
^ - Number of outputs
{ - Analog position lower limit
} - Analog position upper limit
~ - Set address of PCL601
| - Switch to Modbus RTU

A - Acceleration/Deceleration

Format:	A[value]
Description:	This command sets the acceleration profile which is an integer value between 100 and 9,999,999. The higher the value, the faster the motor acceleration.
Range:	100 - 9,999,999

B - Base speed

Format:	B[value]
Description:	This command sets the base (start) speed for motion. This value must be set before motion begins and be less then the maximum speed and fast jog speed.
Range:	1 - 5,000

C - Number of steps between outputs during output on the fly

Format:	C[value]
Description:	This command sets the number of steps between when output 1 turns on during an output on the fly move. If only one output turn on is needed, set this value to 0. This command is used in conjunction with the output on the fly enabled “(“ command.
Range:	0 - 65,535

D - 1st Output on the fly position

Format:	D[value]
Description:	This command sets the position at which output 1 will turn on during an output on the fly move. This command is used in conjunction with the output on the fly enabled “(“ command.
Range:	0 - 65,535

EA - Encoder autocorrect enabled

Format:	EA[0 or 1]
Description:	This command will either enable or disable the encoder autocorrect function. To enable the function use a 1, to disable the function use a 0. When this function is enabled, the relative register is used to calculate the encoder position, therefore before the next move is made, the relative register needs to be set. This command is used in conjunction with the encoder delay “ED”, encoder ratio “EM”, encoder retries “ER” and encoder window “EW” commands.

ED - Encoder delay

Format: ED[value]

Description: This command sets the wait time a specified number of milliseconds after a relative index or absolute move is finished, before reading the encoder. This is used to remove the ringing that might be associated with the mechanics of the system. This command is used in conjunction with the encoder autocorrect “EA” command.

Range: 0 - 65,535

EM - Encoder motor ratio

Format: EM[value]

Description: This represents the ratio for the number of encoder pulses to one motor step. An example is for a 1000 line quadrature encoder and a 400 step/revolution motor, the motor ratio is   (1000 * 4) / 400 = 10. This command is used in conjunction with the encoder autocorrect EA command.

Range: 1 - 255

ER -Encoder retries

Format: ER[value]

Description: This is the number of times the PCL601 controller will try to autocorrect the motor before erroring. This command is used in conjunction with the encoder autocorrect EA command. When the autocorrect errors, Output 8 will be triggered.

Range: 0 - 255

ET - Encoder reset

Format: ET

Description: This command will reset the internal encoder count register to 0.

EW -Encoder window

Format: EW[value]

Description: This is the allowable error in encoder pulses (either plus or minus) from the desired position that is allowed before the motor autocorrects. This command is used in conjunction with the encoder autocorrect EA command.

Range: 0 - 255

G - Go number of steps

Format: G

Description: This command is used to send a set number of clocks out of the PCL601 controller. An N or P command must be entered before the G command. The ramp profile is specified by the B (base speed), M (max speed), and A (acceleration/deceleration) commands.

H - Home

Format: H[0 or 1]

Description:
Home Types: H0: In type 0 homing, the PCL601 will send clocks until a soft limit is reached, then ramp down to base speed. Clocks will continue at base speed until a home limit is reached. The ramp profile is specified by the B (base speed), M (max speed), and A (acceleration/deceleration) commands.

H1: In type 1 homing, the PCL601 will send clocks until a home limit is reached, ramp down to base speed, change directions and run at base speed unit the release of the home limit input. The ramp profile is specified by the B (base speed), M (max speed), and A (acceleration/deceleration) commands.

I - Read inputs

Format 1: IR

Description: This command returns the complemented binary value of the inputs to the PC. The inputs will read as 0 when they are open or high; On the other hand, they will return a 1 when they are grounded. For example; if all inputs are active (grounded), the command will return a 63. If all inputs are inactive (open), the command will return a 0. Input 1 is the LSB and input 6 is the MSB.

Format 2: I[input]

Description: This command returns the binary or logic value of the selected input to the PC. If the input is open or high, it will return a 1. If the input is grounded or low, it will return a 0.

Range: 1 - 6

J - Fast jog speed

Format: J[value]

Description: This command sets the fast jog speed. This value must be set before motion begins and be greater than the base speed.

Range: 1 - 50,000

M - Max speed and analog speed

Format 1:	M[value]
Description:	This command sets the maximum (running) speed for motion. This value must be set before motion begins, and be greater than the base speed.
Range:	1 - 50,000
Format 2:	M
Description:	This command uses the voltage on input 1 to calculate and set the max speed. The analog speed must be enabled for this command to work. This command is used in conjunction with the analog speed “;”, the analog speed lower limit “[“, and the analog speed lower limit “]” commands. This value must be set before motion begins.

N - Number of steps

Format 1:	N[value]
Description:	This command sets the number of clocks for the PCL601 to send out following a G command. It is also used to set the registration index during and index on the fly move. Motion is not activated by this command; it only sets the index register.
Range:	0 - 8,388,607
Format 2:	N
Description:	This command reads the thumbwheel switches to set the number of clocks for the PCL601 to send out following a G command. For this command to work SW2 must be in the TWS position, and the thumbwheel enable bit must be enabled. Motion is not activated by this command, it only sets the index register. This command is used in conjunction with the thumbwheel enabled “/” command.

O - Set outputs

Format 1:	OR[value]
Description:	This command sets the output register according to the binary value entered. Output 1 is the LSB and output 8 is the MSB.
Range:	0 - 255
Format 2:	O[output]=[0 or 1]
Description:	This command sets the selected output either on or off. A 1 will turn the output on (0VDC) and a 0 will turn the output off (open).
	Range: 1 - 8

P - Absolute position and analog position

Format 1:	P[value]
Description:	This command calculates and sets the number of clocks for the PCL601 to send out following a G command. Motion is not activated by this command; it only sets the register. (N = P - Z)
Range:	-8,388,607 to +8,388,607
Format 2:	P
Description:	This command uses the voltage on input 1 to calculate and set the number of clocks for the PCL601 to send out following a G command. The analog position must be enabled for this command to work. Motion is not activated by this command, it only sets the register (N = P - Z). This command is used in conjunction with the analog position“:”, the analog position lower limit “{“, and the analog position lower limit “}” commands.

S - Go slew

Format:	S
Description:	This command will send clocks out to the PCL601. The only commands that can stop the clocks are; “.” (stop motion) or “,” (soft limit). Motion can also be stopped by using the limit switch inputs. The ramp profile is specified by the B (base speed), M (max speed), and A (acceleration/deceleration) commands.

T - Motor current enabled

Format:	T[0 or 1]
Description:	This command will control the On/Off output, which is designed to connect to the on/off input of Anaheim Automation’s step motor drivers. To energize and allow current to flow through the coil of the motor, set the value to 1. To de-energize and turn the current off to the motor, set the value to 0. This is a dedicated output, and not controlled with the output register.

V - Verify

Format: V[command]

Description: This command can be used with most commands to verify the register contents. This is a read only command. Valid Commands are shown below.

Command	Description	Command	Description
A	Verify acceleration/deceleration	O	Verify outputs
B	Verify base speed	T	Verify motor current (1 is on, 0 is off)
C	Verify steps between outputs on the fly	Z	Verify position
D	Verify 1st output on the fly position	‘	Verify index on the fly enabled**
EA	Verify encoder autocorrect enabled**	(	Verify output on the fly enabled**
ED	Verify encoder delay	+	Verify Direction
EM	Verify encoder motor ratio	/	Verify thumbwheel index enabled**
EP	Verify encoder position	:	Verify analog position enabled**
ER	Verify encoder retries	;	Verify analog speed enabled**
EW	Verify encoder window	[	Verify lower analog speed value
F	Verify if controller is busy	]	Verify upper analog speed value
J	Verify jog speed	^	Verify number of outputs
L	Verify Limits (Hard-Bit 0, Soft-Bit 1)	{	Verify lower analog position value
M	Verify max speed	}	Verify upper analog position value
N	Verify number of steps		** 1 is enabled, 0 is disabled

Z - Position

Format: Z[value]

Description: This command sets the current position as a reference. This register can contain a positive or negative value but, cannot be changed while motion is in progress.

Range: -8,388,607 to +8,388,607

! - Error codes register

Format: !

Description: This command requests the PCL601 controller to get the current error code and print it to the screen. For a description of the error codes see page 43.

$ - Version number register

Format: $

Description: This command requests the PCL601 controller to return its internal firmware version number.

% - Verify address register

Format: % (No address is needed before this function. @% will return the address)

Description: This command requests the PCL601 controller to return its internal address number to the PC or PLC.

‘ - Index on the fly enabled

Format: ‘[0 or 1]

Description: This command will either enable or disable the index on the fly function. To enable the function, use a 1. To disable the function use a 0. This command is used in conjunction with the number of steps “N” and go slew “S” commands.

(- Output on the fly enabled

Format: ([0 or 1]

Description: This command will either enable or disable the output on the fly function. To enable the function, use a 1. To disable the function use a 0. This command is used in conjunction with the number of steps between outputs “C”, 1st output position “D”, and number of outputs “^” commands.

+ - Clockwise

Format: +

Description: This command sets the direction output to clockwise.

, - Soft Limit Input Bit

Format: ,

Description: This command will ramp the clocks down to base speed. The move type then determines what will happen. In a relative or absolute type motion the PCL601 controller will continue to the set position and stop. In a slew type motion the PCL601 controller will ramp down and stop. In a home type motion the PCL601 controller will ramp down and run at base speed, until the home limit is activated.

- - Counter-Clockwise

Format: -

Description: This command sets the direction output to counterclockwise.

. - Stop Motion

Format: .

Description: This command will stop all motion. It can also be used to stop the current program that is running.

/ - Thumbwheel index enabled

Format:	/[0 or 1]
Description:	This command will either enable or disable the ability to use the thumbwheel switches for indexing. If enable, SW2 must be in the TWS position for the thumbwheel to be connected to the processor.

: - Analog position enabled

Format:	:[0 or 1]
Description:	This command will either enable or disable input 1 to be used to set the analog position. To enable the function, use a 1. To disable the function use a 0. This command is used in conjunction with the analog position lower limit "{" and analog position upper limit "}" commands.

; - Analog speed enabled

Format:	;[0 or 1]
Description:	This command will either enable or disable input 1 to be used to set the analog speed. To enable the function, use a 1. To disable the function use a 0. This command is used in conjunction with the analog speed lower limit "[" and analog speed upper limit "]" commands.

[- Analog speed lower limit

Format:	[[value]
Description:	This command sets the lower limit that is used during the calculation following an M command for the analog speed input. This command is used in conjunction with the analog speed enabled ";" and max speed "M" commands.
Range:	1 - 50,000

] - Analog speed upper limit

Format:	][value]
Description:	This command sets the upper limit that is used during the calculation following an M command for the analog speed input. This command is used in conjunction with the analog speed enabled ";" and max speed "M" commands.
Range:	1 - 50,000

^ - Number of outputs during output on the fly

Format: ^[value]

Description: This command sets the number of times output 1 will turn on during an output on the fly move. This command is used in conjunction with the output on the fly enabled "(" command.

Range: 0 - 255

{ - Analog position lower limit

Format: {[value]

Description: This command sets the lower limit that is used during the calculation following a P command for the analog position input. This command is used in conjunction with the analog position enable ":" and absolute position "P" commands.

Range: 0 - 65,535

} - Analog position upper limit

Format: }[value]

Description: This command sets the upper limit that is used during the calculation following a P command for the analog position input. This command is used in conjunction with the analog position enable ":" and absolute position "P" commands.

Range: 0 - 65,535

~ - Set address register

Format: ~[value] (No address is needed before this function. @~[value] will set the address)

Description: This command sets the address for communication inside the PCL601 controller.

Range: 0 - 99

| - Switch to Modbus RTU

Format: | (Vertical bar ASCII character; hexadecimal value = 0x7C)

Description: This command switches the communication protocol from AA ASCII to Modbus RTU Note that the same effect can be accomplished by physically pressing the reset button on the device or powering it down and powering it back up.

Section 5: Troubleshooting

Problem:

Cannot establish communications with the PCL601.

Possible Solutions:

1) Make sure the PCL601 is using the appropriate communication protocol. If connecting to the SMC60WIN software, the device must be in AA ASCII protocol.
2) If connecting to an HMI, make sure both devices are using Modbus RTU to communicate.
3) Make sure the PCL601 controller has power. Is the Green LED on?
4) Check the RS232/RS485 connections.
5) Check for loose cable connections either on the PCL601 controller or COM Port.
6) Was the software installed successfully?
7) Go to **Setup | Com Port Settings** and verify COM port and baud rate settings.
8) Go to **Setup | Axis** and verify address settings are the same.
9) Click on **Setup |Connect** icon to communicate with the PCL601 controller.
10) If problems still exist, contact Anaheim Automation Tech Support.

Anaheim Automation, Inc. Tech Support:	4985 E Landon Drive Anaheim, CA 92807 Phone: (714) 992-6990 Fax: (714) 992-0471 www.anaheimautomation.com

Problem:

There is no power to the PCL601 controller.

Possible Solutions:

1) Is the PCL601 controller connected to the appropriate power supply?
2) Check for any blown fuses in line with the PCL601 controller.
3) If problems still exist, contact Anaheim Automation, Inc. Tech Support.

Problem:

My program won’t “Autostart”.

Possible Solutions:

1) Verify that the Autostart Function has been enabled.
2) If the device is connected to an HMI, write the command code **14** to the **command register** and restart device. See page 29 for more details.
3) Go to **Setup | Autostart Program**, click on **Enable**, and restart device.
4) If problems still exist, contact Anaheim Automation Tech Support.

Problem:

The PCL601 controller has a fault condition.

Possible Solutions:

1) Verify your program for improper syntax that may cause an error code.
2) Physically press the reset button on the PCL601 to clear an error.
3) Another way to clear an error is by using either the SMC60WIN software or the direct mode command instructions set.
4) The SMC60WIN can clear an error in the real time motion tab section by clicking on the verify parameters button.
5) The direct mode command "!" can clear an error by simply prompting the error code register to return the value back to the PC or PLC.

Note: Read the Error returned to the screen to better understand what can be causing the fault condition. The error is returned in binary coded decimal format. If two errors were received, their binary values would be added together.

