ActivMEDIA INCORPORATED

Pioneer

Mobile Robots

with Pioneer Server Operating System Software

Pioneer LOGO Programmer's Manual

Pioneer LOGO Programmer's Manual Version 1.0c

Contents

Introduction to Pioneer LOGO

P-LOGO is an interpreted programming language for operation of a Pioneer Mobile Robot or its software simulator. It is an extension of the very popular LOGO programming language originally conceived by Seymour Papert of the Massachusetts Institute of Technology (MIT) and is developed from the LOGO Interpreter and related software from the University of California at Berkeley (UCB-LOGO).

Although mostly transparent to users, P-LOGO is built upon the Pioneer Application Interface (PAI) and Saphira, which are libraries of C functions and definitions for control of a Pioneer Mobile Robot and its simulator. Barry Werger of the University of Southern California built PAI; Saphira and the Pioneer simulator are licensed software from SRI International, Inc., written and maintained principally by Dr. Kurt Konolige. LOGO is not intended to replace PAI or Saphira, but to provide interactive, easy access and operation of a Pioneer Mobile Robot.

Where to get P-LOGO

The Windows95/NT version of P-LOGO comes on disk with every Pioneer Mobile Robot. Updates and other platform-based versions are available online from the *Activ*Media Internet website. The compressed archives contain all the LOGO executables and accessory files you need to run P-LOGO. And, the distribution contains the Pioneer simulator, so you do NOT need to own a Pioneer robot to program with P-LOGO.

`http://robots.activmedia.com/LOGO`

Be sure to choose the P-LOGO version that matches your computer's operating environment; `plogowin.exe` for Windows95/NT computers, for example.

Related Resources

Pioneer Manuals

LOGO for Pioneer depends on a working knowledge of the Pioneer Mobile Robot, its simulator, and its accessories—how to turn it on, enable the motors, connect the radio modems, and so on. Consult your *Pioneer Operation Manual* for details.

Obtain copies of the latest Pioneer manuals (this document, too) from our Internet website:

`http://robots.activmedia.com/docs`

Newsgroups

We announce Pioneer software-related updates and new versions, as well as share ideas and code, through two main email-based newsgroups:

`pioneer-users@activmedia.com`

`saphira-users@activmedia.com`

To join—and please do join—simply send an email message (substitute `saphira-users` for `pioneer-users` to access that separate group):

```
To: pioneer-users-request@activmedia.com
From: <your return email address goes here>
Subject: <choose one command>
   help   (returns instructions)
   lists  (returns list of newsgroups)
   subscribe
   unsubscribe
```

Our SmartList-based listserver will respond automatically. Once subscribed, send your email comments, suggestions, and questions intended for the worldwide community of Pioneer users:

```
To: pioneer-users@activmedia.com
From: <your return email address goes here>
Subject: <something of interest to all members of pioneer-users>
```

Access to the pioneer-users newslist is limited to subscribers, so your address is safe from spam. However, the list currently is unmoderated, so please confine your comments and inquiries to that concerning Pioneer operation and programming.

Support

If something seems (or clearly is) broken with the software or your Pioneer robot, send an email message to

`pioneer-support@activmedia.com`

and a team of experts will leap to the rescue.

Deciphering the Codes

Throughout this manual, we try to give text-style clues to command syntax and so forth. Pathnames, for instance, appear in a monospaced font, and the component directory names each have a forward slash to separate them from other directories and file names.

Commands have two components: A single command word followed by none, one, or more parameters, each separated by a space. Throughout, we present the command word in `monospaced` **`bold`** text. Explicit parameters appear in the same `monospaced font` as its corresponding command, but in regular style, whereas implicit parameter specifications—usually with units—appear *`italicized`*. Options are separated by a vertical bar (“|”); choose `one|other`. Beneath the command definition, we usually include some explanation, in regular text.

Here are some examples:

`gripup`

Raise the Pioneer Gripper to its upper-most and closed position.

