

## PL $\mu$ SNet II Upload/Download Program

---

<b>Description</b>	PL $\mu$ SNet II is a DOS program that will run on most IBM-PC compatible computers. When the serial port of the PC is connected to a PL $\mu$ S Programmable Limit Switch, PL $\mu$ SNet II can transfer programming values between the computer and the controller in either direction. PL $\mu$ SNet II includes its own communications software with selection of baud rate, PL $\mu$ S controller address, and the computer's COM port. No other communication software is needed.
<b>Functions</b>	PL $\mu$ SNet II provides two main functions: <b>Uploading</b> a controller's complete set of programming values from the controller to an ASCII file on the PC; and <b>downloading</b> the contents of an ASCII from a computer to the PL $\mu$ S controller. PL $\mu$ SNet II also provides a text editor to view and change the contents of an ASCII file.
<b>Applications</b>	<p><b>Hard Copy Reference</b>—Using PL<math>\mu</math>SNet II, a PL<math>\mu</math>S controller's programming can be saved as an ASCII file and printed out for reference. The printout can be used to study line operation or to program other PL<math>\mu</math>S controllers in the plant.</p> <p><b>Archival Storage</b>—The ASCII file containing a PL<math>\mu</math>S controller's programming can be stored on a hard drive or floppy disk. In the event of accidental alteration or erasure of the controller's programming, PL<math>\mu</math>SNet II can be used to download the ASCII file to the controller to restore normal operation.</p> <p><b>Programming Multiple Units</b>—If several PL<math>\mu</math>S controllers will have the same values, one controller can be programmed correctly and its setpoints uploaded to a PC using PL<math>\mu</math>SNet II. The programming can then be downloaded to the other PL<math>\mu</math>S controllers, eliminating the need to manually reenter setpoints for each controller.</p> <p><b>Modify Programming</b>—Once a program has been saved as an ASCII file, it can be studied and edited to create other versions of the program.</p>
<b>Contents</b>	The PL $\mu$ SNet II Communications Software Program includes these materials: <ul style="list-style-type: none"><li>(1) Introduction sheet.</li><li>(1) One disk containing the PLUSNET.EXE file.</li></ul>
<b>Cable</b>	To use PL $\mu$ SNet II, a serial communications cable is required to connect the PL $\mu$ S controller to an IBM compatible personal computer. This cable can be purchased from Electro Cam Corp., or it can be built by the customer using the wiring information shown in the PL $\mu$ S Programming and Installation Manual.
<b>Installation</b>	Copy the PLUSNET.EXE file to the desired directory on the PC.
<b>Operation</b>	Connect the PC and the PL $\mu$ S controller with a communications cable and turn both units ON.  Start PLUSNET.EXE from the DOS command line, or from a DOS window within Microsoft Windows. The menus in the program are self-explanatory.

## PLuSNET II Program (cont'd)

---

### Sample ASCII Program Copied from PS-5144 Using PL $\mu$ SNET II

```
2: 5144 ;Model
3: 316 ;Firmware revision
4: 17 ;Output quantity
5: 5,1 ;Option: -H; High resolution
5: 6,1 ;Option: -L; Leading/trailing speed comp
5: 7,1 ;Option: -A; Analog output
6: 1 ;Default Program
9: 1,0 ;Offset: group#, offset
9: 2,0 ;Offset: group#, offset
10: 1,0,2000 ;Analog output: Analog chn#, offset, high rpm
11: 1,10,3000 ;Motion detection: level#, low rpm, high rpm
11: 2,10,3000 ;Motion detection: level#, low rpm, high rpm
14: 0 ;Map limit
17: 0 ;Direction of increasing rotation: 0=CCW, 1=CW
18: 360 ;Scale factor
19: 0 ;Shaft offset
20: 1 ;Analog quantity
21: 0 ;Resolver type: 0=ECC, 1=Other
22: 0 ;Program select mode: 0=bin, 1=BCD, 2=Gray
25: 1,1 ;Termination resistors: grp1 on/off, grp2 on/off
27: 1,1,0,0 ;Rate setup: mpx, div, dec pt, units
28: 20 ;Toggle rpm
29: 0 ;Rpm update rate: 0=1/Sec, 1=2/Sec, 2=10/Sec
30: 1 ;Speed comp mode: 0=Single, 1=L/T
31: 0 ;Group pos display mode: 0=Each, 1=One
32: 1 ;Operator ID number
33: 2 ;Setup ID number
34: 3 ;Master ID number
35: 1;1,1,1,1,1,1,1,1 ;Per chn enable: chns 1-8; chn on/off
35: 2;1,1,1,1,1,1,1,1 ;Per chn enable: chns 9-16; chn on/off
35: 3;0,0,0,0,0,0,0,0 ;Per chn enable: chns 17-24; chn on/off
36: 1 ;Operator enable: Setpoints
37: 1 ;Operator enable: Default program
38: 1 ;Operator enable: Speed comp
39: 1 ;Operator enable: Timed outputs
40: 1 ;Operator enable: Offsets
41: 1 ;Operator enable: Motion Detection
42: 1 ;Operator enable: Analog values
43: 2;0,0,0,0,0,0,0,0 ;Motion ANDing: chns 9-16; chn levels (o=none)
43: 3;0,0,0,0,0,0,0,0 ;Motion ANDing: chns 17-24; chn levels (o=none)
44: 1;0,0,0,0,0,0,0,0 ;Output enable ANDing: chns 1-8; chn on/off
44: 2;0,0,0,0,0,0,0,0 ;Output enable ANDing: chns 9-16; chn on/off
44: 3;0,0,0,0,0,0,0,0 ;Output enable ANDing: chns 17-24; chn on/off
45: 2 ;Output group quantity
46: 1,10,0 ;Output group config: group, #chns, mode
46: 2,6,4 ;Output group config: group, #chns, mode
49: 1,1,0,90 ;Pulse: pgm, chn, on, off
49: 1,1,180,270 ;Pulse: pgm, chn, on, off
49: 1,2,0,180 ;Pulse: pgm, chn, on, off
49: 1,3,45,270 ;Pulse: pgm, chn, on, off
```

# Serial Communications Using Electro Cam Corp. Protocol (Standard 5144 Units)

## Background

PS-5144 controllers include programming that allows them to accept and respond to a set of serial commands issued by a system host such as a PLC or other computer. The commands can interrogate the PS-5144 for operating and control data, and they can also change programming values within the PS-5144.

Serial communications are initiated when the system host sends a command to the PS-5144. The PS-5144 processes the command and sends a reply to the host. Modbus ASCII protocol is available (see page 8-13).