Error Codes

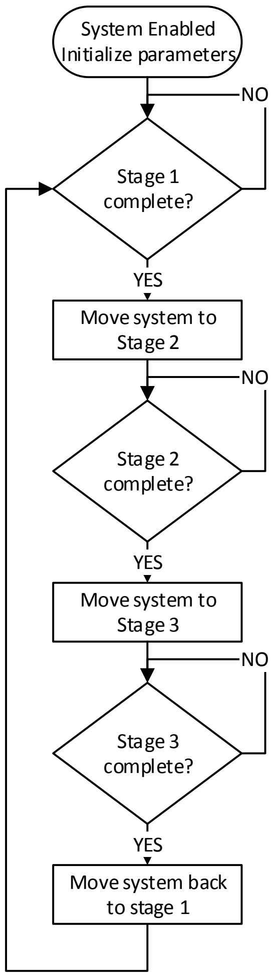
Error Code	Type	Description
1	Receive Overflow Error	The serial communications had a receiving error. This is an internal error caused by the computer.
2	Encoder Error 1	The encoder needed to correct the index.
4	Encoder Error 2	The encoder could not finish the index. Output 8 is set with this error.
8	Command Error	A bad command was sent to the controller. Please check to see that the command being sent is valid.
16	Motor Error	Motor speed profiles are set incorrectly. Please make sure that the base speed is less than the max speed and that the speeds are within their valid ranges.
32	Range Overflow Error	The go to position has an overflow error. This is caused by the P command trying to find a position that is out of its range.
64	Range Error	There was an invalid number of commands and characters sent to the controller. Check to see if the parameters are invalid for the command that was sent.
128	Transmitt Error	To many parameters sent back to the PC. This is an internal error caused by the eeprom.
256	Mode Error	Controller is in a wrong mode. Some commands are good only in programming mode, while others are good only in direct mode. Check the direct mode section to see which commands are good in direct mode.
512	Zero Parameters Error	There were no parameters sent to the controller. A command was sent to the controller that expected to see parameters after the command.
1024	Busy Error	The controller is busy indexing. The controller is sending out clocks to the driver and can not execute the next instruction.
2048	Memory Range Error	The specified address is out of range. This is caused by overflowing the program memory by having a program that is to large.
4096	Memory Command Error	The command pulled from memory is invalid. The command that was stored into the eeprom was non executable by the program. This is an internal error.
8192	Thumbwheel Read Error	There was an error reading the thumbwheel or the thumbwheel is not present.

Section 6: Tutorials

Sample Application using a Human Machine Interface:

In this tutorial, a simple I/O-controlled application using the Kinco MT4230T Human Machine Interface and the PCL601 is illustrated.

Note: The Kinco HMIware software is used to create the interface for this application; therefore, it is recommended that the user be familiar with this software or refer to the software's user manual for more details regarding its functionality. The software can be downloaded at anaheimautomation.com.



Requirements:

All the other system components must be enabled and disabled by the operator from the HMI using one of the outputs on the PCL601.

The following parameters must be initialized:

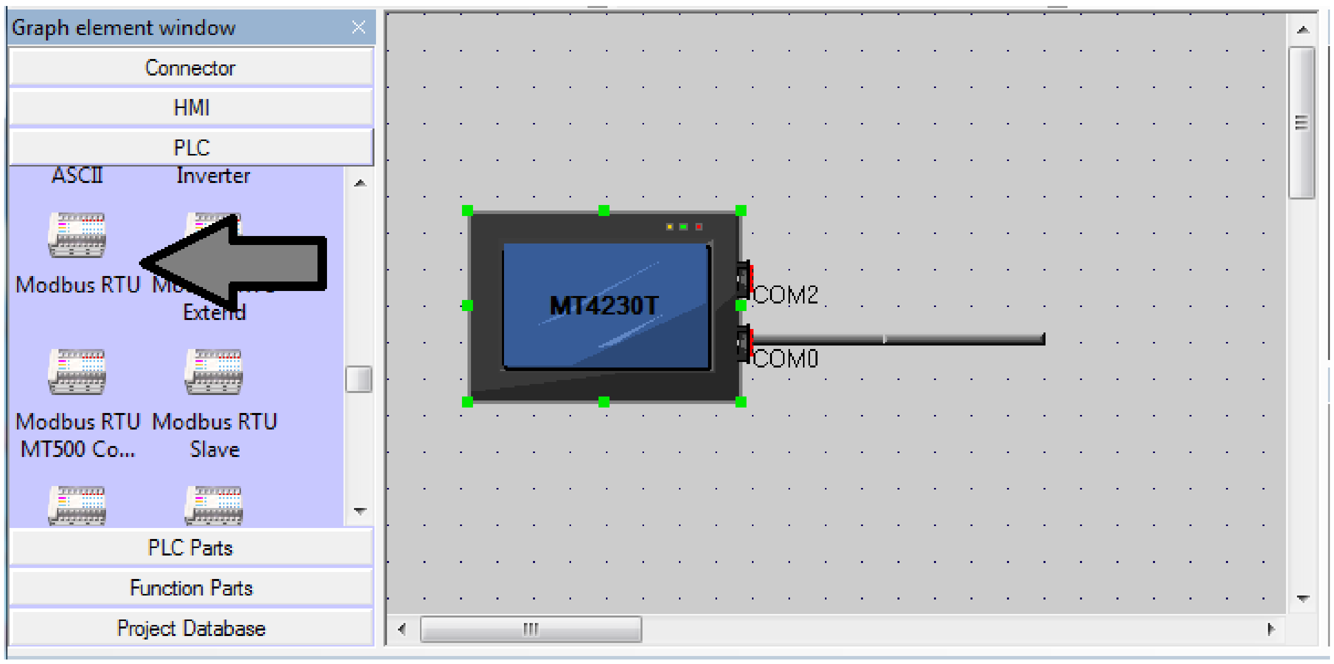
Acceleration = 1000
Base Speed = 500
Max Speed = 5000

The different stages of the system will signal the completion of their repective processes through a pulse signal.

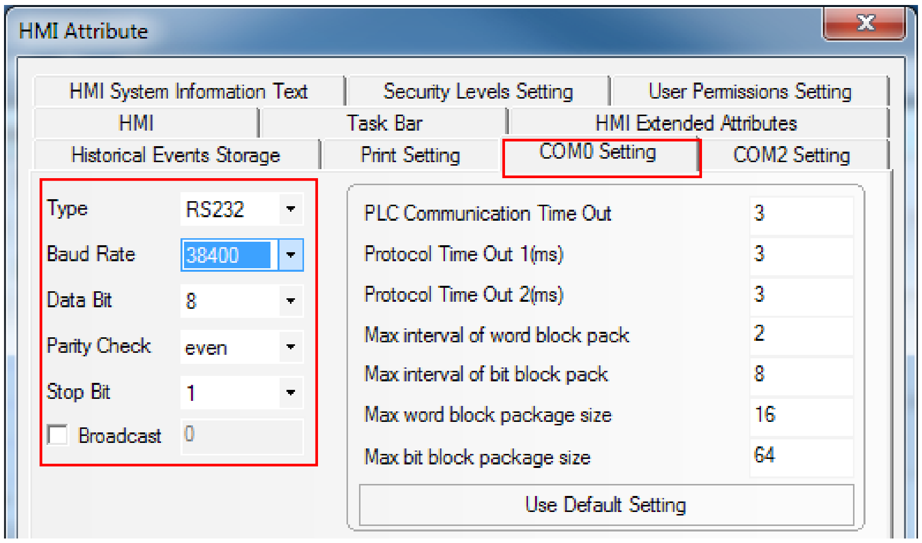
The different stages are positioned as follows:

Stage 1 - position = 0
Stage 2 - position = 4000
Stage 3 - position = 8000

Step 1:
We create a new project and set up our HMI - PLC serial connection. In this tutorial, the HMI being used is the Kinco MT4230T model. For our PLC, we select the Modbus RTU option:



Next, the serial port on the HMI is configured:

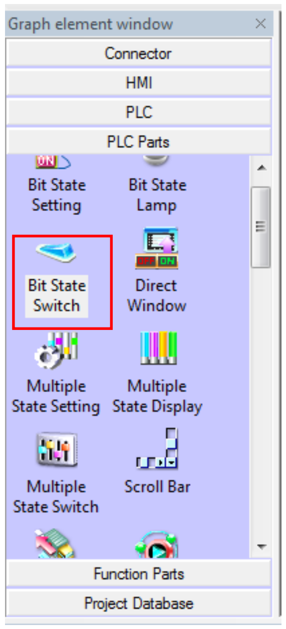


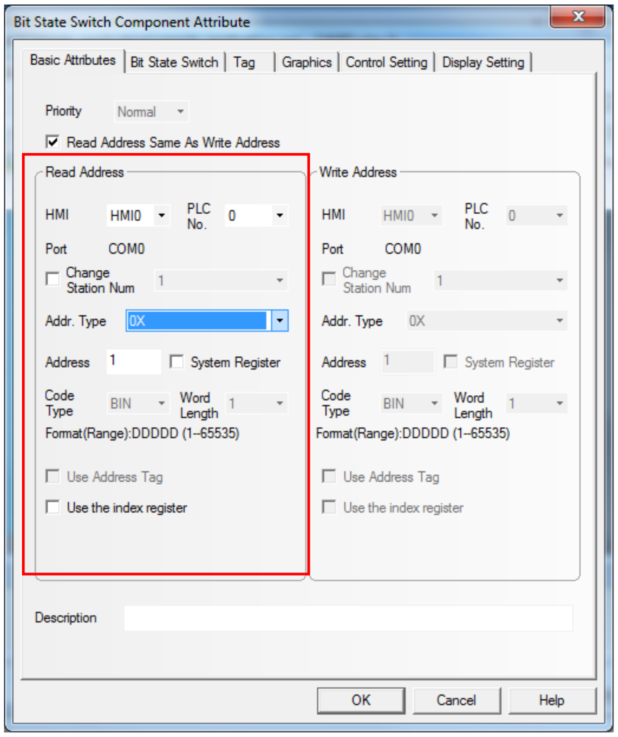
Type	RS232
Baud Rate	38400
Data Bit	8
Parity Check	even
Stop Bit	1
Broadcast	0

PLC Communication Time Out	3
Protocol Time Out 1(ms)	3
Protocol Time Out 2(ms)	3
Max interval of word block pack	2
Max interval of bit block pack	8
Max word block package size	16
Max bit block package size	64

Step 2:
Now, we are ready to start adding components to the interface to be displayed on the HMI. For this application, we will use a single frame.

The first component to be added is a system enable button which will toggle output number 1 on the PCL601. For this, we use a "Bit State Switch" and configure it to toggle output 1. Notice that we need to select address type 0x, which is the reference address for outputs in Modbus. In addition, tags are used to display "ON" and "OFF" on the button itself. The graphics selection is up to the user.




Bit State Switch Component Attribute

Basic Attributes | Bit State Switch | Tag | Graphics | Control Setting | Display Setting

Priority: Normal

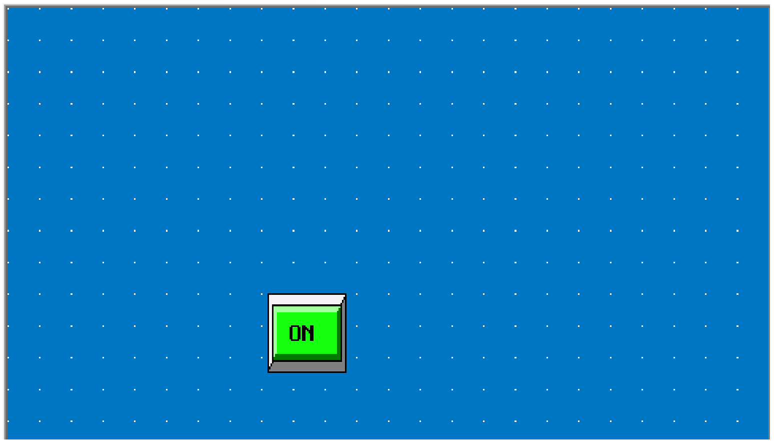
[✓] Read Address Same As Write Address

Read Address
HMI: HMI0 PLC No.: 0
Port: COM0
[ ] Change Station Num: 1
Addr. Type: 0X
Address: 1 [ ] System Register
Code Type: BIN Word Length: 1
Format(Range):DDDDD (1--65535)
[ ] Use Address Tag
[ ] Use the index register

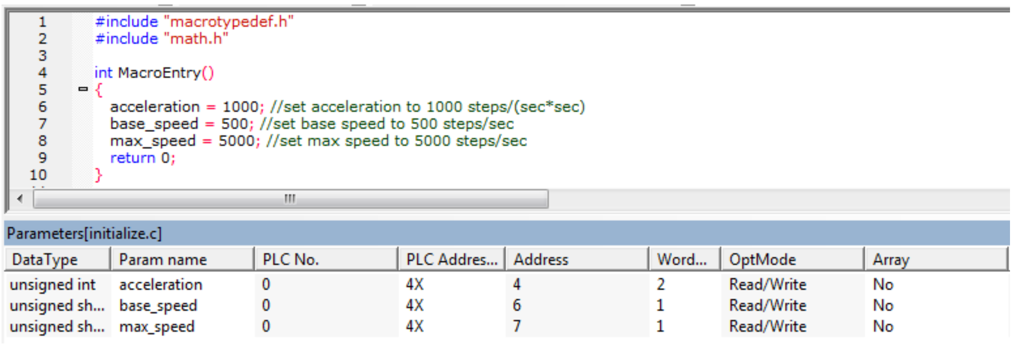
Write Address
HMI: HMI0 PLC No.: 0
Port: COM0
[ ] Change Station Num: 1
Addr. Type: 0X
Address: 1 [ ] System Register
Code Type: BIN Word Length: 1
Format(Range):DDDDD (1--65535)
[ ] Use Address Tag
[ ] Use the index register

Description:

OK | Cancel | Help



Additionally, we want this button to initialize the values for acceleration, base speed, and max speed. To accomplish this, we create the following macro and name it “initialize”. Note that the parameters must be of address type 4x, which are data registers. Refer to Appendix 1 for more details on the Memory Map.