`forward|fd` *`distance`*

Drive the robot forward the specified *distance* in centimeters. So, `forward 50` and `fd 50` mean the same thing and tell the robot to move forward 50 centimeters.

`xmodled` `"right|"left|"side "on|"off`

Switch the selected LED on or off. For example, `xmodled "right "on` (sorry, but the quote marks are necessary), means to turn on the LED which is on the right side of the Gripper/Experimenter's Module.

Programming with P-LOGO

Installation

P-LOGO comes in several versions, one of which should match your computer system—`plogowin.exe` for a PC running the Windows95/NT operating system, for example. Each distributed version is a compressed archive containing the logo executable, associated data files, and the Pioneer simulator. Obtain the software as described in the Introduction of this document.

Windows95/NT

Either insert the distribution 3.5-inch floppy disk in the drive and open the disk from your Windows95/NT desktop, or put the downloaded copy of `plogowin.exe` in the root directory of your boot drive (usually C:). Locate the `plogowin.exe` program and execute it from either the Start menu or by double-clicking its icon from the desktop. The self-extracting archive automatically creates a p-logo\ directory (or one which you specify during installation), inside which are `p-logo.exe` and `pioneer.exe`, the Pioneer-LOGO and robot simulator executables, respectively.

The P-LOGO installation program also lets you specify a text processor, `notepad.exe`, for example, from which you may edit your LOGO programs.

Linux and UNIX

Place `plogo-`<system>.tgz into `/usr/local/` (you will probably need superuser access to do this). Then,

```
% tar -zxvf plogo-<system>.tgz
```

The “%”, of course, is the shell command prompt; don’t type it. It’s also a good idea to give users easy access to p-logo and the Pioneer simulator programs by including `/usr/local/bin/` in their executables PATH.

P-LOGO Directory Structure

Once uncompressed, the fresh p-logo directory should contain these following files and subdirectories:

```
/usr/local/p-logo/ or C:\p-logo\
           README (abbreviated installation guide)
           logoman.pdf (this doc in Adobe Acrobat format)
           userman (UCB LOGO text documentation)
           logo (or p-logo.exe; yup, this is the LOGO interpreter)
           pio_exe (or pioneer.exe; pioneer simulator)
           logolib/  (library of canned logo routines)
           helpfiles/ (collection of LOGO commands with explanations)
           worlds/ (simulated worlds for the simulated robot)
           params/ (robot-specific parameters)
           csls/ (LOGO examples)
```

In UNIX/Linux systems, we also place executables into `/usr/local/bin/`:

```
  /usr/local/bin/
           p-logo (csh script sets environment and starts P-LOGO)
           pioneer (csh script sets environment and starts simulator)
```

In Windows95/NT systems, we also put some special library files in the main P-LOGO directory::

```
  C:\p-logo\ (Windows95/NT only)
           sf-p.dll (the Saphira dynamic-load library of utilities)
           msvcrt.dll (a required MSW library)
           msvcrt40.dll (another required MSW library)
```

Running with P-LOGO

You may use P-LOGO with either a real Pioneer Mobile Robot or with its software simulator. We presume you have successfully attached and operated the robot and/or its simulator from the computer on which you plan to run LOGO. Consult the *Pioneer Operations Manual* for details. You may start up the Pioneer robot or simulator at any time after starting P-LOGO, but *before* you actually connect to the robot or simulator from within the LOGO environment.

Starting P-LOGO Windows95/NT

Locate and execute the `p-logo.exe` program from the Start menu. Or locate the p-logo program icon in the `p-logo\` directory in which you installed the software and double-click it with the mouse.

Starting P-LOGO UNIX/Linux

P-LOGO for UNIX and Linux systems requires the X-Window System. Start X and open a shell command window. Then, either locate the p-logo(.exe) icon on your GUI desktop and double-click it with the mouse, or enter the following from the UNIX/Linux command line:

```
% /usr/local/bin/p-logo  (UNIX/Linux only)
```

If your friendly system administrator was thoughtful enough to include `/usr/local/bin/` in your executables PATH, you could also have just typed "p-logo" to achieve the same effect.