## Syntax

All commands are sent and received as ASCII character strings in the following syntax. **Do not include spaces between fields.**

**Command from Host:** STX ADR CMD <DTA> CSM ETX

**Reply from PLS:** ACK or NAK <DTA> CSM ETX

<u>Field</u>	<u>No. of Characters</u>	<u>Description</u>
STX	1	Start of text. The PLS uses “!” for this character.
ADR	2 hex	Address of PLS controller on network (0-255)
CMD	2 hex	Command number. Commands are listed later on in this chapter.
DTA	n hex	The number and type of data elements is determined by the command, reply, or the error. <b>All data is sent and received in hex.</b>
CSM	2 hex	Checksum. The method by which the PS-5144 calculates the checksum is described later in this chapter. When the host sends a command, it must include a checksum calculated in the same way so that the PS-5144 can check the command for communication errors. The host should also use this calculation method to analyze the reply from the PS-5144 for possible communication errors.
ETX	1	End of text. The PLS uses a carriage return, or <CR>, for this character
ACK	1	Positive acknowledge. The PLS uses the letter “A” for ACK.
NAK	1	Negative acknowledge, or error condition. The PLS uses the letter “N” for NAK. A list of error replies are included later in this section.

The specified number of ASCII characters must be sent for each field. Include leading zeroes if the data in a field is less than the field length. The control will also include leading zeroes in its replies.

# Serial Commands

---

## Description

The PS-5144 controller recognizes a set of 95 commands. Some of these commands involve testing and diagnostic functions performed at the factory. Because these commands are of little use in field installations, they are not included in the following pages. For information on the complete command set, contact the factory.

The commands are grouped by general function. In the syntax shown for each command and reply, the characters used for STX, ETX, ACK, and NAK are substituted, as listed on the previous page.

**The commands are listed in hex.**

<b>CMD (hex)</b>	<b>Name</b>	<b>Function</b>
<b>04</b>	Hello	Are you there? Cmd: ! ADR <b>04</b> CSM <CR> Reply: A <CR>
<b>Supervisory Commands</b>	<b>06</b>	Com Stop Stop operation & idle; changes will be written directly to EEPROM with no other action taken. Cmd: ! ADR <b>06</b> CSM <CR> Reply: A <CR>
	<b>07</b>	Checksum Sets new checksums in EEPROM. Cmd: ! ADR <b>07</b> CSM <CR> Reply: A <CR>
	<b>08</b>	Start Resume operation. Cmd: ! ADR <b>08</b> CSM <CR> Reply: A <CR>
	<b>09</b>	Reset Create hard reset through watchdog. Cmd: ! ADR <b>09</b> CSM <CR> Reply: A <CR>
	<b>Status Commands</b>	<b>0A</b>
<b>38</b>		Shaft Pos Shaft position. Put: ! ADR <b>38</b> P XXXX CSM <CR> Reply: A <CR> Get: ! ADR <b>38</b> G CSM <CR> Reply: A XXXX CSM <CR> where "XXXX" is the shaft position in hex.
<b>0B</b>		Grp Pos Current position. Cmd: ! ADR <b>0B</b> XX CSM <CR> Reply: A YYYY CSM <CR> where "XX" is the group number minus one. "YYYY" is that group's position in hex.

## Serial Commands (cont'd)

---

	<b>CMD (hex)</b>	<b>Name</b>	<b>Function</b>
<b>Configuration Commands</b>	<b>56</b>	Kbd Qty	Number of keypads connected.  Put: ! ADR <b>56</b> P XX CSM <CR> Reply: A <CR>  Get: ! ADR <b>56</b> G CSM <CR> Reply: A XX CSM <CR>  where "XX" = number of keypads connected.
	<b>0D</b>	Setup ID	Setup ID code.  Put: ! ADR <b>0D</b> P XXXX CSM <CR> Reply: A <CR>  Get: ! ADR <b>0D</b> G CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" = Setup Enable Code in hex.
	<b>0E</b>	Operator ID	Operator ID code.  Put: ! ADR <b>0E</b> P XXXX CSM <CR> Reply: A <CR>  Get: ! ADR <b>0E</b> G CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" = Operator Enable Code in hex.
	<b>58</b>	Master ID	Master ID code.  Put: ! ADR <b>58</b> P XXXX CSM <CR> Reply: A <CR>  Get: ! ADR <b>58</b> G CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" = Master Enable Code in hex.
	<b>0F</b>	User Pgm	User programming enable/disable.  Put: ! ADR <b>0F</b> P XX <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>0F</b> G XX CSM <CR> Reply: A <00 or 01> CSM <CR>  where "XX" is the channel number minus 1, in hex. "00" = disable, and "01" = enable.
	<b>10</b>	Motion Enab	Motion detection on/off for a specified output channel.  Put: ! ADR <b>10</b> P XX <00, 01, or 02> CSM <CR> Reply: A <CR>  Get: ! ADR <b>10</b> G XX CSM <CR> Reply: A <00, 01, or 02> CSM <CR>  where "XX" is the channel number minus 1, in hex. "00" = L1 & L2 off; "01" = L1 on; "02" = L2 on.

## Serial Commands (cont'd)

---

	<b>CMD (hex)</b>	<b>Name</b>	<b>Function</b>
<b>Configuration Commands (cont'd)</b>	<b>12</b>	Inc Direction	Direction of increasing rotation.
		Put:	! ADR <b>12</b> P <00 or 01> CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>12</b> G CSM <CR>
		Reply:	A <00 or 01> CSM <CR>
			where "00" = CCW, and "01" = CW
	<b>13</b>	Scale Factor	Scale factor.
		Put:	! ADR <b>13</b> P XXXX CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>13</b> G CSM <CR>
		Reply:	A XXXX CSM <CR>
			where "XXXX" = scale factor in hex.
	<b>48</b>	Lo Limit	Motion detection low limit.
		Put:	! ADR <b>48</b> P <00 or 01> YYYY CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>48</b> G <00 or 01> CSM <CR>
		Reply:	A YYYY CSM <CR>
			where "YYYY" = low limit RPM in hex. "00" = Level 1, "01" = Level 2.
	<b>49</b>	Hi Limit	Motion detection high limit.
		Put:	! ADR <b>49</b> P <00 or 01> YYYY CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>49</b> G <00 or 01> CSM <CR>
		Reply:	A YYYY CSM <CR>
			where "YYYY" = high limit RPM in hex. "00" = Level 1, "01" = Level 2.
	<b>17</b>	Time Delay	Delay value for Timed Output channels.
		Put:	! ADR <b>17</b> P XX YYYY CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>17</b> G XX CSM <CR>
		Reply:	A YYYY CSM <CR>
			where "XX" is the channel minus 1, in hex, and "YYYY" is the delay in msec, in hex.
	<b>18</b>	Default Pgm	Default program.
		Put:	! ADR <b>18</b> P XXXX CSM <CR>
		Reply:	A <CR>
		Get:	! ADR <b>18</b> G CSM <CR>
		Reply:	A XXXX CSM <CR>
			where "XXXX" is the Default Program minus 1, in hex.