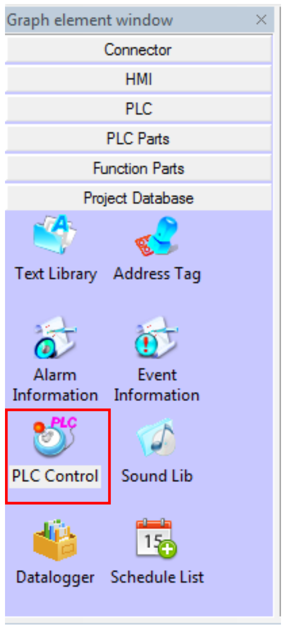


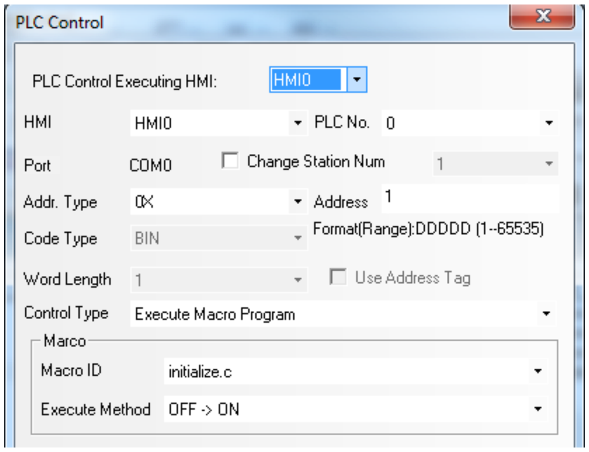
```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
  acceleration = 1000; //set acceleration to 1000 steps/(sec*sec)
  base_speed = 500; //set base speed to 500 steps/sec
  max_speed = 5000; //set max speed to 5000 steps/sec
  return 0;
}
```

Parameters[initialize.c]

DataType	Param name	PLC No.	PLC Addres...	Address	Word...	OptMode	Array
unsigned int	acceleration	0	4X	4	2	Read/Write	No
unsigned sh...	base_speed	0	4X	6	1	Read/Write	No
unsigned sh...	max_speed	0	4X	7	1	Read/Write	No

Now that the macro has been created, we need to add a “PLC Control” event that will trigger it. The macro has to be triggered when output 1 is active:





PLC Control

PLC Control Executing HMI: HMI0

HMI: HMI0 PLC No.: 0

Port: COM0 Change Station Num: 1

Addr. Type: 0X Address: 1

Code Type: BIN Format(Range):DDDDD (1--65535)

Word Length: 1 Use Address Tag

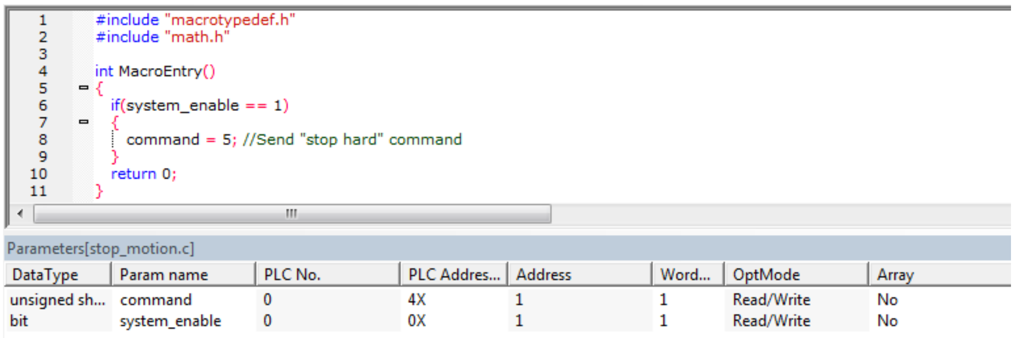
Control Type: Execute Macro Program

Marco

Macro ID: initialize.c

Execute Method: OFF -> ON

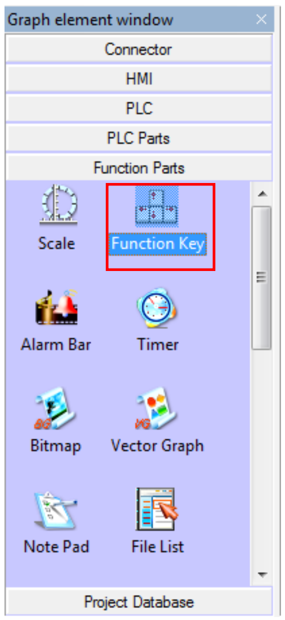
In a similiar fashion, we add a stop-motion “Function Key” as a safety feature. This “Function Key” will send the stop hard command to the PCL601 by triggering a macro. The stop hard command code is 5. Refer to page 29 for more details on command codes. Note that in this macro, and the rest of macros in this tutorial, we first check if output 1 is active, so that the system will only respond if it has been enabled by the operator.

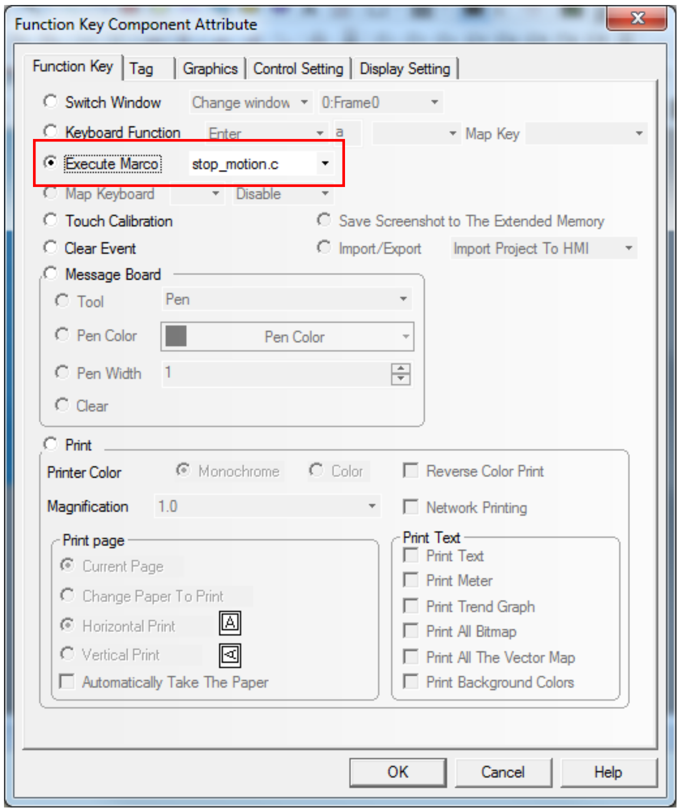


```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
    if(system_enable == 1)
    {
        command = 5; //Send "stop hard" command
    }
    return 0;
}
```

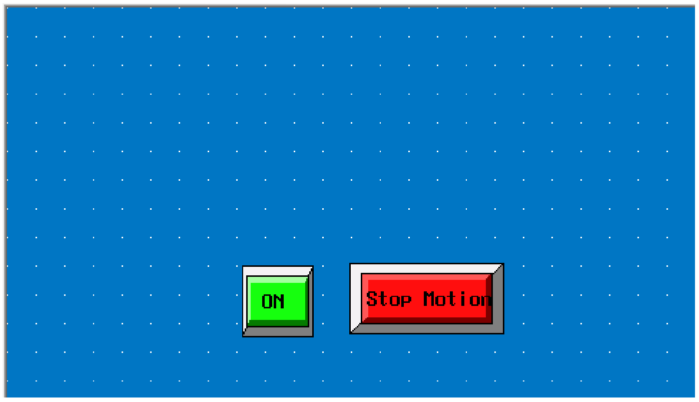
Parameters[stop_motion.c]

DataType	Param name	PLC No.	PLC Addres...	Address	Word...	OptMode	Array
unsigned sh...	command	0	4X	1	1	Read/Write	No
bit	system_enable	0	0X	1	1	Read/Write	No

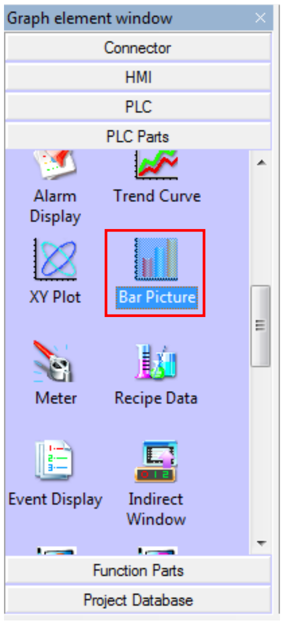


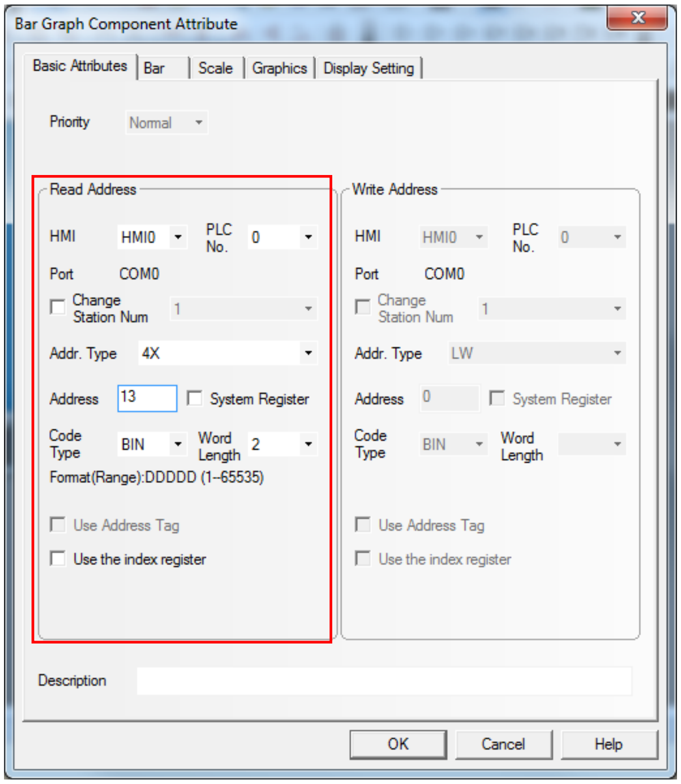


Execute Marco: stop_motion.c



Now, we add a “Bar Picture” which will represent the position of the system at any given time. We configure the bar graph to use the position register as the reference value as shown below:





Priority: Normal

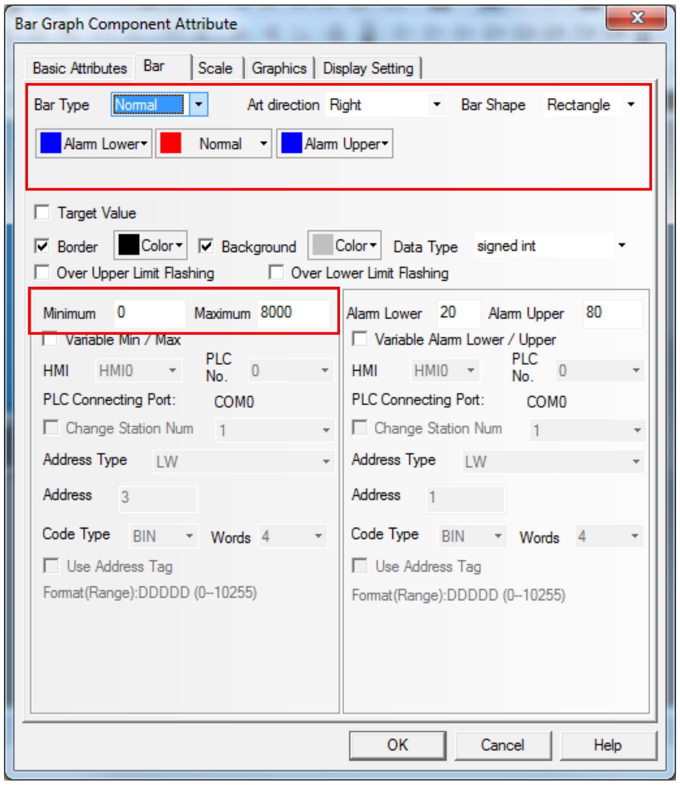
Read Address

HMI: HMI0 PLC No.: 0
Port: COM0
Change Station Num: 1
Addr. Type: 4X
Address: 13 System Register
Code Type: BIN Word Length: 2
Format(Range):DDDDD (1--65535)
Use Address Tag
Use the index register

Write Address

HMI: HMI0 PLC No.: 0
Port: COM0
Change Station Num: 1
Addr. Type: LW
Address: 0 System Register
Code Type: BIN Word Length:
Use Address Tag
Use the index register

Description:


Bar Graph Component Attribute

Basic Attributes | Bar | Scale | Graphics | Display Setting

Bar Type: Normal — Art direction: Right — Bar Shape: Rectangle

Alarm Lower — Normal — Alarm Upper

☐ Target Value

☑ Border — Color ☑ Background — Color — Data Type: signed int

☐ Over Upper Limit Flashing ☐ Over Lower Limit Flashing

Minimum	0	Maximum	8000

☐ Variable Min / Max

HMI: HMI0 — PLC No.: 0

PLC Connecting Port: COM0

☐ Change Station Num: 1

Address Type: LW

Address: 3

Code Type: BIN — Words: 4

☐ Use Address Tag

Format(Range):DDDDD (0--10255)