Connecting with the Pioneer 1 Robot

To operate the Pioneer 1 Mobile Robot or its simulator from P-LOGO, you must first establish a connection. To connect with the physical robot, issue the `startrobot` command from the P-LOGO command prompt:

```
startrobot "connection_port
```

The command connects P-LOGO to the Pioneer robot through the indicated connection port (opening quote mark required). For all systems, the connection ports are `"serial1` or `"serial2`, which refer to COM1 and COM2, respectively, for Windows95/NT PCs, or to the appropriate /dev /cuax or /dev/ttyx port on UNIX/Linux systems. You may also specify another serial communication device directly by name; `"/dev/cua3` for a UNIX serial port, for example.

You must have a licensed version of P-LOGO to connect with a Pioneer Mobile Robot, not just its simulator.

Connecting with the Pioneer Simulator

The Pioneer simulator and its accessory simulated worlds and robot parameter files come bundled with P-LOGO. The simulator executable is named `pioneer(.exe)` and is found in the `plogo\` directory of Windows95/NT-based systems, or in `/usr/local/bin/` directory on UNIX and Linux systems. The `worlds/` and `params/` are in the `/usr/local/p-logo/` Linux/UNIX directory or in the `plogo\` directory of Windows95/NT. Anyone may use P-LOGO with the Pioneer simulator.

Start the Pioneer simulator from the Windows Start menu or by double-clicking the `pioneer.exe` icon in the p-logo directory, or issue the following command from your UNIX/Linux command shell (*not* from within P-LOGO):

```
 % /usr/local/bin/pioneer &       (UNIX/Linux only)
```

If your UNIX/Linux system administrator is truly friendly, he or she may have put `/usr/local/bin/` in your executables PATH, so the command `pioneer` alone may be sufficient to launch the simulator.

The Pioneer simulator should appear as a GUI-based window on your computer screen. Operate its various pull-down menus to make the simulated robot visible, to load a simulated world, and so on, as described in the *Pioneer Operation Manual*.

To connect with the Pioneer simulator, issue the following command at the P-LOGO prompt:

```
startrobot "simulator
```

which, not surprisingly, connects P-LOGO with the Pioneer simulator (opening quote mark required).

Connection Errors

If the robot or simulator is not properly attached and on, or if you select the wrong port or device for connection, P-LOGO will tell you that the connection is "false". Sometimes the robot itself needs to be restarted, so you might try resetting or cycling its power. Also, with UNIX/Linux systems, make sure you have the proper permissions to access the connection port and execute the proper device drives and applications.

And again, to connect with a physical robot, you need to obtain a licensed version of P-LOGO.

Disconnecting from the Robot or Simulator

When finished working with the robot or simulator, but *before* you exit P-LOGO, you should disconnect.

```
shutdownrobot
```

The `shutdownrobot` command disconnects P-LOGO from the robot or its simulator. The Pioneer robot should stop moving and stop “pinging” its sonars. If not, you may have to reset it. Once disconnected, you may reconnect to the same or to another robot or simulator from the P-LOGO command line.

Exiting P-LOGO

Although not required, you should be graceful when finished running P-LOGO: First disconnect from any connected robot or simulator, then from the command prompt, enter:

```
bye
```

Interacting with P-LOGO

There are many excellent textbooks available on the use of the LOGO language, including *Computer Science LOGO Style* and others by the developers of UCB-LOGO, on which P-LOGO is based. This manual can only present (briefly at that!) the details of the implementation specific to P-LOGO.

P-LOGO Command Primer

To issue a P-LOGO command, simply type the command word next to the P-LOGO prompt along with any parameters (command and each parameter separated by a space) and press the Enter or Return key. To see a list of available P-LOGO commands, for example, type the word "help" next to the P-LOGO prompt and press the Return or Enter key:

```
? help
```

(Don't type the "?" prompt, of course.)