## Serial Commands (cont'd)

---

	<b>CMD</b>		
	<b>(hex)</b>	<b>Name</b>	<b>Function</b>
<b>Configuration Commands</b> (cont'd)	<b>1A</b>	Spd Cmp Mode	Standard or Leading/Trailing mode. Put: ! ADR <b>1A</b> P <00 or 01> CSM <CR> Reply: A <CR> Get: ! ADR <b>1A</b> G CSM <CR> Reply: A <00 or 01> CSM <CR> where "00" = Standard, "01" = Leading/Trailing
	<b>1B</b>	Spd Cmp Val	Speed comp value. Put: ! ADR <b>1B</b> P XX YYYY ZZZZ CSM <CR> Reply: A <CR> Get: ! ADR <b>1B</b> G XX CSM <CR> Reply: A YYYY ZZZZ CSM <CR> where "XX" is the channel minus 1, in hex. "YYYY" is the value in tenths of a msec for the leading edge, and "ZZZZ" is the value for the trailing edge. For standard speed comp, "YYYY" = "ZZZZ". "Y" and "Z" values are hex.
	<b>4B</b>	Analog Qty	Number of analog outputs used. Put: ! ADR <b>4B</b> P XX CSM <CR> Reply: A <CR> Get: ! ADR <b>4B</b> G CSM <CR> Reply: A XX CSM <CR> where "XX" is the number of analog outputs used. "XX" can be 00, 01, or 02.
	<b>1C</b>	Analog	Analog values. Put: ! ADR <b>1C</b> P XX YYYY ZZZZ CSM <CR> Reply: A <CR> Get: ! ADR <b>1C</b> G XX CSM <CR> Reply: A YYYY ZZZZ CSM <CR> where "XX" is the channel minus one, in hex. "YYYY" is the Offset from 0 to 4095, converted to hex. "ZZZZ" is the High RPM in hex.
	<b>1D</b>	Grp Qty	Output group quantity. Put: ! ADR <b>1D</b> P XX CSM <CR> Reply: A <CR> Get: ! ADR <b>1D</b> G CSM <CR> Reply: A XX CSM <CR> where "XX" is the number of output groups, from one to six.
	<b>4A</b>	Offset Mode	One offset for all groups, or individual offset for each group. Put: ! ADR <b>4A</b> P <00 or 01> CSM <CR> Reply: A <CR> Get: ADR <b>4A</b> G CSM <CR> Reply: A <00 or 01> CSM <CR> "00" = Each; "01" = One.

## Serial Commands (cont'd)

---

	<b>CMD</b>	<b>Name</b>	<b>Function</b>
<b>Configuration Commands</b> (cont'd)	<b>1E</b>	Grp Offset	Output group offset value.  Put: ! ADR <b>1E</b> P XX YYYY CSM <CR> Reply: A <CR>  Get: ADR <b>1E</b> G XX CSM <CR> Reply: A YYYY CSM <CR>  where "XX" is the group number minus 1. "YYYY" is the offset value for that group, in hex.
	<b>3C</b>	Shaft Offset	Shaft position offset.  Put: ! ADR <b>3C</b> P XXXX CSM <CR> Reply: A <CR>  Get: ! ADR <b>3C</b> G CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" is the shaft offset in hex.
	<b>1F</b>	Grp Chn Qty	Number of channels in a specified output group.  Put: ! ADR <b>1F</b> P XX YY CSM <CR> Reply: A <CR>  Get: ! ADR <b>1F</b> G XX CSM <CR> Reply: A YY CSM <CR>  where "XX" is the group number minus one. "YY" is the number of output channels in that group, in hex.
	<b>21</b>	Mode	Mode for the specified output group.  Put: ! ADR <b>21</b> P XX YY CSM <CR> Reply: A <CR>  Get: ! ADR <b>21</b> G XX CSM <CR> Reply: A YY CSM <CR>  where "XX" is the group number minus one. "YY" is the operating mode, from zero to five.
	<b>47</b>	Output Enab	Output Enable ANDing on or off for specified channel.  Put: ! ADR <b>47</b> P XX <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>47</b> G XX CSM <CR> Reply: A <00 or 01> CSM <CR>  where "XX" is the channel number minus one, in hex. "00" = ANDing "off"; "01" = ANDing "on".
	<b>4D</b>	Pgm Sel Mode	Program select terminals use Binary, Gray Code, or BCD format.  Put: ! ADR <b>47</b> P <00, 01, or 02> CSM <CR> Reply: A <CR>  Get: ! ADR <b>47</b> G CSM <CR> Reply: A <00, 01, or 02> CSM <CR>  "00" = Binary; "01" = Gray Code; "02" = BCD.

## Serial Commands (cont'd)

---

	<b>CMD</b>		
	<b>(hex)</b>	<b>Name</b>	<b>Function</b>
<b>Setpoint Commands</b>	<b>22</b>	Spt Count	Return number of pulses.  Cmd: ! ADR <b>22</b> CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" is the total number of pulses in hex. Includes all pulses in all channels and programs in the controller.
	<b>23</b>	Wipe Spt	Deletes all pulses from EEPROM.  Cmd: ! ADR <b>23</b> CSM <CR> Reply: A <CR>
	<b>24</b>	Get Spt	Return program, channel, and on/off points for the specified pulse.  Cmd: ! ADR <b>24</b> XXXX CSM <CR> Reply: A XX YY ZZZZ TTTT CSM <CR>  where "XXXX" is the number of the pulse in hex. Pulses are numbered starting at Channel 1, Program 1, Position 0. As the transducer rotates through a complete cycle, each pulse encountered is numbered sequentially. After one cycle, the pulses in Channel 2 are numbered, and so on.  In the reply, "XX" is the program number of the specified pulse, minus one. "YY" is the channel number, minus one. "ZZZZ" and "TTTT" are the "on" and "off" points of the pulse, respectively. All values are in hex.
	<b>25</b>	Add Spt	Adds a setpoint.  Cmd: ! ADR <b>25</b> XX YY ZZZZ TTTT CSM <CR> Reply: A <CR>  where "XX" is the program number minus one, and "YY" is the channel number minus one. "ZZZZ" and "TTTT" are the "on" and "off" points of the pulse, respectively. All values are in hex.
	<b>26</b>	Del Spt	Deletes a setpoint.  Cmd: ! ADR <b>26</b> XX YY ZZZZ TTTT CSM <CR> Reply: A <CR>  where "XX" is the program number minus one, and "YY" is the channel number minus one. "ZZZZ" and "TTTT" are the "on" and "off" points of the pulse, respectively. All values are in hex.
	<b>27</b>	Mod Spt	Modifies one edge of a setpoint.  Cmd: ! ADR <b>27</b> XX YY ZZZZ TTTT MM NNNN CSM <CR> Reply: A <CR>  where "XX" is the program number minus one and "YY" is the channel number minus one.  "ZZZZ" and "TTTT" are the <b>current</b> "on" and "off" points of the pulse, respectively.  "MM" is the edge to be modified: "00" is the "off" edge, "01" is the "on" edge.  "NNNN" is the new value for the specified edge. All values are in hex.