Alarm Lower	20	Alarm Upper	80

☐ Variable Alarm Lower / Upper

HMI: HMI0 — PLC No.: 0

PLC Connecting Port: COM0

☐ Change Station Num: 1

Address Type: LW

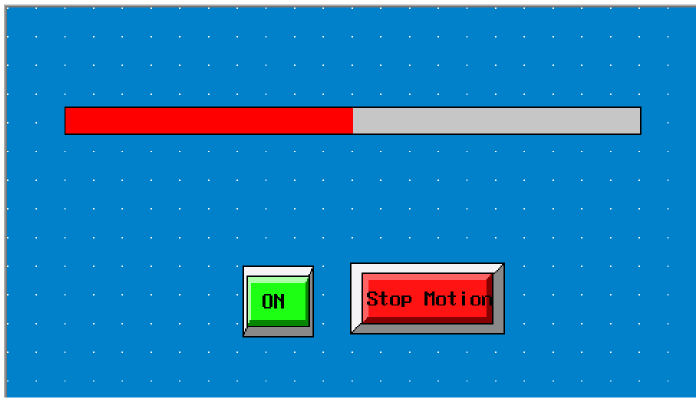
Address: 1

Code Type: BIN — Words: 4

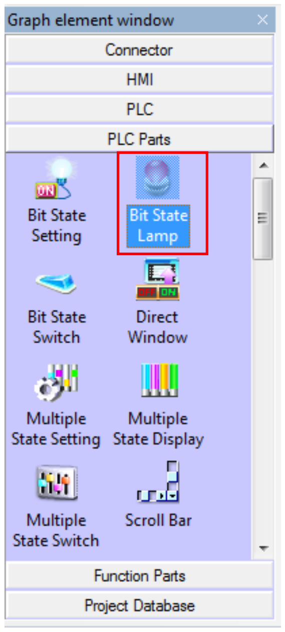
☐ Use Address Tag

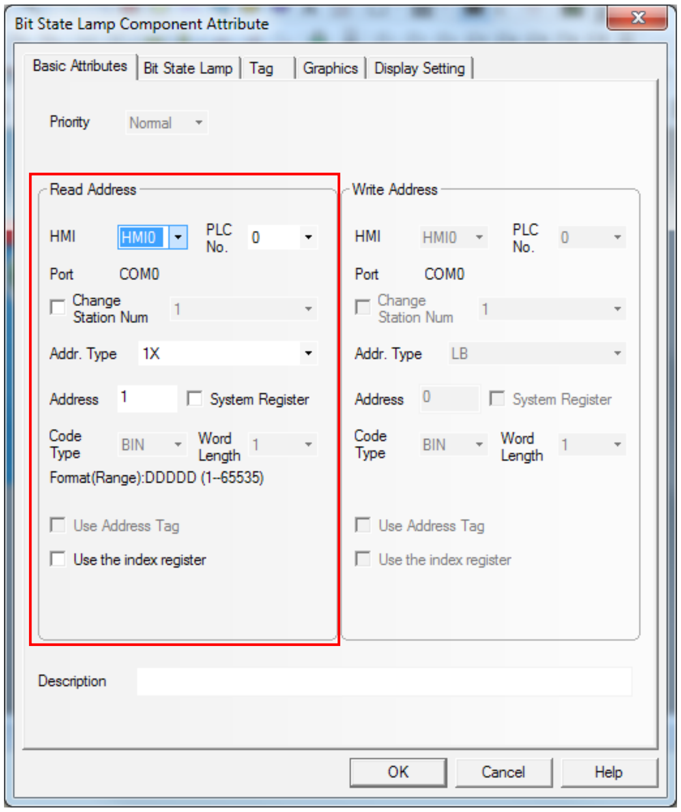
Format(Range):DDDDD (0--10255)

OK — Cancel — Help



Next, we will add three “Bit State Lamp” objects to represent the three stages and to visualize the completion of each stage. Each bit state lamp is configured to represent an input on the PCL601; these inputs are connected to the lines that will signal the end of each stage. Inputs 1, 2 and 3 are used in this tutorial. Note that we select address type 1x, which is the reference address type for inputs in Modbus.




Bit State Lamp Component Attribute

Basic Attributes | Bit State Lamp | Tag | Graphics | Display Setting

Priority Normal

Read Address

HMI HMI0 PLC No. 0

Port COM0

☐ Change Station Num 1

Addr. Type 1X

Address 1 ☐ System Register

Code Type BIN Word Length 1

Format(Range):DDDDD (1~65535)

☐ Use Address Tag

☐ Use the index register

Write Address

HMI HMI0 PLC No. 0

Port COM0

☐ Change Station Num 1

Addr. Type LB

Address 0 ☐ System Register

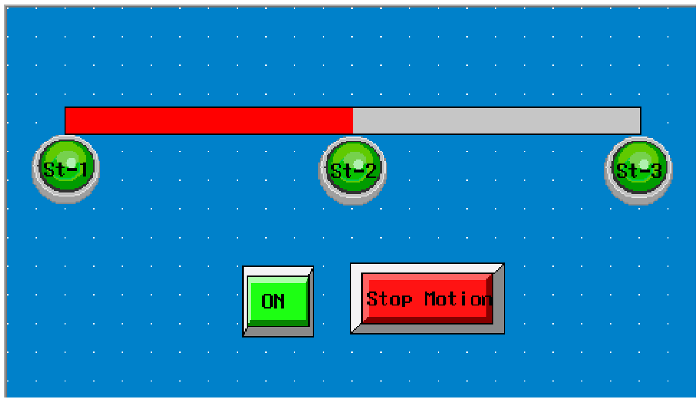
Code Type BIN Word Length 1

☐ Use Address Tag

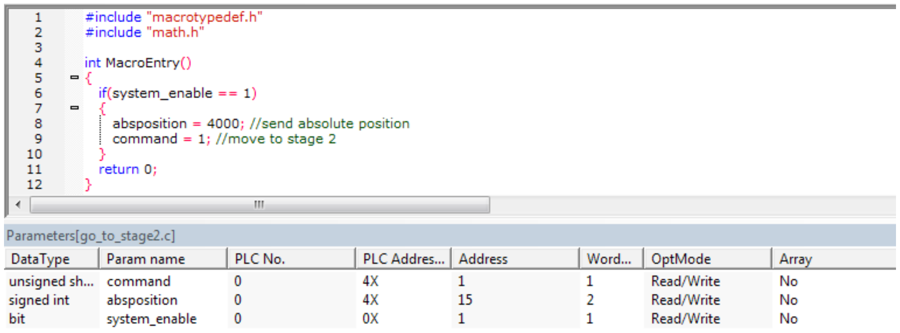
☐ Use the index register

Description

OK Cancel Help



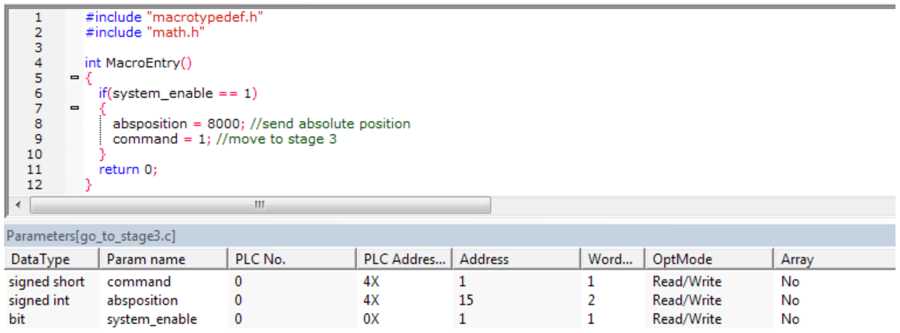
In addition to being able to helping us visualize the completion of each stage, we want this input pulse signal to trigger the motion of the system. In order to do this, we will create three separate macros, and each of these will be triggered by a “PLC Control” event, which will be the detection of the respective pulse.


```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
  if(system_enable == 1)
  {
    absposition = 4000; //send absolute position
    command = 1; //move to stage 2
  }
  return 0;
}
```

Parameters[go_to_stage2.c]

DataType	Param name	PLC No.	PLC Addres...	Address	Word...	OptMode	Array
unsigned sh...	command	0	4X	1	1	Read/Write	No
signed int	absposition	0	4X	15	2	Read/Write	No
bit	system_enable	0	0X	1	1	Read/Write	No

Macro triggered by input 1 at stage 1

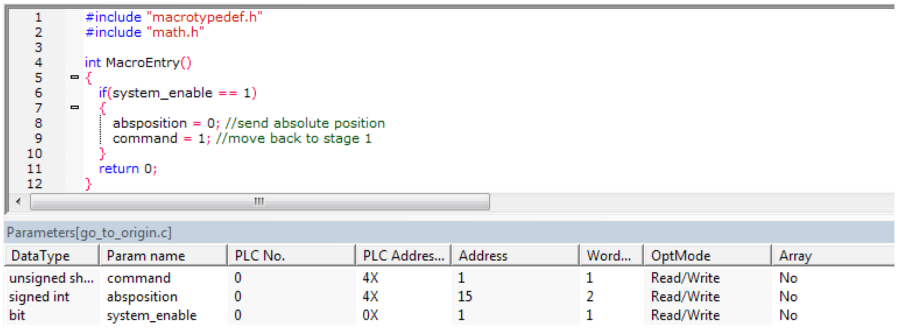


```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
  if(system_enable == 1)
  {
    absposition = 8000; //send absolute position
    command = 1; //move to stage 3
  }
  return 0;
}
```

Parameters[go_to_stage3.c]

DataType	Param name	PLC No.	PLC Addres...	Address	Word...	OptMode	Array
signed short	command	0	4X	1	1	Read/Write	No
signed int	absposition	0	4X	15	2	Read/Write	No
bit	system_enable	0	0X	1	1	Read/Write	No

Macro triggered by input 2 at stage 2



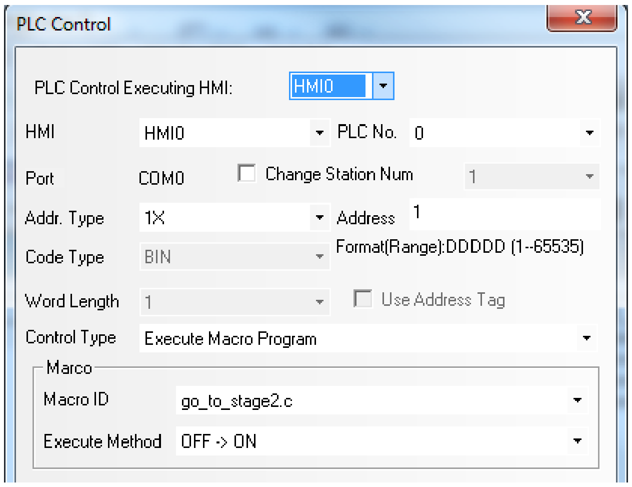
```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
  if(system_enable == 1)
  {
    absposition = 0; //send absolute position
    command = 1; //move back to stage 1
  }
  return 0;
}
```

Parameters[go_to_origin.c]

DataType	Param name	PLC No.	PLC Addres...	Address	Word...	OptMode	Array
unsigned sh...	command	0	4X	1	1	Read/Write	No
signed int	absposition	0	4X	15	2	Read/Write	No
bit	system_enable	0	0X	1	1	Read/Write	No

Macro triggered by input 3 at stage 3

Once the macros are written, we can add three separate "PLC Control" events to trigger them. The configuration shown below is done for each input corresponding to the respective macro. Also, we arbitrarily chose the execute method to be "OFF->ON", so that the macro executes on the rising edge of the pulse. It could as well be done on the falling edge of the pulse if the width of the pulse is negligible.
Note that the PCL601 has active low inputs, so in this tutorial we assume that the complement of the pulse is available.



PLC Control

PLC Control Executing HMI: HMI0

HMI: HMI0 PLC No. 0

Port: COM0 ☐ Change Station Num 1

Addr. Type: 1X Address: 1

Format(Range):DDDDD (1--65535)

Code Type: BIN

Word Length: 1 ☐ Use Address Tag

Control Type: Execute Macro Program

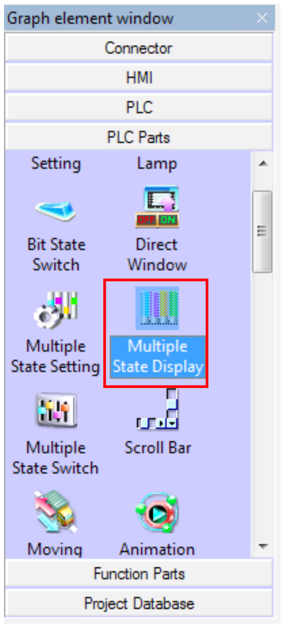
Marco

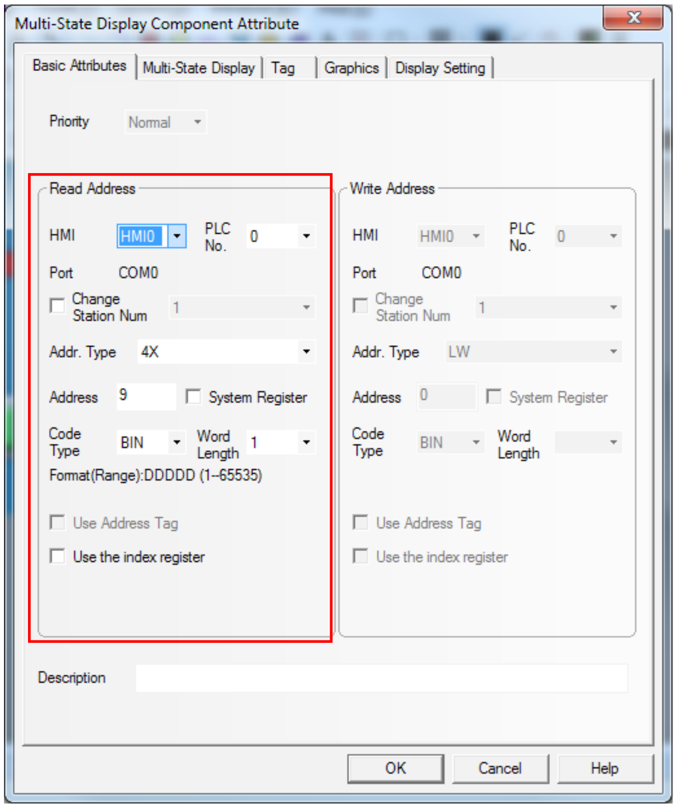
Macro ID: go_to_stage2.c

Execute Method: OFF -> ON

Finally, we will include two added features to the interface: two “Multiple State Display” objects to show the direction in which the system is moving, one for right and another for left, as well as a “number display” object to display the exact position of the system at any given time:

We start by adding the “Multiple State Display” objects. Note that the PCL601 has a direction register as opposed to a direction bit. This is the reason we use a “Multiple State Display” instead of a “Bit State Lamp”.