The command will be executed immediately. If a command you type at the prompt returns a value, P-LOGO responds with that value and a message that it does not know how to use it, for example:

```
2? sqrt 9
I don't know what to do with 3
```

You may also use the values returned from commands as parameters to other commands:

```
? sonardist sqrt 9
I don't know what to do with 123
```

LOGO makes assumptions about parameters based on an order of operations; parameters are evaluated before the command itself is invoked. This can be tricky; the command:

```
? sonardist 4 + sonardist 3
```

will not return a sum of two sonar readings, but will attempt to return a reading from a sonar numbered 4 plus the value of `sonardist 3` (for example, 126 - an error as there are only 7 sonar pingers on the Pioneer). In cases where you must override the order of operations, or whenever the meaning may be ambiguous, it is a good idea to parenthesize commands. The proper response for our example can be generated by:

```
? (sonardist 4) + (sonardist 3)
```

You may enter several commands on a single line, each separated by a space. P-LOGO executes the commands in order of appearance, but not until you press the Return or Enter key to execute them.

If you want to put a long, single command on several lines, end each line except the last one with the tilde (~) character. For example:

```
? ifelse (sonardist 1) > (sonardist 2)~
    [fd 100]~
    [rt 90 fd 100]
```

Parenthesized functions and bracketed lists of commands do NOT strictly need the tildes for continuation, but in practice it is a good habit to always use the them to build multi-line commands.

Commands are not case sensitive, although parameters may be case sensitive. Hence, forward, FORWARD, and FoRwArD are all the same command to move the onscreen LOGO graphic turtle or Pioneer robot or simulator.

When in immediate-execution mode (not entering a P-LOGO program; prompt is "?"), the robot or LOGO system immediately will perform your command. When in programming mode (prompt is ">"), your P-LOGO command won't work until you exit programming mode and execute the program that contains the command.

You may enter several commands on a single line, each separated by a space. P-LOGO executes the commands in order of appearance, but not until you press the Return or Enter key to execute them immediately in immediate-command mode, or when you run the program in which you've stored the series of commands.

When you execute any graphics- or robot-related command for the first time, P-LOGO automatically will open and display the LOGO graphic turtle—an onscreen robot agent. P-LOGO will not, however, send your command to the Pioneer robot or simulator unless you have established a connection with it, nor will P-LOGO automatically establish that connection.

P-LOGO Parameters

P-LOGO command parameters come in at least three flavors: names, variables, and constants. Constants are the simplest parameters to include in your programs: Simply type a value or word. For instance, to make the LOGO turtle move forward 100 centimeters, you simply type the command and the constant parameter on the command line:

```
? forward 100
```

A variable is a name that has an associated, but changeable value. By definition, you can change a variable's value, and in doing so, readily change the behavior of your programs. For instance, we could set the value of a distance variable called "dist" and use it to move the robot forward that variable distance:

```
? make "dist 100
dist
? forward :dist
```

In the previous example, you create the variable dist with `make` and give it a value of 100. P-LOGO responds with the variable name after *make*. You then refer to the variable by its given name, preceded with a colon (":"); `:dist`, for example, which when used in the example command above, tells the robot to move 100 centimeters forward.

Finally, *name* P-LOGO parameters are for programs and routines, and sometimes for variables, as in the parameter of `make` in the example above. Some names already exist as part of the LOGO system itself, such as all the P-LOGO commands themselves. You may create your own names, such as the name you give to your P-LOGO programs.

When using a name as a parameter in a P-LOGO command, always precede it with a single, open-quote mark ("). For example,

```
? help "forward
```

tells P-LOGO to give you some detailed information about the forward command. If you didn't include the preceding double-quote mark, P-LOGO would tell you that there are, "not enough inputs to forward." Ugh.

P-LOGO Programming Mode

`to` *`program-name parameter(s)`*

Enter P-LOGO programming mode and declare any variables.

The **`to`** P-LOGO command puts you into programming mode, identified by the ">" command prompt. In programming mode, you enter a series of commands that are not executed right away, but rather are stored in memory with the given program name.