## Serial Commands (cont'd)

---

	<b>CMD</b>		
	<b>(hex)</b>	<b>Name</b>	<b>Function</b>
<b>Setpoint Commands</b> (cont'd)	<b>28</b>	Inc Spt	Advances one edge of a pulse, both edges, or all pulses in a channel, by one scale factor increment.  Cmd: ! ADR <b>28</b> XX YY ZZZZ TTTT MM CSM <CR> Reply: A <CR>  where "XX" is the program number minus one, and "YY" is the channel number minus one. "ZZZZ" and "TTTT" are the <b>current</b> "on" and "off" points of the pulse, respectively. "MM" specifies the scope of the change: "00" is the "off" edge; "01" is the "on" edge; "02" is both edges of the pulse; and "03" is all edges of all pulses in the channel. All values are in hex.
	<b>29</b>	Dec Spt	Retards one edge of a pulse, both edges, or all pulses in a channel, by one scale factor increment.  Cmd: ! ADR <b>29</b> XX YY ZZZZ TTTT MM CSM <CR> Reply: A <CR>  where "XX" is the program number minus one, and "YY" is the channel number minus one. "ZZZZ" and "TTTT" are the <b>current</b> "on" and "off" points of the pulse, respectively. "MM" specifies the scope of the change: "00" is the "off" edge; "01" is the "on" edge; "02" is both edges of the pulse; and "03" is all edges of all pulses in the channel. All values are in hex.
<b>Display Commands</b>	<b>30</b>	Def Disp	Default display on start-up.  Put: ! ADR <b>30</b> P XX CSM <CR> Reply: A <CR>  Get: ! ADR <b>30</b> G CSM <CR> Reply: A XX CSM <CR>  where "XX" is the display mode: "00" is Speed, "01" is Position, and "02" is Auto.
	<b>31</b>	Tog RPM	Toggle RPM speed.  Put: ! ADR <b>31</b> P XXXX CSM <CR> Reply: A <CR>  Get: ! ADR <b>31</b> G CSM <CR> Reply: A XXXX CSM <CR>  where "XXXX" is the toggle RPM speed in hex.
	<b>57</b>	Rate Setup	Multiplier and units for RPM display.  Put: ! ADR <b>57</b> P XX YY CSM <CR> Reply: A <CR>  Get: ! ADR <b>57</b> G CSM <CR> Reply: A XX YY CSM <CR>  "XX" is the multiplier: "01" = 1X; "02" = 2X; "03" = 3X; "04" = .5X. "YY" = units: "00" = RPM; "01" = BPM; "02" = CPM

## Serial Commands (cont'd)

---

	<b>CMD</b>		
	<b>(hex)</b>	<b>Name</b>	<b>Function</b>
<b>Special Commands</b>	<b>2A</b>	Key Press	Adds a value to the keyboard buffer; just like pressing a key.  Cmd: ! ADR <b>2A</b> XX CSM <CR> Reply: A <CR>  where "XX" is the key number in hex. See "Keypad Diagnostics" in Section 7 for a method to determine the key number for each key on the keypad.
	<b>2B</b>	En Mot Spt	Enable "Motion ANDing" programming at operator level.  Put: ! ADR <b>2B</b> P <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>2B</b> G CSM <CR> Reply: A <00 or 01> CSM <CR>  where "00" = disabled, "01" = enabled.
	<b>2C</b>	En Offset	Enable "Offset" programming at operator level.  Put: ! ADR <b>2C</b> P <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>2C</b> G CSM <CR> Reply: A <00 or 01> CSM <CR>  where "00" = disabled, "01" = enabled.
	<b>2D</b>	En Act Pgm	"Active Program" enable at operator level.  Put: ! ADR <b>2D</b> P <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>2D</b> G CSM <CR> Reply: A <00 or 01> CSM <CR>  where "00" = disabled, "01" = enabled.
	<b>2E</b>	En Spd Cmp	Enable "Speed Comp" programming at operator level.  Put: ADR <b>2E</b> P XX <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>2E</b> G XX CSM <CR> Reply: A <00 or 01> CSM <CR>  where "XX" is the channel number minus 1, in hex. "00" = disabled, "01" = enabled.
	<b>2F</b>	En Timed Out	Enable "Timed Output" programming at operator level.  Put: ! ADR <b>2F</b> P XX <00 or 01> CSM <CR> Reply: A <CR>  Get: ! ADR <b>2F</b> G XX CSM <CR> Reply: A <00 or 01> CSM <CR>  where "XX" is the channel number minus 1, in hex. "00" = disabled, "01" = enabled.

## Error Codes

---

### Error Replies

If a command sent to the PS-5144 cannot be processed for any reason, the controller sends a reply in the following format:

N <error code> CSM <CR>

The error codes are listed below.

<b>Code</b>	<b>Name</b>	<b>Meaning</b>
00	OK	Processed ok.
01	BAD BUFFER	Buffer not correct.
02	NOT OUR ADDRESS	To someone else.
03	BAD COMMAND	Illegal command.
04	BAD DATA	Illegal data.
05	NOT IN MOTION	Can't do while running.
06	TOO MANY TIMERS	Too many timers for time base.
07	NOT AN OPTION	Option not on unit.
08	NOT STOPPED	Can't do this unless STOPPED.
09	BAD FORMAT	Bad input or output format string.
0A	TIMEOUT	Timeout error.
0B	BAD KEY	Illegal key value.
0C	FLASH ERROR	Flash programming error.
0D	BAD PROGRAM#	Illegal program number.
0E	BAD CHANNEL#	Illegal channel number.
0F	KEYBOARD CONFLICT	Conflict with keyboard activity.

## Checksum

---

### Calculating Checksum

The PS-5144 calculates checksums in four steps:

1. Add the ASCII values of the command string, not including STX (!) or ETX (<CR>).
2. Make the decimal value from Step 1 negative.
3. Convert the value from Step 2 to hex.
4. Use the two least significant digits from Step 3.