Priority: Normal

Read Address

HMI	HMI0
PLC No.	0
Port	COM0
Change Station Num	1
Addr. Type	4X
Address	9
System Register	☐
Code Type	BIN
Word Length	1

Format(Range):DDDDD (1--65535)

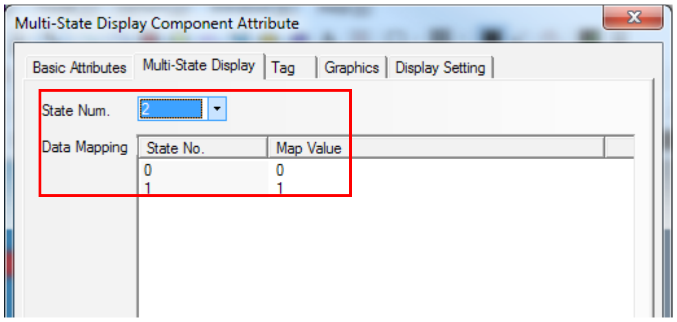
☐ Use Address Tag
☐ Use the index register

Write Address

HMI	HMI0
PLC No.	0
Port	COM0
Change Station Num	1
Addr. Type	LW
Address	0
System Register	☐
Code Type	BIN
Word Length	

☐ Use Address Tag
☐ Use the index register

Description

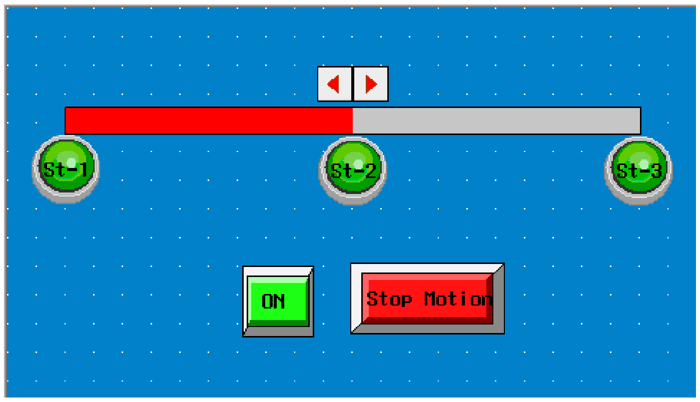


State Num. 2

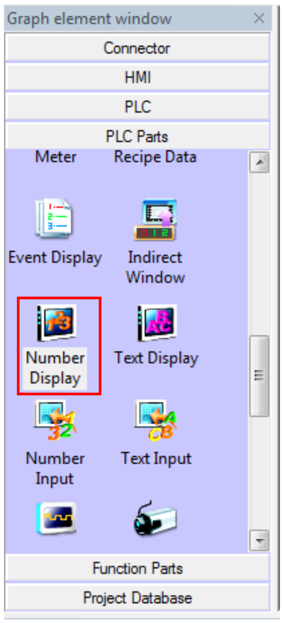
Data Mapping

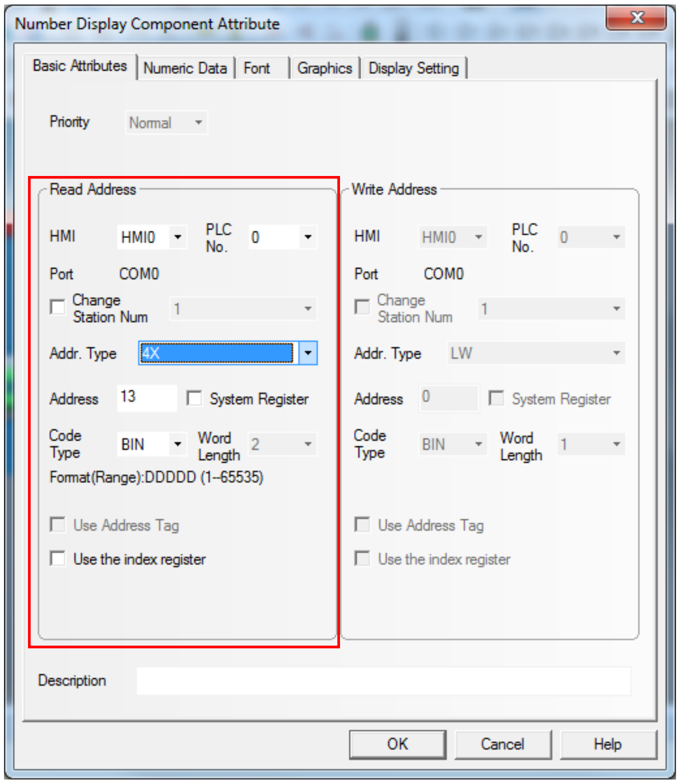
State No.	Map Value
0	0
1	1

We repeat this configuration for the second “multiple state display” object with the only difference that we reverse the values that we map our two states to; this way, one of them will be active when the system is moving right, and the other when the system is moving left.



Then, we add the “Number Display”, which will display the position of the system.





Number Display Component Attribute

Basic Attributes | Numeric Data | Font | Graphics | Display Setting

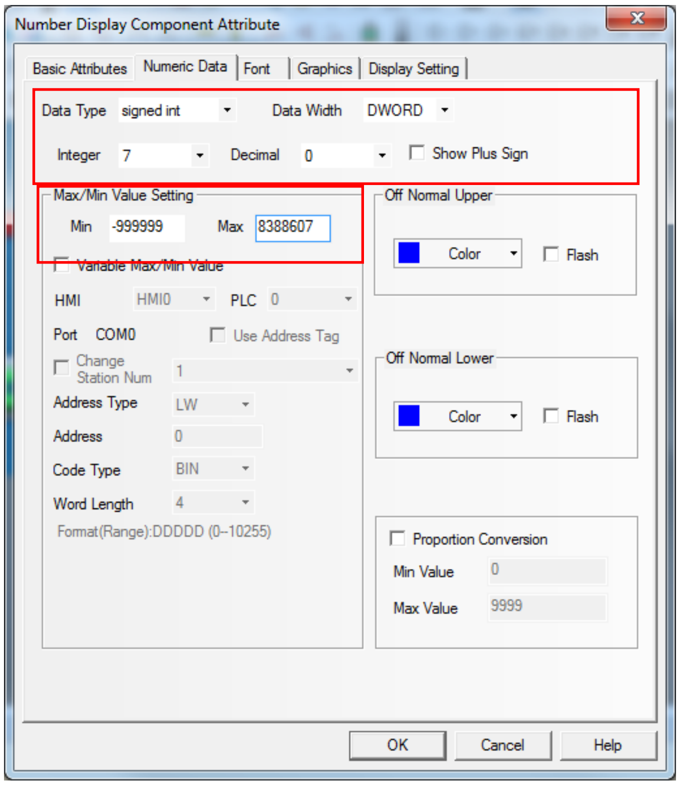
Priority: Normal

Read Address
- HMI: HMI0 PLC No.: 0
- Port: COM0
- Change Station Num: 1
- Addr. Type: 4X
- Address: 13 System Register
- Code Type: BIN Word Length: 2
- Format(Range):DDDDD (1--65535)
- Use Address Tag
- Use the index register

Write Address
- HMI: HMI0 PLC No.: 0
- Port: COM0
- Change Station Num: 1
- Addr. Type: LW
- Address: 0 System Register
- Code Type: BIN Word Length: 1
- Use Address Tag
- Use the index register

Description

OK Cancel Help


Number Display Component Attribute

Basic Attributes | Numeric Data | Font | Graphics | Display Setting

Data Type: signed int — Data Width: DWORD

Integer: 7 — Decimal: 0 — ☐ Show Plus Sign

Max/Min Value Setting
Min: -999999 — Max: 8388607
☐ Variable Max/Min Value

HMI: HMI0 — PLC: 0
Port: COM0 — ☐ Use Address Tag
☐ Change Station Num: 1
Address Type: LW
Address: 0
Code Type: BIN
Word Length: 4
Format(Range):DDDDD (0--10255)

Off Normal Upper
🟦 Color — ☐ Flash

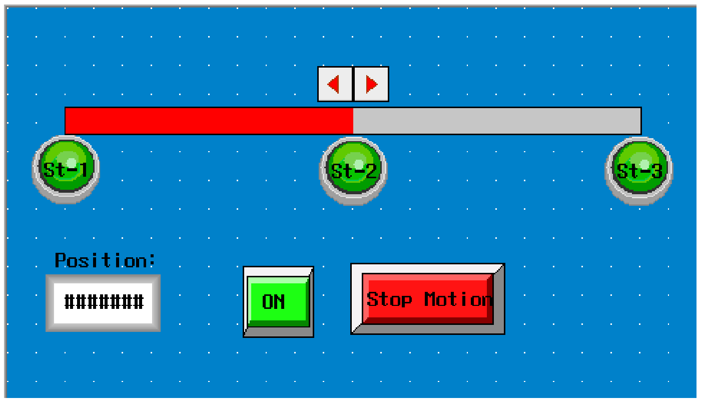
Off Normal Lower
🟦 Color — ☐ Flash

☐ Proportion Conversion
Min Value: 0
Max Value: 9999

OK | Cancel | Help

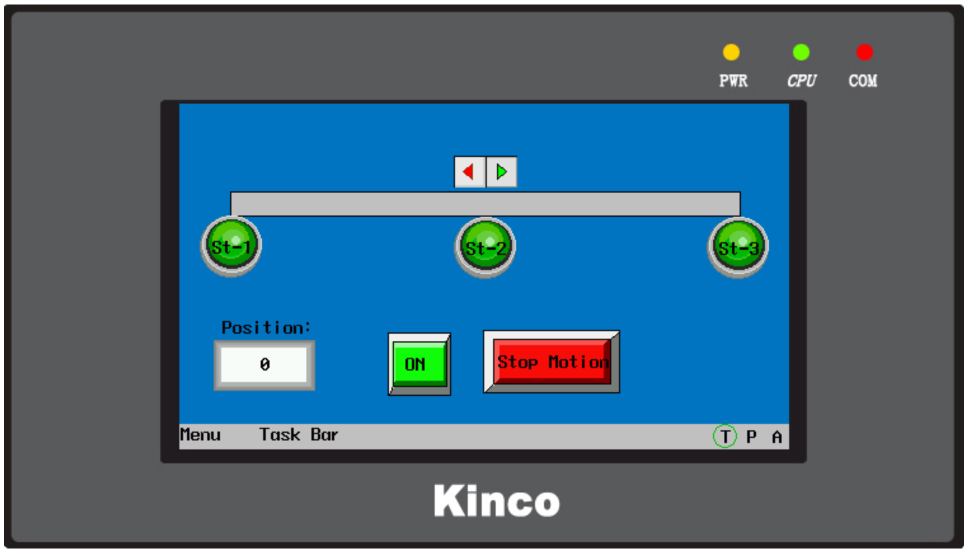
We configure the "Numeric Data" tab to display position as a double word since the position variable is mapped as two words in the PCL601.

Then, we add a textbox above the "Number Display" object to label it as "Position"

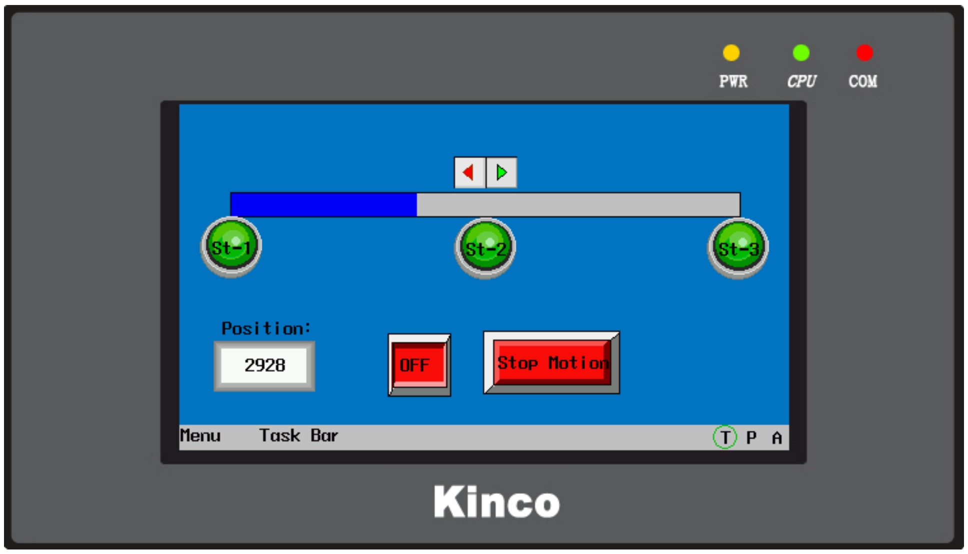


That completes the interface for our application. All we have to do now is compile the project, fix any errors we may have, and test it.

Note that, in order to connect the HMI to the PCL601, a straight through RS-232 DB9 connector can be used to accompish this (i.e. pins 2, 3, and 5 one the PCL601MOD connect to pins 2, 3, and 5 on the HMI respectively).



Interface on standby



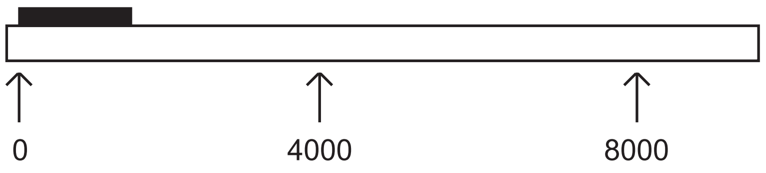
Interface during operation

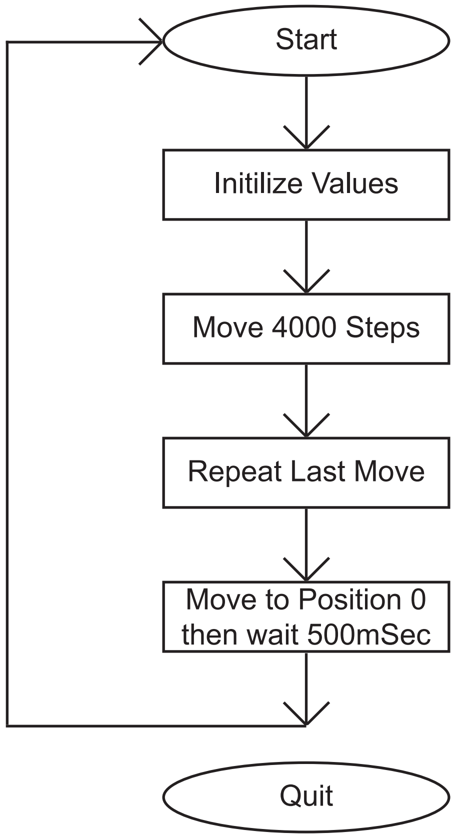
Sample Programs using the SMC60WIN software

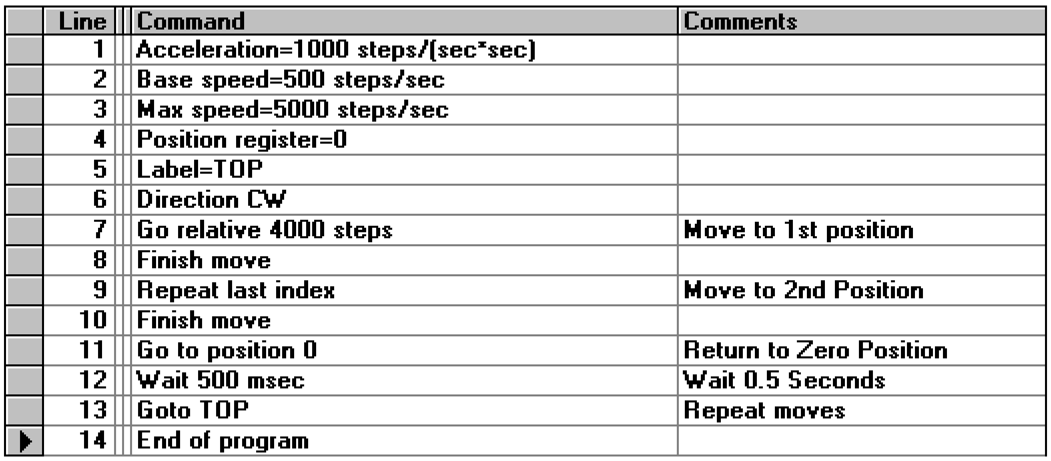
These sample programs can be created using the SMC60WIN software described in Section 3.