Enter the series of commands, several per line if you like. When done, finish the program with the `end` command:

`end`

End P-LOGO programming mode.

For example, this program drives the robot in a square. The program “square” gets composed and stored in memory, and does not get executed until you command P-LOGO to do so—the very last entry in the following:

```
? to square :dist
> fd :dist right 90
> fd :dist right 90
> fd :dist right 90
> fd :dist right 90
> end
? square 50
```

P-LOGO Program Editing

Under most operating systems, P-LOGO provides the ability to edit new or previously defined programs in an editor more convenient than the command-line interface. This is done with the edit command:

```
? edit "square
```

will open the editor with the current definition of `square` displayed if it exists, and a blank program template if the specified function has not yet been defined.

Pioneer-Related LOGO Commands

P-LOGO contains many Pioneer robot-related extensions to the standard LOGO language, as implemented by UCB LOGO. This section describes mostly only those Pioneer-related extensions, *not* the wealth of commands supported by UCB LOGO and included in P-LOGO. Please consult the many fine LOGO textbooks available on the subject, the UCB-LOGO user manual found as a formatted text file inside the `p-logo/` directory, and avail yourself of the `help` command and its useful features that accompany the language online.

Also, please be aware at the outset that real robots do not exist in the perfect world of their computer-simulated counterparts. Even though it has very accurate position detectors, real world things like wheel slippage, as well as deliberate disturbances like people kicking or shoving the physical robot, all affect the performance of any real robot, including Pioneer. Good educators will point out that the real world is not as ideal or as perfect as computers would like it to be, which is precisely why a real robot is so valuable, compared with its simulated counterparts.

Motion Controls

P-LOGO's motion commands fall into three rough categories: distance-based, velocity-based, and rotational motion. Distance and rotational motions are discrete; that is, they cause a translation or rotation of a specified distance, much as standard LOGO commands do. Velocity-based motions are continuous; the robot continues moving until told to stop in a manner that has no analogue in standard LOGO. While the turtle graphics on the screen will accurately reflect the robot's motion resulting from discrete commands, they will be unpredictable when you use one of the continuous commands. Certain safety features attempt to keep the Pioneer from running into anything while moving, but be aware of the distinctions and possible dangers.

Issuance of any motion-related command causes the LOGO turtle to appear on screen, if it isn't already visible. On the other hand, the P-LOGO motion commands *do not* automatically make a connection with the Pioneer robot or simulator. If not connected, the robot or simulator will not respond to a P-LOGO motion command, even though the onscreen Turtle does.

Distance-Based Motion Commands

`forward|fd` *`distance`*
`back|bk` *`distance`*
`movevel`
`setmovevel` *`velocity`*

Driving the robot forward and backward are fundamental motion commands (distance is the specified distance in centimeters). If stopped, the robot automatically accelerates to a set velocity until it nears the distance goal, whereupon it deaccelerates and stops when it has achieved the distance.

The forward|back moving velocity is originally set to around 20 centimeters per second. You can reset it to make the robot go faster or slower with the `setmovevel` command (*velocity* is the speed you want in centimeters per second). The minimum velocity to get the robot moving at all is around five centimeters per second; top speed depends on the robot. The Pioneer 1's top speed is 50 centimeters per second; the AT can move over 800 centimeters per second. Use the `movevel` command to find out what the current forward|back moving velocity is set to.

Velocity-Based Motion Commands

`setvel` *`velocity`*

Whereas forward and back tell the robot to drive a certain distance, the `setvel` comand tells the robot to (de)accelerate to the specified *velocity,* and keep going at that speed in centimeters per second. And unlike forward or back which make you wait for the robot to complete the command before accepting another command, P-LOGO after the `setvel` command goes on immediate to accept other commands. In fact, the robot won't stop until, and only until, you tell it to stop moving (below), you dramatically shut down P-LOGO, or if the robot runs into something or stalls (see next section). So, be careful.