The following examples will clarify how Checksums are calculated:

#### Example 1—Command 0A: Request RPM from Controller #1

Command: !010A<CSM><CR>

Checksum Calculation:

```
  0  1  0  A
  |  |  |  |
 48+49+48+72 = 217(decimal)
```

-217 decimal = FF27 hex; therefore: Checksum = 27

String sent to controller = !010A27<CR>

#### Example 2—Command 25: Add Pulse to Control #2

Pulse Values: Program 15, Output Channel 9, "On" at 25, "Off" at 290

Command: !02250E0800190122<CSM><CR>

Checksum Calculation:

```
  0  2  2  5  0  E  0  8  0  0  1  9  0  1  2  2
  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
48+50+50+53+48+69+48+56+48+48+49+57+48+49+50+50 = 821(decimal)
```

-821 decimal = FCCB hex; therefore: Checksum = CB

String sent to controller = !02250E0800190122CB<CR>

# Serial Communications Using Modbus ASCII Protocol (PS-5144-MB Units)

---

## Data Organization

---

This section describes the internal data structure of PLS controllers, and how this data may be accessed via serial communications. The data has been organized as a series of "Coils" and "Registers" compatible with PLC programming techniques. You access and/or change the data within a PLS controller by forcing coils ON or OFF, and by reading and writing register data.

A PLS Controller can be completely programmed via the serial interface. All controller data, such as pulses, speed compensation, timed output values, etc., are available as registers. Configuration data, such as the direction of rotation, number of keyboards, number of analog outputs, etc., is also available as register data. The controller is programmed by writing to these registers. Data is monitored within the controller by reading from these registers.

Note: The ability of the EEPROM to retain data is reduced after 100,000 write cycles. Do not set up routines that constantly write data to the EEPROM's.

## Mapping

---

In addition to accessing controller data via dedicated registers, specific indexed data items can be accessed through the 240 data display registers. This is done by "mapping" a specific indexed data element to a data display register; a data display register is assigned to represent a pulse, speed comp value, etc. Once an indexed data element is mapped it can be accessed either through the data display register or through the dedicated register.

Mapping is useful when displaying more than one instance of an indexed data element at once. For instance, speed compensation is accessed via three registers; 1) a channel index, 2) a leading edge value, and 3) a trailing edge value. This means that the values of speed compensation for all channels can be accessed, but only one at a time. To display more than one value of speed compensation at once, simply map the values to a series of data display registers.

***You must define how many mappings are available through the Map Limit register.***

## Modbus

---

Modbus ASCII protocol is used for serial communications.

Set host controller communication parameters to 7 data bits, 2 stop bits, no parity.

Limit the number of consecutive registers or coils read to 32.

## Quick Reference

<i>Discrete Elements</i>	
<b>Inputs</b>	
10001 - 10016	DC Inputs
<b>Outputs</b>	
00001 - 00100	Channel Outputs
<b>ORing and ANDing</b>	
00101 - 00200	Channel ORing
00201 - 00300	Channel ANDing
<b>Special Purpose</b>	
00301 - 00400	Special Purpose
00301	Global Unforce
00302	Pulse Register Enable
00303	Create New Pulse
00304	Move Both Edges of Pulse
00305	Move All Pulses in Channel
00314	NAK Bad Address Reads
00315	Execute Special Function
00316	Auto Increment

<i>Registers</i>	
<b>Special Purpose &amp; Data Display</b>	
40001	Message and Special Function (16 registers)
40017	Data Display (240 registers)
<b>RPM</b>	
40257	RPM
<b>Position</b>	
40258	Position Mapping
40259	Position Index
40260	Position
<b>Pulse Programming</b>	
40261	Pulse Mapping
40262	Total Pulse Count
40263	Channel Pulse Count

<b>Pulse Programming (Cont.)</b>	
40264	Program Index
40265	Channel Index
40266	Pulse Index
40267	Pulse On
40268	Pulse Off
40269	New On
40270	New Off
<b>Default Program</b>	
40271	Default Program
<b>Speed Compensation</b>	
40272	Speed Comp Mapping
40273	Channel Index
40274	Leading Edge Comp
40275	Trailing Edge Comp
<b>Timed Outputs</b>	
40276	Timed Output Mapping
40277	Channel Index
40278	Time Delay
<b>Offset</b>	
40279	Offset Mapping
40280	Group Index
40281	Group Offset
<b>Motion Detection</b>	
40282	Motion Detection Mapping
40283	Channel Index
40284	Low Motion Detection RPM
40285	High Motion Detection RPM
<b>Analog Output</b>	
40286	Analog Output Mapping
40287	Channel Index
40288	Analog Offset
40289	Analog High RPM
<b>Gray Code Speed Compensation</b>	
40290	Gray Code Speed Comp

## Quick Reference

---

### Mapping Registers

40296	Map Limit
40297	Map Quantity
40298	Map Store
40299	Map Recall

### Model Information

40300	Model
40301	Revision
40302	Output Quantity
40303	Option Index
40304	Option

### Hardware Configuration

40306	Increasing Direction
40307	Scale Factor
40308	Shaft Position
40309	Shaft Offset
40310	Analog Quantity
40311	Resolver Type
40312	Program Select Mode
40313	Gray Level
40314	Time Base
40315	Termination Resistor One
40316	Termination Resistor Two

### Display Configuration

40317	Default Display
40318	Rate Multiplier
40319	Rate Divisor
40320	Rate Decimal Point Position
40321	Rate Units
40322	Toggle RPM
40323	RPM Update Rate
40324	Speed Comp Display Mode
40325	Group Position Display Mode

### Password ID Numbers

40326	Operator ID
40327	Setup ID
40328	Master ID

### Per Channel Enable

40329	Per Channel Enable Index
40330	Per Channel Enable

### Operator Function Enable

40331	Operator Function Enable Bitmask
-------	----------------------------------

### Motion ANDing

40332	Channel Index
40333	Motion Enable Level

### Output Enable ANDing

40334	Output Enable Index
40335	Output Enable

### Group Programming

40336	Group Quantity
40337	Group Index
40338	Channel Quantity
40339	Group Mode

### Run Time Control

40340	Stop Control
40341	EEPROM Checksum
40342	EEPROM Changed

---

The following registers are not supported by early versions of Modbus Controllers.

---

### Active Program

40343	Active Program
-------	----------------

### I/O Control

40350 - 40359	Input Status
40360 - 40369	Output Status
40370 - 40379	ORing Bits
40380 - 40389	ANDing Bits

### Communications

40390	Type (RS485/RS232)
40391	Baud Rate
40392	Address

## Discrete I/O

---

### Inputs

---

10001 - 10016

#### DC Inputs

These points represent the status of the DC inputs.

### Outputs

---

00001 - 00100

#### Channel Outputs

These coils represent the status of the channel outputs. Forcing these coils directly will set/clear the appropriate ORing and ANDing coils as required. The Channel Output Coil status before OR/ANDing is determined by setpoints, group modes, speed compensation, motion ANDing, enable input ANDing, timed outputs, and resolver fault status.