Sample Program 1:

Sample Program 1 illustrates a typical application where a system moves to a specific position required. The sample program shows how to use the motion and goto instruction commands.



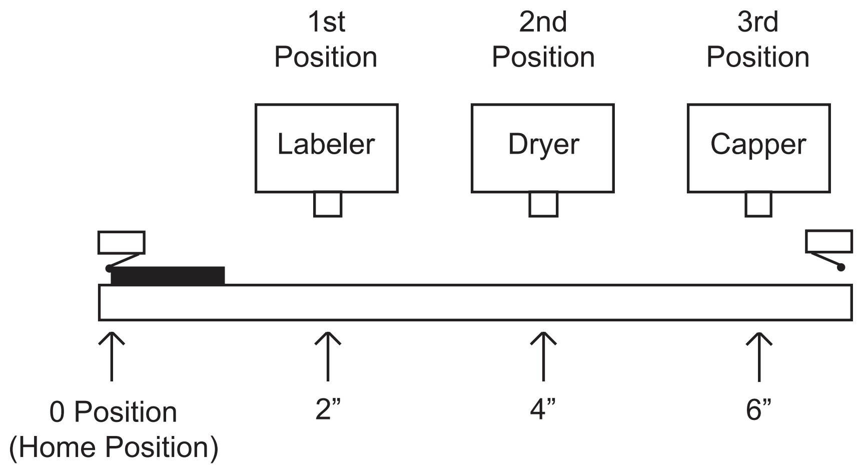


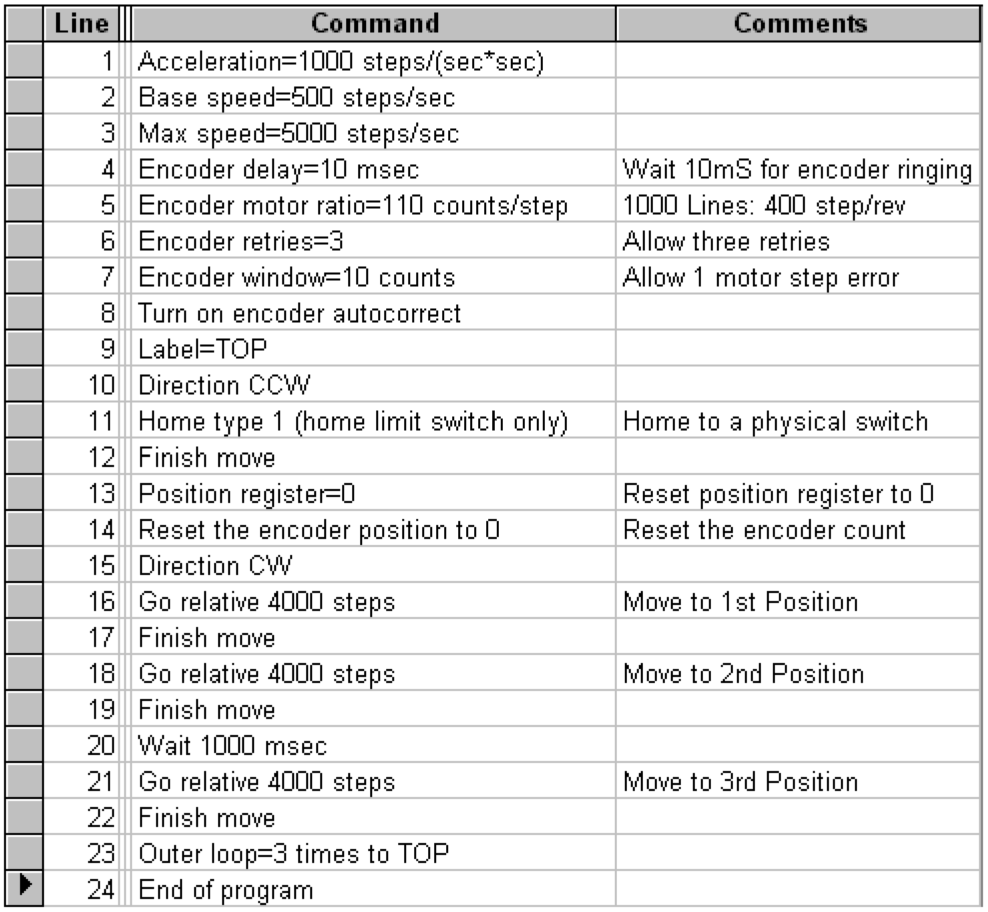


	Line	Command	Comments
	1	Acceleration=1000 steps/(sec*sec)	
	2	Base speed=500 steps/sec	
	3	Max speed=5000 steps/sec	
	4	Position register=0	
	5	Label=TOP	
	6	Direction CW	
	7	Go relative 4000 steps	Move to 1st position
	8	Finish move	
	9	Repeat last index	Move to 2nd Position
	10	Finish move	
	11	Go to position 0	Return to Zero Position
	12	Wait 500 msec	Wait 0.5 Seconds
	13	Goto TOP	Repeat moves
▶	14	End of program	

Sample Program 2:

Sample Program 2 illustrates a typical application where a system is first sent home to a datum or 0 position. This sample program shows how a motor will move to 3 different positions utilizing some of the motion commands, loop routines and encoder routine.

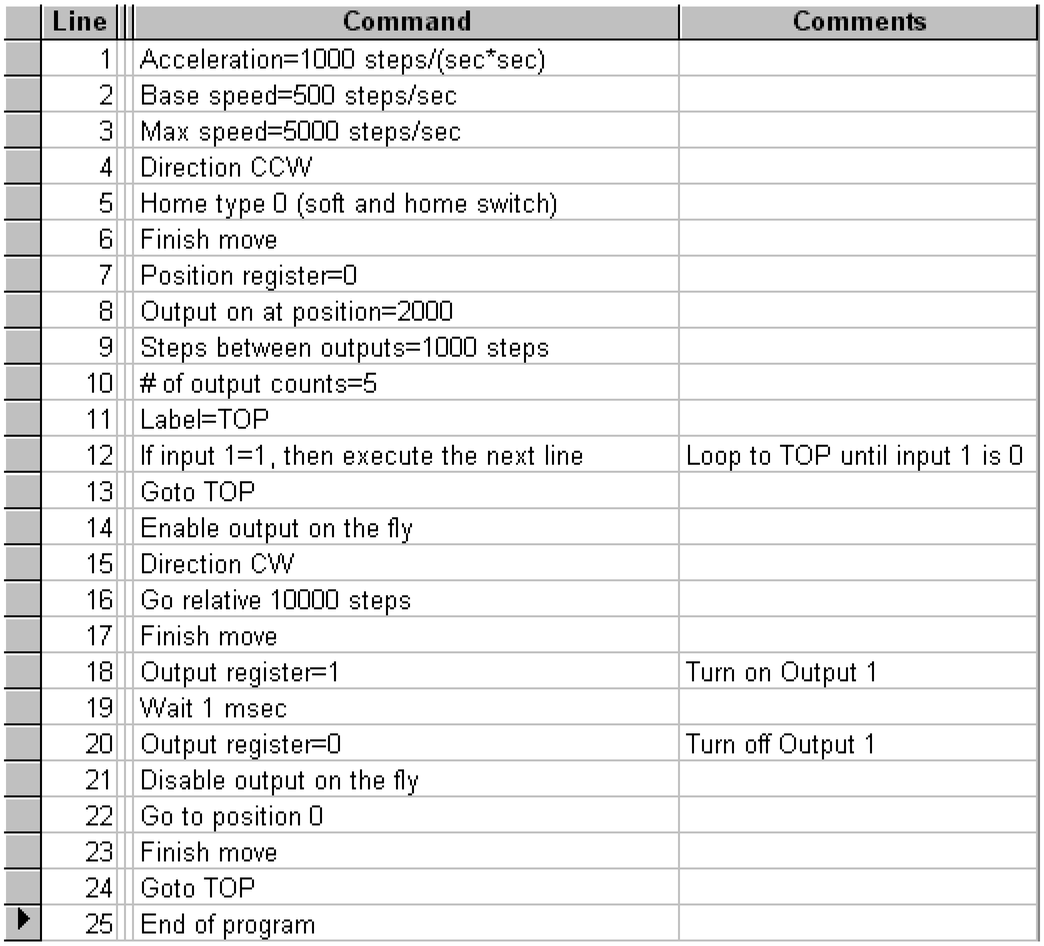




	Line	Command	Comments
	1	Acceleration=1000 steps/(sec*sec)	
	2	Base speed=500 steps/sec	
	3	Max speed=5000 steps/sec	
	4	Encoder delay=10 msec	Wait 10mS for encoder ringing
	5	Encoder motor ratio=110 counts/step	1000 Lines: 400 step/rev
	6	Encoder retries=3	Allow three retries
	7	Encoder window=10 counts	Allow 1 motor step error
	8	Turn on encoder autocorrect	
	9	Label=TOP	
	10	Direction CCW	
	11	Home type 1 (home limit switch only)	Home to a physical switch
	12	Finish move	
	13	Position register=0	Reset position register to 0
	14	Reset the encoder position to 0	Reset the encoder count
	15	Direction CW	
	16	Go relative 4000 steps	Move to 1st Position
	17	Finish move	
	18	Go relative 4000 steps	Move to 2nd Position
	19	Finish move	
	20	Wait 1000 msec	
	21	Go relative 4000 steps	Move to 3rd Position
	22	Finish move	
	23	Outer loop=3 times to TOP	
▶	24	End of program	

Sample Program 3:

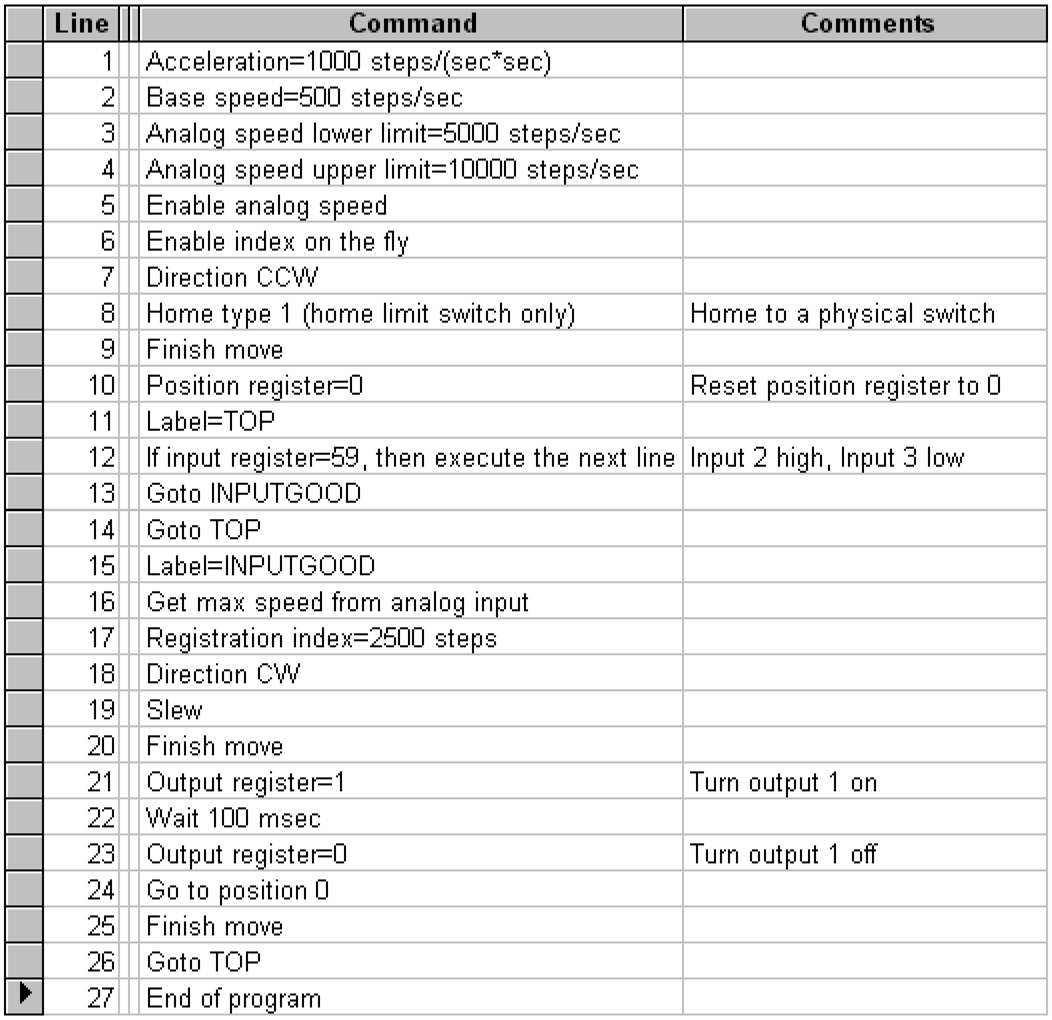
Sample Program 3 illustrates the setup and operation of the output on the fly function, and the use of the if/then statement. The system is first homed using home type 0, waits for input 1 to be a value of 0 (grounded) and then is indexed 10,000 steps. During this index, output 1 needs to be turned on every 1000 steps 5 times starting at position 2000. At the end of the index, output 1 is then turned on again for 1 mS and the unit is then sent back the 10,000 steps to position 0, but the output should not be turned on. The unit then repeats waiting for input 1 to be 0 and indexes again.