`stopmoving`

Stops a `setvel` moving robot. Note that you cannot issue this command to stop a forward or backward moving robot.

Collision Avoidance and Stall Detection

An example of Pioneer's intelligence is its built-in collision-avoidance and stall behaviors. When moving forward, the robot automatically will avoid collisions by stopping and telling you, "Something is *not* mean that the real robot always will stop in time to avoid a collision, although the simulated Pioneer usually does stop—it depends on the speed at which the robot is traveling and its sensitivity to oncoming objects. Shins may get bruised, for instance, if someone jumps close in front of a fast-moving Pioneer, or if you've set the safety distance too short relative to the robot's moving velocity. Also, when traveling backwards, or sometimes when on an oblique trajectory relative to an obstruction, the robot may not "see" an obstacle so to avoid it. Even the simulated robot can crash into walls. Good

When the robot does run into something immoveable, or somehow gets stopped when it wants to go, such as trying to climb a steep hill, its drive motors will stall. Even if the robot cannot "see" what is stalling its motors, it will complain, "Something is in my way." Whenever you get the "Something is in my way" message, P-LOGO automatically stops moving forward or back (distance-based motion commands). With velocity-based motion (`setvel`), however, the robot may start moving again once the obstruction is removed.

Adjust the robot's sensitivity to obstructions and stalling with the following commands:

`sonarsafetydist`

Reports the sonar-detected distance (in centimeters) at which nearby objects trigger the robot's collision avoidance behavior. Note that the robot pays attention only to the five forward-facing sonars for collision avoidance; P-LOGO ignores the side sonar readings in this case, and the robot has none facing backwards.

`setsonarsafetydist` `distance`

Change the distance in centimeters at which objects will trigger Pioneer's collision avoidance.

`stalled`

Report whether ("true) or not ("false) the motors are stalled. Only a moving robot can be stalled. (The Pioneer AT can never appear stalled since it lacks the proper electronics.)

Turns, Headings, and Positions

P-LOGO maintains information about position and heading of the robot (when it is connected) or of the screen turtle (when Pioneer is not connected). The position (0,0) and heading of zero degrees are the position and heading of the robot at the time you make a connection. Or, it is the center of the screen, facing upwards, for the screen turtle. This global origin can be reset to the robot's current position with the `resetpos` command. While the screen turtle's position will maintain accuracy when the robot is not connected, accuracy of the position of the physical robot will degrade over time due to cumulative dead-reckoning error.

`left|lt|right|rt` *`degrees`*

Turn the robot counterclockwise (left) or clockwise (right) the specified number of degrees (mod 360).

`heading`

Report the clockwise angle in degrees of the robot's current heading.

`setheading|seth` *`degrees`*

Rotate the robot to the specified heading in degrees (0-359) relative to its starting position.

`pos`

Report where the robot thinks it is relative to where it started in 2-D space in ±X,Y millimeters.

`xcor|ycor`

Report the current position of the robot; ±X or ±Y centimeters away from the starting position (0,0).

`resetpos`

Reset the robot’s local map so that its heading is 0 degrees and it is at the origin (0,0).

Sonar Ranging

The Pioneer Mobile Robot has 7 sonar pingers; five on the front and one on each side. They are numbered 0 to 6 (left-to-right), and point at 90, 30, 15, 0, -15, -30, and -90 degrees relative to the robot's heading, respectively.

Sonars work by periodically emitting a finely tuned, high-pitched click and "listening" for any echo that may return when deflected off a surface nearly. Each sonar determines how far away the sound-reflecting surface is by the time it takes its "ping" to return (if ever).

The Pioneer robot and its simulator automatically and routinely fire the sonars and report each sonar's (0-6) range findings in centimeters; the minimum reported range is about 20 centimeters and the maximum range is 500 centimeters. The ranges are subject to distortion due to the surface angle and sound reflectivity of detected objects.

`sonardist` *`pinger`*

Report distance in centimeters to the nearest object detected by the specified pinger. A reading of 500 indicates that no object was detected.