### ORing and ANDing

---

00101 - 00200

#### Channel ORing

Setting these coils to '1' will force the corresponding Channel Output Coil ON.

00201 - 00300

#### Channel ANDing

Setting these coils to '1' will force the corresponding Channel Output Coil OFF.

### Special Purpose

---

00301 - 00400 **Special Purpose**

301

#### Global Unforce

Clears all OR and AND coils when set from '0' to '1' (edge active).

302

#### Pulse Register Enable

When '1', this coil enables the creation of new pulses through writes to the New Off Register. When this coil is '0', writes to New Off Register do not create a new pulse.

303

#### Create New Pulse

Creates a new pulse defined by the New On and New Off registers when set from '0' to '1' (edge active). This coil is ignored if coil 302 is '1'.

304

#### Move Both Edges of Pulse

When '1', this coil will cause both edges of a pulse to move when either the leading or trailing edge is changed by '1' (incremented or decremented).

305

#### Move All Pulses in Channel

When '1', this coil will cause all edges of all pulses in a channel to move when either the leading or trailing edge is changed by '1' (incremented or decremented).

314

#### NAK Bad Address Reads

When '1', this coil will cause the controller to NAK attempted reads to non-existent registers. When this coil is '0', reads to non-existent registers return a value of zero.

315

#### Execute Special Function

Executes the special function defined by the contents of the Special Purpose Registers (40001-40017) when set from '0' to '1'.

316

#### Auto Increment

When '1', this coil enables the auto increment feature on index registers. This feature allows sequential reading of indexed values without changing the index register.

## Registers

---

### Special Purpose & Data Display

---

- 40001 Special Function (16 registers)**  
The first 16 registers (001 - 016) are used for entering data used by the special functions.
- 40017 Data Display (240 registers)**  
These registers (017 - 256) are used by the Mapping functions to display individual instances of indexed data.

### RPM

---

- 40257 RPM**  
Read only  
Returns the current RPM.

### Position

---

- 40258 Position Mapping**  
Read/write  
Values: 17 - 256  
Specifies the general purpose register used to display the position for the output group specified by the Group Index Register.
- 40259 Position Index**  
Read/write  
Values: 1 - 6  
Specifies the output group whose position is displayed in the Position Register.
- 40260 Position**  
Read only  
Values: 0 - ( Scale Factor - 1 )  
returns the current position for the output group specified by the Group Index Register.

### Pulse Programming

---

- 40261 Pulse Mapping**  
Read/write  
Values: 17 - 255  
General Purpose register used for mapping the On and Off values for the pulse specified by the index registers. Two registers will be used; the first will contain the On value, the second will contain the Off value.
- 40262 Total Pulse Count**  
Read/write  
Values: 0 - n  
Returns the total number of pulses for all channels. Writing a value of '0' to this register will erase all pulses. You can only write to this register when the Stop register is '1'.
- 40263 Channel Pulse Count**  
Read only  
Values: 0 - n  
Returns the number of pulses in the channel defined by the index registers below.
- 40264 Program Index**  
Read/write  
Values: 0 - Max Program Number  
Contains the current program number for pulse access. Writing to this register resets the Channel Index Register and the Pulse Index Register to '1'. When this register is '0', the current active program is used for setpoint access and for mapping (setpoints mapped with a program index of '0' will automatically change when the active program changes).

## Registers (Cont'd)

---

### Pulse Programming (Con'td)

---

40265	<b>Channel Index</b> Read/write Values: 1 - Max Channel Number Contains the current channel number for pulse access. Writing to this register resets the Pulse Index Register to '1'. This register is reset to '1' when the Program Index Register is changed.
40266	<b>Pulse Index</b> Read/write Values: 1 - n Contains the current pulse number for pulse access. This register is reset to '1' when the Program Index Register or Channel Index Registers are changed.
40267	<b>Pulse On</b> Read/write Values: 0 - ( Scale Factor - 1 ) Pulse On Value.
40268	<b>Pulse Off</b> Read/write Values: 0 - ( Scale Factor - 1 ) Pulse Off Value.
40269	<b>New On</b> Read/write Values: 0 - ( Scale Factor - 1 ) New Pulse On Value. Writing to this register loads the On setpoint of a new pulse for the program and channel specified by the index registers above.
40270	<b>New Off</b> Read/write Values: 0 - ( Scale Factor - 1 ) New Pulse Off Value. Writing to this register loads the Off setpoint of a new pulse for the program and channel specified by the index registers above. The pulse is stored when the Off value is written if the Pulse Register Enable Coil is set to '1'; otherwise the pulse is stored when the Create New Pulse Coil is changed from '0' to '1' (edge active).

### Default Program

---

40271	<b>Default Program</b> Read/Write. Values: 1 - Max program number Defines the program that will be active if no hardware program select inputs are active.
-------	---

### Speed Compensation

---

40272	<b>Speed Comp Mapping</b> Read/Write Values: 17 - 255 General purpose register used for mapping speed compensation values. Two registers will be used; the first will contain the leading edge value, the second will contain the trailing edge value.
40273	<b>Channel Index</b> Read/Write Values: 1 - Max Channel Number Channel index for speed comp values.

## Registers (Cont'd)

---

### Speed Compensatin (Cont'd)

---

- 40274**      **Leading Edge Comp**  
Read/Write  
Values: 0 - n (.1mS)  
Specifies the leading edge speed comp value.
- 40275**      **Trailing Edge Comp**  
Read/Write  
Values: 0 - n (.1mS)  
Specifies the trailing edge speed comp value.

### Timed Outputs

---

- 40276**      **Timed Output Mapping**  
Read/write  
Values: 17 - 255  
General purpose register used for mapping timed output values.
- 40277**      **Channel Index**  
Read/Write  
Values: 1 - Max Channel Number  
Channel index for time delay values.
- 40278**      **Time Delay**  
Read/write  
Values: 0 - n (1mS)  
Specifies the maximum time in milliseconds that a channel may stay on after it has been turned on.

### Offset

---

- 40279**      **Offset Mapping**  
Read/write  
Values: 17 - 256  
General purpose register used for mapping Group Offset values.
- 40280**      **Group Index**  
Read/write  
Values: 1 - 6  
Group index for offset values.
- 40281**      **Group Offset**  
Read/write  
Values: 0 - ( Scale Factor - 1 )  
Offset value for the specified group.  
Note that this value is a PRESET value for groups in modes 1 or 2.

### Motion Detection

---

- 40282**      **Motion Detection Mapping**  
Read/write  
Values: 17 - 255  
General purpose register used for mapping low and high motion detection values. Two registers will be used; the first will contain the low motion detection rpm value, the second will contain the high motion detection rpm value.
- 40283**      **Channel Index**  
Read/write  
Values: 1, 2  
Motion detection level index for high and low motion detection values.