Line	Command	Comments
1	Acceleration=1000 steps/(sec*sec)	
2	Base speed=500 steps/sec	
3	Max speed=5000 steps/sec	
4	Direction CCW	
5	Home type 0 (soft and home switch)	
6	Finish move	
7	Position register=0	
8	Output on at position=2000	
9	Steps between outputs=1000 steps	
10	# of output counts=5	
11	Label=TOP	
12	If input 1=1, then execute the next line	Loop to TOP until input 1 is 0
13	Goto TOP	
14	Enable output on the fly	
15	Direction CW	
16	Go relative 10000 steps	
17	Finish move	
18	Output register=1	Turn on Output 1
19	Wait 1 msec	
20	Output register=0	Turn off Output 1
21	Disable output on the fly	
22	Go to position 0	
23	Finish move	
24	Goto TOP	
25	End of program	

Sample Program 4:

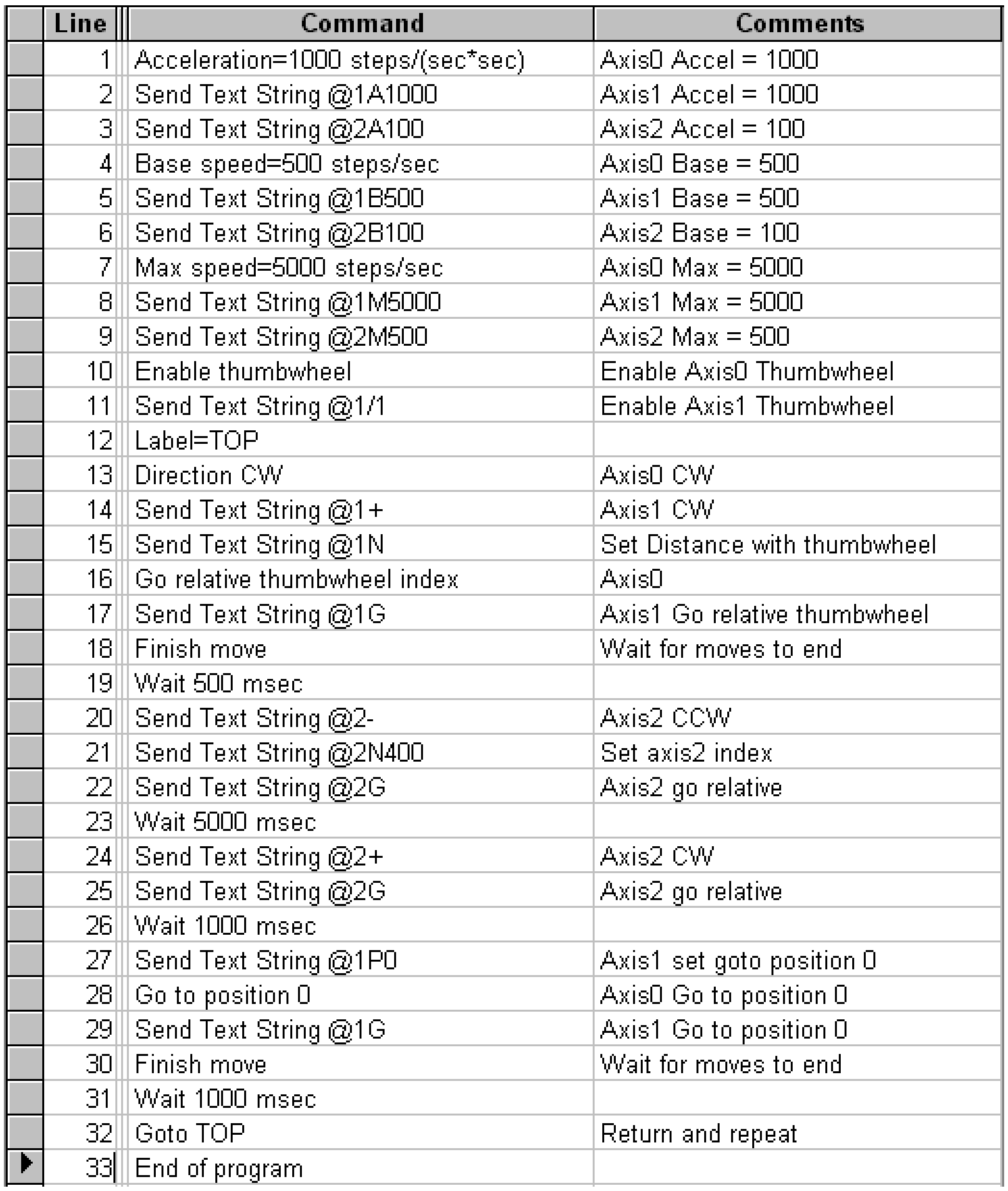
Sample Program 4 illustrates the setup of the analog speed function and the use of “indexing-on-the-fly.” The system is first homed using home type 1. The next step is to wait for the input register to read 110111 (input 2 must be high while input 3 is low, all other inputs are not used and input 1 is masked high due to the analog function being used). The third step is to slew using the analog input as the maximum speed between 5000 and 10000 steps/revolution. When input 2 is switched low, the unit will index 2500 more steps and ramp down to base speed and stop. At the end of the index, output 1 will turn on for 100 mS. After the output is turned off the unit will be sent back to position 0. The program then is sent back to repeat itself, waiting for the input register to be 110111 again.



Line	Command	Comments
1	Acceleration=1000 steps/(sec*sec)	
2	Base speed=500 steps/sec	
3	Analog speed lower limit=5000 steps/sec	
4	Analog speed upper limit=10000 steps/sec	
5	Enable analog speed	
6	Enable index on the fly	
7	Direction CCW	
8	Home type 1 (home limit switch only)	Home to a physical switch
9	Finish move	
10	Position register=0	Reset position register to 0
11	Label=TOP	
12	If input register=59, then execute the next line	Input 2 high, Input 3 low
13	Goto INPUTGOOD	
14	Goto TOP	
15	Label=INPUTGOOD	
16	Get max speed from analog input	
17	Registration index=2500 steps	
18	Direction CW	
19	Slew	
20	Finish move	
21	Output register=1	Turn output 1 on
22	Wait 100 msec	
23	Output register=0	Turn output 1 off
24	Go to position 0	
25	Finish move	
26	Goto TOP	
27	End of program	

Sample Program 5:

Sample Program 5 illustrates a typical 3 axis application where one PCL601 (Axis 0) is controlling the other two axes (Axis 1 and Axis 2) by using the send text string commands. Note that, because the send text command is being used, this sample program is intended to run with the device configured to use the AA ASCII protocol. The program first sets the accelerations, base speeds and maximum speeds for each axis. It then is enabling Axis 0 and Axis 1 to use the thumbwheel switch that is connected to each unit for indexing. The direction is then set for Axis 0 and Axis 1 and these two axes are then told to index the distance set in each thumbwheel at the same time. The program then waits for the motion to finish before it tells Axis 2 to move. Axis 2 first moves in the negative direction waits for the index to finish, and then moves back in the positive direction. After Axis 2 is finished moving, Axis 0 and Axis 1 are both sent back to their zero positions at the same time. The program is then repeated.



Line	Command	Comments
1	Acceleration=1000 steps/(sec*sec)	Axis0 Accel = 1000
2	Send Text String @1A1000	Axis1 Accel = 1000
3	Send Text String @2A100	Axis2 Accel = 100
4	Base speed=500 steps/sec	Axis0 Base = 500
5	Send Text String @1B500	Axis1 Base = 500
6	Send Text String @2B100	Axis2 Base = 100
7	Max speed=5000 steps/sec	Axis0 Max = 5000
8	Send Text String @1M5000	Axis1 Max = 5000
9	Send Text String @2M500	Axis2 Max = 500
10	Enable thumbwheel	Enable Axis0 Thumbwheel
11	Send Text String @1/1	Enable Axis1 Thumbwheel
12	Label=TOP	
13	Direction CW	Axis0 CW
14	Send Text String @1+	Axis1 CW
15	Send Text String @1N	Set Distance with thumbwheel
16	Go relative thumbwheel index	Axis0
17	Send Text String @1G	Axis1 Go relative thumbwheel
18	Finish move	Wait for moves to end
19	Wait 500 msec	
20	Send Text String @2-	Axis2 CCW
21	Send Text String @2N400	Set axis2 index
22	Send Text String @2G	Axis2 go relative
23	Wait 5000 msec	
24	Send Text String @2+	Axis2 CW
25	Send Text String @2G	Axis2 go relative
26	Wait 1000 msec	
27	Send Text String @1P0	Axis1 set goto position 0
28	Go to position 0	Axis0 Go to position 0
29	Send Text String @1G	Axis1 Go to position 0
30	Finish move	Wait for moves to end
31	Wait 1000 msec	
32	Goto TOP	Return and repeat
33	End of program	

Appendix 1: PCL601’s Memory Map

Data/Holding Registers

The PCL601’s motion parameters and data are mapped to 38 registers; these are referenced to as 4x.

Address (4x)	Access	Description	Range
1	Read/Write	Command Register	1 to 15
2	Read/Write	Number of Steps (Low word)	0 to 8,388,607
3	Read/Write	Number of Steps (High word)	
4	Read/Write	Acceleration (Low word)	100 to 9,999,999
5	Read/Write	Acceleration (High word)	
6	Read/Write	Base Speed	1 to 5,000
7	Read/Write	Maximum Speed	1 to 50,000
8	Read/Write	Jog Speed	1 to 50,000
9	Read/Write	Direction (CW = 1, CCW = 0)	0 or 1
10	Read/Write	Device Address	1 to 99
11	Read/Write	Analog Enable	0 or 1
12	Read/Write	Motor Current Enabled	0 or 1
13	Read/Write	Current Position (Low word)	-8,388,607 to 8,388,607
14	Read/Write	Current Position (High word)	
15	Read/Write	Absolute position (Low word)	-8,388,607 to 8,388,607
16	Read/Write	Absolute position (High word)	
17	Read/Write	Analog speed lower limit	1 to 50,000
18	Read/Write	Analog speed upper limit	1 to 50,000
19	Read/Write	Analog position lower limit	0 to 65,535
20	Read/Write	Analog position upper limit	0 to 65,535
21	Read/Write	Index on the fly enable	0 or 1
22	Read/Write	Thumbwheel index enable	0 or 1
23	Read/Write	Output on the fly enable	0 or 1
24	Read/Write	First output on the fly position	0 to 65,535
25	Read/Write	Steps between outputs on the fly	0 to 65,535
26	Read/Write	Number of outputs on the fly	0 to 255
27	Read/Write	Encoder delay	0 to 65,535
28	Read/Write	Encoder window	0 to 255
29	Read/Write	Encoder ratio	1 to 255
30	Read/Write	Encoder autocorrect enable	0 or 1
31	Read/Write	Encoder retries	0 to 255
32	Read only	Encoder position (Low word)	-8,388,607 to 8,388,607
33	Read only	Encoder position (High word)	
34	Read only	Analog index	0 - 9,999,999
35	Read only	Busy	0 or 1
36	Read only	Error code register	2^{0-13}
37	Read only	Firmware version (Low word)	Fixed
38	Read only	Firmware version (High word)	Fixed

Output Registers

The PCL601 has a total of eight outputs, and these are mapped to a single 16-bit register as shown here. Ouput registers are referenced to as 0x.

Address (0x)	Access	Description	Range
1	Read/Write	Outputs 1 - 8	1 - 255

The output register holds the status of the outputs in the following arrangement:

bits 16 - 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	output 8	output 7	output 6	output 5	output 4	output 3	output 2	output 1

Individual Inputs

The PCL601 has a set of six user programmable inputs in addition to a set of limit switch inputs. Individual inputs are referenced to as 1x.

Address (1x)	Access	Description
1	Read	Input 1(1)
2	Read	Input 2
3	Read	Input 3
4	Read	Input 4
5	Read	input 5
6	Read	input 6
7	Read	Hard Limit
8	Read	Soft Limit

(1) Note that input 1 can also be configured as an analog input; in that case, input one will read as active at all times.

Input Registers

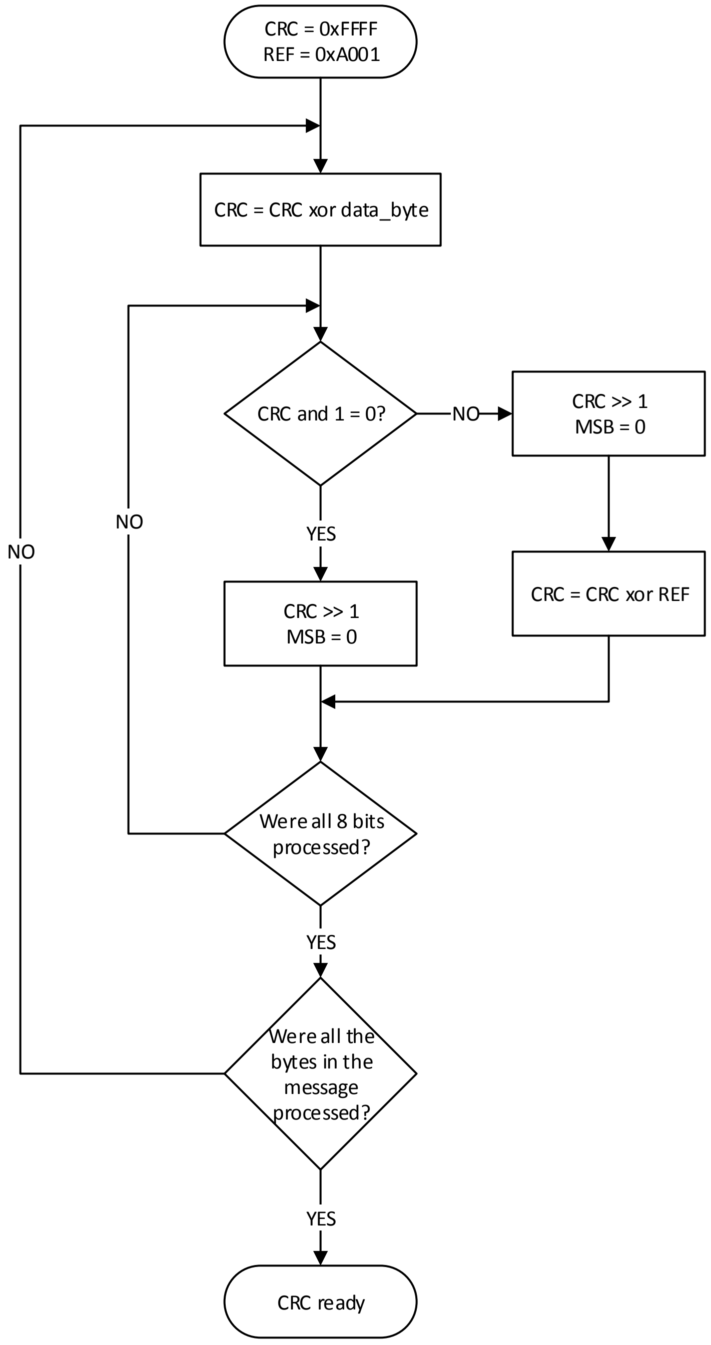
The PCL601 has a set of six user programmable inputs in addition to a set of limit switch inputs. These are mapped to a single 16-bit register. Input registers are referenced to as 3x.