I/O Commands

The standard Pioneer robot comes with eight digital input/output ports, one analog-to-digital port, and a timer, which you may use to operate your custom robotics accessories and options. P-LOGO gives you a way to control and read those ports from software.

Unfortunately, the current implementation of P-LOGO displays and responds to decimal values that correspond to digital bit/byte patterns, so you need to do some quick calculations to set or read a single bit or decode the state of bits from a composite byte.

Also, be aware that some of Pioneer’s optional accessories use and expand some of these I/O ports for their own functions. In particular, the Gripper and Experimenter’s Module support different, more general P-LOGO commands that may override an explicit I/O command. Consult the next Section and the respective manuals for details if you own and wish to program the Gripper and/or Experimenter’s Module from P-LOGO.

`analog`

Reports the current value (0-255 for 0-5VDC) connected to the analog-to-digital port. With the Experimenter’s Module, the single A/D port is multiplexed into eight (AN0-7). To read a particular ANx port, first select the port number by setting the bit pattern for OD5-7, then read the analog port. See `digout` and `setdigout`.

`digin`

Reports the composite byte value (0-255) on the state of the eight digital input ports.

`diginbit` *`ID`*

Reports the bit value (0 or 1) of the selected digital input port (ID number 0-7).

`digout`

Reports the composite byte value (0-255) on the state of the eight digital output ports.

`digoutbit` *`OD`*

Reports the bit value (0 or 1) of the selected digital output ports (OD number 0-7).

`setdigout` *`byte`*

Sets the digital output ports (0-7) to the respective state on (1) or off (0) as specified by the bit pattern of the byte (0-255). For example,

`setdigout 255`

turns all eight output ports on, whereas

`setdigout 4`

turns on output port 3 and all the rest off.

Table of bits and decimal equivalents.

Bit number	0	1	2	3	4	5	6	7
Decimal equivalent (bit set to 1)	1	2	4	8	16	32	64	128

To set bits 1, 4 and 5, for example,

`setdigout 50` `(2+16+32=50)`

`setdigoutmask` *`mask byte`*

Set only those digital output ports (0-7) selected by a respective on (1) bit in the decimal mask (0-255) either on (1) or off (0) as specified in the bit pattern of the byte (0-255). For example,

`setdigoutmask 255 0`

turns off all eight output ports, whereas

`setdigoutmask 4 0`

turns off only output port 3 because the mask specifies only that port bit (other output ports remain at their original states).

`setdigoutbit` *`OD`* `0|1`

Sets the digital output port number 0-7 on (1) or off (0).

`starttimer` *`pin`*

Starts the microsecond timer and trigger for an event on the specified pin.

`inputtimer`

Reports the timer value.

`setpulsewidth` *`pulse`*

Sets the width of the PTU pulse to the specified pulse in milliseconds.

Gripper & Experimenter's Module Commands

Pioneer has an optional Gripper attachment that comes with an Experimenter's Module (also sold separately). Their functions are controlled by Pioneer's various I/O ports. Use the following explicit P-LOGO commands to control the Gripper and certain features of the Experimenter's module. You may also use the specific I/O port commands detailed in for much finer control. Consult the *Gripper & Experimenter's Module Manual* for details.

`gripup|gripdown`

Puts the Gripper in its fully up and closed or fully down and open position. On its way up, the Gripper will grab and lift an object that is within its paddles' grasp, and will lower and release the object on its way down.

`gripcarry`

Raises or lowers the Gripper to a mid-position. Grasps an object within the paddles if starting from the `gripdown` position, but does not release an object if starting from the `gripup` position.

`gripopen`

Same as `gripdown`. There is no way to open the Gripper while it is up.

`gripcontact`

Reports if either or both of the bump switches on the front of the paddles is pushed (true) or not (false).

`gripfrontbeam|griprearbeam`

Reports if something is (true) between the front or rear breakbeam in the Gripper paddles, or not (false). This is, of course, the way you know if there is an object between the Gripper paddles.