## Registers (Cont'd)

---

### Motion Detection (Cont.)

---

- 40284 Low Motion Detection RPM**  
Read/write  
Values: 0 - n  
Motion detection low limit for the level specified by the index register.
- 40285 High Motion Detection RPM**  
Read/write  
Values: 0 - n  
Motion detection high limit for the level specified by the index register.

### Analog Output

---

- 40286 Analog Output Mapping**  
Read/write  
Values: 17 - 255  
General purpose register used for mapping analog offset and high RPM values. Two registers will be used; the first will contain the analog offset value, the second will contain the high RPM value.
- 40287 Channel Index**  
Read/write  
Values: 1, 2  
Analog channel index for analog offset and high RPM values.
- 40288 Analog Offset**  
Read/write  
Values: 0 - 4095  
Analog output at 0 RPM.
- 40289 Analog High RPM**  
Read/write  
Values: 0 - 3000  
RPM at which analog output is 4095.

### Gray Code Speed Compensation

---

- 40290 Gray Code Speed Comp**  
Read/write  
Values: 0 - n (.1mS)  
In controllers equipped with the "-G" option, the Gray code bit pattern is speed compensated by this amount.

### Mapping Registers

---

- 40296 Map Limit**  
Read/write  
Values: 0 - 256  
Sets the maximum number of data mappings.
- 40297 Map Quantity**  
Read/write  
Values: 0 - 256  
Returns the number of data mappings active in the controller.  
NOTE: Writing a '0' to this register will delete all data mappings!
- 40298 Map Store**  
This register is only for use by utility programs.
- 40299 Map Recall**  
This register is only for use by utility programs.

## Registers (Cont'd)

---

### Model Information

---

40300	<b>Model</b> Read only Returns the PLuS model number (5144, 6144, etc.).
40301	<b>Revision</b> Read only Returns the major software revision.
40302	<b>Output Quantity</b> Read only Returns the number of output channels (8, 9, 16, 17, 25, etc).
40303	<b>Option Index</b> Read/write Values: 1 - n Used as index for reading installed controller options through the Option Register.
40304	<b>Option</b> Read only Values: 0 - n Returns installed controller options as specified through the Option Index Register. A value of '0' at index '1' means no options are installed.

### Hardware Configuration

---

40306	<b>Increasing Direction</b> Read/write Values: 0 = CCW, 1 = CW Specifies the direction of rotation of the resolver (viewed from the shaft end) that will result in an increasing numerical display of position.
40307	<b>Scale Factor</b> Read/write Values: 2 - 1024 (4096 with "-H" Option) Scale factor used for pulse, position, and offset programming.
40308	<b>Shaft Position</b> Read only Values: 0 - ( Scale Factor - 1 ) Returns the current resolver shaft position, including the shaft offset.
40309	<b>Shaft Offset</b> Read/write Values: 0 - ( Scale Factor - 1 ) Offset that is added to raw resolver position to make Shaft Position.
40310	<b>Analog Quantity</b> Read/write Values: 0, 1, 2 Specifies the number of analog modules active.
40311	<b>Resolver Type</b> Read/write Values: 0 = Electro Cam, 1 = Other Specifies type of resolver attached to controller.
40312	<b>Program Select Mode</b> Read/write Values: 0 = Binary, 2 = BCD, 1 = Gray code Specifies how the program select inputs determine the active program.

## Registers (Cont'd)

---

### Hardware Configuration (Cont'd)

---

- 40313 Gray Level**  
Read/write  
Values: 0 = Positive True, 1 = Negative True  
On controllers equipped with the "-G" Option, this register specifies the logic level of the Gray code bit pattern.
- 40314 Time Base**  
Read only  
Values: 0 = 1mS, 1 = .5mS, 2 = .2mS  
Returns the timer interrupt rate.
- 40315 Termination Resistor One**  
Read/write  
Values: 0 = Off, 1 = On  
Termination resistor On/Off RS485 port; keyboard port for 6000's, RS485 Communication port for 5144's.
- 40316 Termination Resistor Two**  
Read/write  
Values: 0 = Off, 1 = On  
Termination resistor On/Off for RS232/RS485 port; communication port for 6000's with 5144A Input Board.

### Display Configuration

---

- 40317 Default Display**  
Read/write  
Values: 0 = RPM, 1 = Position, 2 = Auto Select  
Specifies Pos/Rpm display mode; only applicable on 5XXX controllers.
- 40318 Rate Multiplier**  
Read/write  
Values: 1 - 1091  
RPM rate multiplier; 6000 controllers only.
- 40319 Rate Divisor**  
Read/write  
Values: 1 - 63  
RPM rate divisor, 6000 controllers only.
- 40320 Rate Decimal Point Position**  
Read/write  
Values: 0 - 3  
RPM decimal point position; 6000 controllers only.
- 40321 Rate Units**  
Read/write  
Values: 0 = RPM, 1 = BPM, 2 = CPM, 3 = IPM  
RPM display units; 6000 controllers only.
- 40322 Toggle RPM**  
Read/write  
Values: 0 - n  
Specifies RPM which will cause position display to blank (6000 series) or to change from Position to RPM (5000 series).
- 40323 RPM Update Rate**  
Read/write  
Values: 0 = 1/Sec, 1 = 2/Sec, 2 = 10/Sec  
Rate at which the RPM display is updated.

## Registers (Cont'd)

---

### Display Configuration

---

- 40324 Speed Comp Display Mode**  
Read/write  
Values: 0 = One, 1 = L/T  
Specifies whether speed comp values are displayed as one value for both leading and trailing edges, or as a value for each.
- 40325 Group Position Display Mode**  
Read/write  
Values: 0 = Each, 1 = One  
Specifies whether the positions for output groups are individually displayed, or if they are displayed as one value for all groups. Output group positions can only be displayed as one if none are in mode 1 or mode 2 (rezero modes).

### Password ID Numbers

---

- 40326 Operator ID**  
Read/write  
Values: 0 - n  
Specifies the Operator ID number used to enable the Operator access level for programming.
- 40327 Setup ID**  
Read/write  
Values: 0 - n  
Specifies the Setup ID number used to enable the Setup access level for programming.
- 40328 Master ID**  
Read/write  
Values: 0 - n  
Specifies the Master ID number used to enable the Master access level for programming.

### Per Channel Enable

---

- 40329 Per Channel Enable Index**  
Read/write  
Values: 1 - Max Channel Number  
Channel index for the Per Channel Enable register.
- 40330 Per Channel Enable**  
Read/write  
Values: 0=No Operator access, 1=Operator access enabled  
Specifies whether channel data can be modified under the Operator access level (0=no, 1=yes).  
Channel data such as speed comp and timed output values can be individually enabled per channel for operator access through this register.