Address (3x)	Access	Description
1	Read	Inputs 1 - 6 and Hard and Soft Limits

The input register holds the status of the outputs in the following arrangement:

bits 16 - 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Soft Limit	Hard Limit	input 6	input 5	input 4	input 3	input 2	input 1

Appendix 2: CRC Generation Algorithm



The Cyclical Redundancy Check is a 16-bit number calculated by the device, after receiving and before transmitting a Modbus message, using the procedure outlined above. The 16-bit reference value used is the hexadecimal number 0xA001. The CRC value is transmitted as the last two bytes of the message with the low byte byte being transmitted before the high byte.

CRC >> 1:	CRC = CRC shifted once to the right
MSB:	Most significant bit

Appendix 3: Supported Modbus Function Codes

The Modbus protocol provides a wide range of functions as published by Modicon's Modbus Protocol Reference Guide. This device, however, only implements the standard read and write functions that are commonly used between devices that use the this protocol. The functions supported by this device are described below.

0x01 - Read Outputs

This function reads the status of the device's discrete outputs. This function is executable at all times.

Example: The following example shows the request and response for reading a sequence of five outputs, starting with output one (address 0x0000), from the device. The response yields the status of the outputs to be 0x03, emaning outputs 1 and 2 are ON, and outputs 3, 4, and 5 are OFF.

Request:	Address	Function	Starting address		No. of outputs		CRC	
	0x01	0x01	0x00	0x00	0x00	0x05	0xFC	0x09

Response:	Address	Function	Byte count	Output data	CRC	
	0x01	0x01	0x01	0x03	0x11	0x89

0x02 - Read Inputs

This function reads the status of the device's discrete inputs. This function is executable at all times.

Example: The following shows the request and response for reading a sequence of four inputs, starting with input one (adress 0x0000), from the device. The response returns the value 0x04, meaning that inputs 1, 2 and 4 are OFF, and input 3 is ON.

Request:	Address	Function	Starting address		No. of inputs		CRC	
	0x01	0x02	0x00	0x00	0x00	0x04	0x79	0xC9

Response:	Address	Function	Byte count	Input data	CRC	
	0x01	0x02	0x01	0x04	0xA0	0x4B

0x03 - Read Data Registers

This function reads the binary contents of the data registers in the device. This function is executable at all times.

Example: The following shows the request and response for reading three sequential data registers starting at register number three (address 0x0002). The response returns the value of the three registers with the bytes in each 16-bit word being arranged from high to low as they are stored in the device.

Request:	Address	Function	Starting address		No. of registers		CRC	
	0x01	0x03	0x00	0x02	0x00	0x03	0xA4	0x0B

Response:	Address	Function	Byte count	Reg. 3		Reg. 4		Reg. 5		CRC	
	0x01	0x03	0x06	0x00	0x00	0x27	0x10	0x00	0x00	0x2A	0x04

0x04 - Read Input Registers

This function reads the contents of the input registers in the device. Note that the PCL601 only has one 16-bit input register. This function is executable at all times.

Example: The following shows the request and response for reading input register number one (address 0x0000). The response returns the current 16-bit value of the regster as shown below. In this instance the value returned is 0x0010 (binary: 0000_0000_0001_0000), which tells us the only active input is input 5.

Request:	Address	Function	Starting address		No. of registers		CRC	
	0x01	0x04	0x00	0x00	0x00	0x01	0x31	0xCA

Response:	Address	Function	Byte count	Input register 1		CRC	
	0x01	0x04	0x02	0x00	0x10	0xB8	0xFC

0x05 - Write Single Output

This function forces a single output either ON or OFF. Note that the PCL601 only has one 16-bit input register. This function is not executable when the device is running a program.

Example: The following shows the request and response for setting output number 3 ON (ON = 0xFF00; OFF = 0x0000). The response is simply the echo of the request; this confirms that the request has been satisfied.

Request:	Address	Function	Output address		ON or OFF		CRC	
	0x01	0x05	0x00	0x02	0xFF	0x00	0x2D	0xFA

Response:	Address	Function	Output address		ON or OFF		CRC	
	0x01	0x05	0x00	0x02	0xFF	0x00	0x2D	0xFA

0x06 - Write Single Data Register

This function writes a 16-bit value to a single 16-bit data register. This function is not executable when the device is running a program.

Example: The following shows the request and response for writing the 16-bit value 0x07D0 (decimal 2000) to data register number seven (address 0x0006). The response is simply the echo of the request; this confirms that the request has been satisfied.

Request:	Address	Function	Register address		Data to be written		CRC	
	0x01	0x06	0x00	0x06	0x07	0xD0	0x6A	0x67

Response:	Address	Function	Register address		Data written		CRC	
	0x01	0x06	0x00	0x06	0x07	0xD0	0x6A	0x67

0x0F - Write Multiple Outputs

This function forces each inidivdual output in a sequential group of outputs either ON or OFF. This function is not executable when the device is running a program.

Example: The following shows the request and response for writing to outputs 2, 3, 4 and 5. The value being written is 0x05 (binary 0000_0101); this forces outputs 2 and 4 ON, and output 3 and 5 OFF. The response echoes all the parts of the request except for the byte count and the data; this confirms that the request has been satisfied.

Request:	Address	Function	Starting address		No. of outputs		Byte count	Data	CRC	
	0x01	0x0F	0x00	0x01	0x00	0x04	0x01	0x05	0xC3	0x55

Response:	Address	Function	Staring address		No. of outputs		CRC	
	0x01	0x0F	0x00	0x01	0x00	0x04	0x05	0xC8

0x10 - Write Multiple Registers

This function writes a sequence of 16-bit values to a sequence of 16-bit data registers. This function is not executable when the device is running a program.

Example: The following shows the request and response for writing the value 0x01F4, 0x0BB8, and 0x07D0 to data registers six, seven, and eigth respectively. The response echoes all the parts of the request except for the byte count and the value being written.

Request:	Address	Function	Starting address		No. of registers		Byte count
	0x01	0x10	0x00	0x05	0x00	0x03	0x06

Reg. 6 new value		Reg. 7 new value		Reg. 8 new value		CRC	
0x01	0xF4	0x0B	0xB8	0x07	0xD0	0xC7	0x39

Response:	Address	Function	Starting address		No. of registers		CRC	
	0x01	0x10	0x00	0x05	0x00	0x03	0x90	0x09

0x11 - Report Slave ID

This function reads the version number and the current status of the device. This function is executable at all times.

Request:	Address	Function	CRC	
	0x01	0x11	0xC0	0x2C

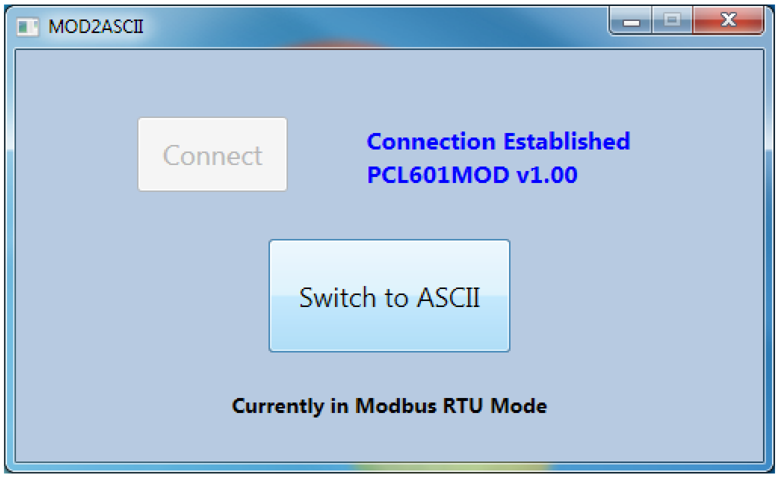
Response:	Address	Function	Byte count	Current address	Firmware version in ASCII				CRC	
	0x01	0x11	0x05	0x01	0x31	0x2E	0x30	0x30	0xC7	0xB2

Appendix 4: Switching to AA ASCII Protocol

Although this device is intended to be used mainly in the Modbus RTU protocol, the user can easily switch the device to AA ASCII mode in order to program it or simply use it in direct talk mode. In order to do this, the user can use an HMI to write the command code 11 to the command register, or the MOD2ASCII application provided with the software download can be used to easily switch protocol from a computer

MOD2ASCII

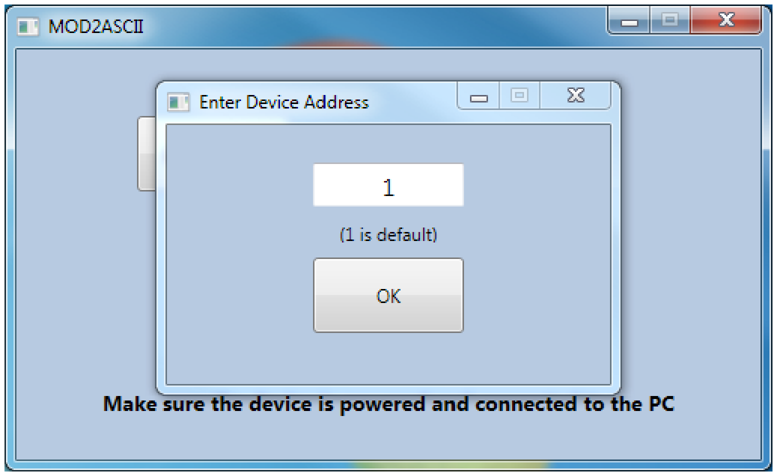
First, the user must install the application by running the setup file in the MOD2ASCII folder. Once installed, the application can be opened, and it should automatically recognize the device (assuming that the device is powered and its device address is 1).



Connection Established
PCL601MOD v1.00

Currently in Modbus RTU Mode

If the address of the device is not 1, the user will be prompted to enter the device's current address as shown below. Once the address has been entered, the program will try to connect with the device.



1

(1 is default)

Make sure the device is powered and connected to the PC

Note that this little application can also be used to switch the device back to Modbus in the same manner; however, switching back to Modbus is more easily achieved by repowering the device.

wxTerminal:

Alternatively, the user can use a simple terminal program with the proper serial port settings in order to send the command. The free wxTerminal program is provided with the software download, and it can be used to send the command.

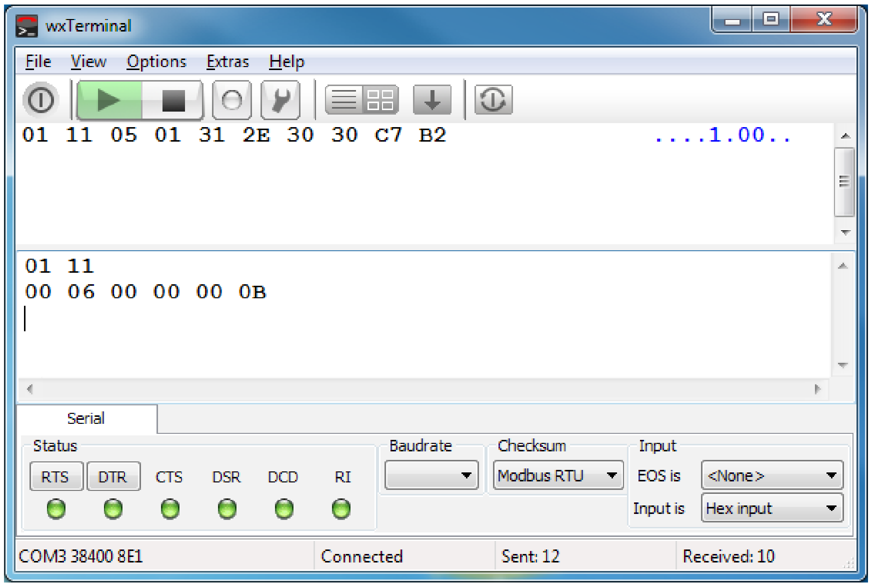
Once the program is installed and the connection settings have been configured properly, the user can connect to the deivce by going to the “File” menu on the menu bar and clicking on “Connect device”.

To verify the connection, the user can send the command 0x01 0x11 + CRC as shown below. Note that the program takes care of the CRC code, so the user can type the command with no CRC. If the connection is correct, the user should get a response that contains the version of the device, 1.00.

Now that the connection has been confirmed the user can send the switch command:

Command: 0x00 0x06 0x00 0x00 0x00 0x0B + CRC

Once again, the user can type the command without the CRC as this will be generated by the terminal program. Note that this command will yield no reponse.



```
01 11 05 01 31 2E 30 30 C7 B2          ....1.00..
```

```
01 11
00 06 00 00 00 0B
```

Baudrate:
Checksum: Modbus RTU
Input
EOS is <None>
Input is Hex input

COM3 38400 8E1 | Connected | Sent: 12 | Received: 10

Once the device has been switched to the AA ASCII protocol, it can be used in direct talk mode or be programmed using the SMC60WIN software.

Note that the device can be switched back to Modbus RTU protocol through a manual or power-on reset as well as by sending the proper ASCII command described in the AA ASCII protocol command summary.

Appendix 5: ASCII Table for Direct Mode

ASCII Symbol	Hex Value	ASCII Symbol	Hex Value	ASCII Symbol	Hex Value
0	30	J	4A	#	23
1	31	K	4B	$	24
2	32	L	4C	%	25
3	33	M	4D	“	27
4	34	N	4E	(	28
5	35	O	4F	+	2B
6	36	P	50	,	2C
7	37	Q	51	-	2D
8	38	R	52	.	2E
9	39	S	53	:	3A
A	41	T	54	;	3B
B	42	U	55	@	40
C	43	V	56	[	5B
D	44	W	57	]	5D
E	45	X	58	^	5E
F	46	Y	59	{	7B
G	47	Z	5A	|	7C
H	48	Carriage Return	0D	}	7D
I	49	!	21	~	7E

Appendix 6: Firmware Revisions

Version 1.00 - Initial Release.

Limited Warranty
All Anaheim Automation, Inc. products are warranted against defects in workmanship, materials and construction, when used under Normal Operating Conditions and when used in accordance with specifications. This warranty shall be in effect for a period of twelve months from the date of purchase or eighteen months from the date of manufacture, whichever comes first. Warranty provisions may be voided if the products are subjected to physical damage or abuse.

Anaheim Automation, Inc. will repair or replace at its option, any of its products which have been found to be defective and are within the warranty period, provided that the item is shipped freight prepaid, with RMA (return material authorization), to Anaheim Automation's plant in Anaheim, California.

ANAHEIM AUTOMATION, INC.