### Operator Function Enable

---

- 40331 Operator Function Enable Bitmask**  
Read/write  
Values: 0 - 0FFFFH  
Bit mask which specifies which programming functions the operator may perform.  
Bit 0: Pulse on/off values.  
Bit 1: Default program.  
Bit 2: Speed compensation.  
Bit 3: Timed outputs.  
Bit 4: Offsets.  
Bit 5: Motion detection.  
Bit 6: Analog offset & high rpm.

## Registers (Cont'd)

---

### Motion ANDing

---

- 40332 Channel Index**  
Read/write  
Values: 1 - Max Channel Number  
Channel index for the Motion Enable Level Register.
- 40333 Motion Enable Level**  
Read/write  
Values: 0 = Off, n = Motion Detection Level  
Specifies the motion detection level used for a channel.

### Output Enable ANDing

---

- 40334 Output Enable Index**  
Read/write  
Values: 1 - Max Channel Number  
Channel index for the Output Enable register.
- 40335 Output Enable**  
Read/write  
Values: 0=Channel not ANDed, 1=Channel ANDed  
Specifies whether a channel is ANDed with the Enable Input.

### Group Programming

---

- 40336 Group Quantity**  
Read/write  
Values: 1 - 6  
Specifies the number of output groups.
- 40337 Group Index**  
Read/write  
Values: 1 - 6  
Group index for Channel Quantity and Group Mode Registers.
- 40338 Channel Quantity**  
Read/write  
Values: 0 - n  
Defines the number of channels in the output group specified by the Group Index Register.
- 40339 Group Mode**  
Read/write  
Values: 0 - 5  
Defines the operating mode for the output group specified by the Group Index Register. Note that groups in mode '0' do not need (or have) an enable input.

### Run Time Control

---

- 40340 Stop Control**  
Read/write  
Values: 0 = Running, 1 = Stopped  
When PLuS is STOPPED, changes written to registers do not update the checksum in EEPROM memory. Changes are faster when unit is stopped, but you must read from the Checksum Register when changes are complete to establish a valid checksum. Writing a '1' value to this register will place the PLuS in STOPPED mode. Writing a '0' to this register will restart the PLuS via a watchdog timer reset.
- 40341 EEPROM Checksum**  
Read only  
Returns the current checksum of EEPROM memory. If computed checksum of EEPROM memory does not match the current value (i.e. if changes were made while unit STOPPED), a new value will be written to EEPROM memory.

## Registers (Cont'd)

---

### Run Time Control (Cont'd)

---

**40342 EEPROM Changed**  
Read only  
Values: 0 = no change, 1 = changed.  
A value of '1' in this register means that the EEPROM has been changed (through the keyboard) since the last time this register was read. Reading this register sets it to '0'.

### Active Program

---

**40343 Active Program**  
Read/Write.  
Values: 1 - Max program number  
Returns to program currently active; determined either by hardware inputs or by the value of the default program. If hardware inputs are active, writes to this register will change the default program, but the active program will not change.

### I/O Control

---

**40350 - 40359 Input Status**  
Read Only.  
Values: 0 - 65535  
Each register represents the status of 16 inputs.

**40360 - 40369 Output Status**  
Read/Write.  
Values: 0 - 65535  
Each register represents the status of 16 outputs. The least significant bit of the register corresponds to the lowest numbered output. Writing to one of these registers will force 16 outputs. The ORing and ANDing registers (and coils) will reflect the forced conditions.

**40370 - 40379 ORing Bits**  
Read/Write.  
Values: 0 - 65535  
Each register represents the status of 16 ORing bits. The least significant bit of the register corresponds to the lowest numbered output. When a '1' is present in an outputs' bit position, the output will be forced ON. The OUTPUT STATUS register will reflect the forced condition.

**40380 - 40389 ANDing Bits**  
Read/Write.  
Values: 0 - 65535  
Each register represents the status of 16 ANDing bits. The least significant bit of the register corresponds to the lowest numbered output. When a '1' is present in an outputs' bit position, the output will be forced OFF. The OUTPUT STATUS register will reflect the forced condition.

### Host Communications Setup

---

**40390 Communication Type (RS485/RS232)**  
Read/Write.  
Values: 0/1 (0=RS485, 1=RS232)  
Determines the communication type used by the controller. This register may only be written to when the controller is stopped (via the STOP CONTROL register).

**40391 Communication Baud Rate**  
Read/Write.  
Values: 2/3/4/5 (2=4800, 3=9600, 4=19200, 5=38400)  
Determines the baud rate used by the controller. This register may only be written to when the controller is stopped (via the STOP CONTROL register).

## Registers (Cont'd)

---

### Host Communications Setup (Cont'd)

---

40392

**Communication Address**

Read/Write.

Values: 1-255

Determines the address used by the controller. This register may only be written to when the controller is stopped (via the STOP CONTROL register).

NOTE: If the three address switches on the input board are all UP (address 7), the controller will be automatically configured to be RS232, 9600 baud, address 1. Use this feature to enable communications with a controller if no keyboard is available or if you are unsure of the communication parameters currently in use.

## Special Functions

---

### Overview

---

Special functions are used to implement features not directly defined by the standard registers. Special functions are executed by loading the special purpose registers (40001-40016) with data, and then bringing the Execute Special Function Coil (00315) from '0' to '1'.

The data loaded into the special purpose registers is dictated by the special function being performed; each different special function will define the number and use of the special purpose registers.

Register 40001 will define the special function to be performed; registers 40002-40016 will hold the data needed for the special function.

### Pulse Copy

---

This function will add a series of pulses to a specific program and channel.

Register Use:

- 40001: 1 (Pulse Copy)
- 40002: Program number.
- 40003: Channel number.
- 40004: Beginning on value of pulse envelope.
- 40005: Ending off value of pulse envelope
- 40006: Number of pulses within envelope.
- 40007: Duration (width) of each pulse within envelope.

Registers 40004 and 40005 define the on and off values of the envelope pulse that will be divided into a series of smaller pulses.

Register 40006 contains the number of pulses that the envelope pulse will be divided into.

Register 40007 contains the duration of each of the smaller pulses.

This function will not be completed if the envelope pulse would overlap any other pulse in the specified program and channel, or if the count and duration values would result in overlapping pulses within the envelope pulse.

Once the registers have been loaded, bring the special purpose coil number 315 from '0' to '1'. The command will be acknowledged when pulse programming is complete. Special purpose coil number 315 must be made '0' before this function can be used again.

### EEPROM Clearing

---

This function will clear various areas of EEPROM memory.

Register Use:

- 40001: -3 (EEPROM Clearing)
- 40002: EEPROM Clearing Function Number:
  - 7000: Clear all EEPROM memory.
  - 7001: Clear configuration memory.
  - 7002: Clear setpoint memory.