`gripstate`

Reports the current state of the Gripper: `"upclosed` (at the top and paddles closed); `"downopen` (fully down with paddles open); `"carry` (paddles closed and part-way up); `"moving` (yeah, that's right—it's going to some position up, down or carry); or `"stopped` (something has jammed the Gripper).

`pickup|drop`

Tells the Gripper to pick up any object that may come within its paddles. This command stays in effect until the robot encounters and picks up an object, or until you tell it to `drop`, at which point it gently puts down the object it is carrying or, if it hadn't already found one, just stops looking for a pickup.

`xmodbutton`

Reports if the pushbutton switch on the Experimenter's Module is pressed (`"true`) or not (`"false`).

`xmodswitch`

Reports if the slide switch on the Experimenter's Module is `"up` or `"down`.

`xmodled "right|"left|"side "on|"off`

Turns on or off the selected LED on the Experimenter's Module (opening quotation marks are required).

Miscellaneous & Enigmatic Commands

`robotrunnningp`

Reports whether ("true) or not ("false) a connection exists between P-LOGO and the robot or its simulator.

`robotstatus`

Reports the robot's current status (a byte value which has some cryptic meaning not elaborated here).

`voltage`

Reports the robot's current battery charge in volts.

Appendix. P-LOGO Quick-Reference

`analog`

Reports the current value (0-255 for 0-5VDC) connected to the analog-to-digital port.

`back|bk` *`distance`*

Drives the robot backward the specified distance in centimeters.

`digin`

Reports the composite byte value (0-255) on the state of the eight digital input ports.

`diginbit` *`ID`*

Reports the bit state (true or false) of the selected digital input port (ID number 0-7).

`digout`

Reports the composite byte value (0-255) on the state of the eight digital output ports.

`digoutbit` *`OD`*

Reports the bit state (true or false) of the selected digital output port (OD number 0-7).

`drop`

Tells the Gripper to gently put an object down that it may be carrying within its paddles.

Also cancels the `pickup` command.

`forward|fd` `distance`

Drives the robot forward the specified distance in centimeters.

`gripcarry`

Raises or lowers the Gripper to a mid-position.

`gripcontact`

Reports if either or both of the bump switches on the front of the paddles is pushed (“true) or not

`gripdown`

Puts the Gripper in its fully down and open-paddle position.

`gripfrontbeam|griprearbeam`

Reports if something is between the front or rear breakbeam in the Gripper paddles ("true), or not ("false).

`gripopen`

Same as gripdown. There is no way to open the Gripper while it is up.

`gripup`

Puts the Gripper in its fully up and closed position.

`gripstate`

Reports the current state of the Gripper.

`heading`

Reports the clockwise angle in degrees of the robot's current heading.

`inputtimer`

Reports the timer value in microseconds.

`left|lt` *`degrees`*

Turns the robot counterclockwise (left) the specified number of degrees (mod 360).

`movevel`

Reports current moving velocity in centimeters per second.

`pickup`

Tells the Gripper to pick up an object that comes within its paddles. This command stays in effect until the robot encounters and picks up an object or until you tell it to `drop`.

`pos`

Reports where in the robot thinks it is relative to where it started in 2-D space in ±X,Y millimeters.

`resetpos`

Resets the robot’s local map so that its heading is 0 degrees and it is at the origin (0,0).

`right|rt` *`degrees`*

Turns the robot clockwise (right) the specified number of degrees (mod 360).

`robotrunnningp`

Reports whether ("true) or not ("false) a connection exists between P-LOGO and the robot or its simulator.

`robotstatus`

Reports the robot's current status (a byte value which has some cryptic meaning not elaborated here).

`setdigoutbit` *`OD`* `0|1`

Sets the digital output port number 0-7 on (1) or off (0).

`setdigoutmask` *`mask byte`*

Sets the selected (bit set to 1 in mask byte; 0-255) on (1) or off (0) specified by byte (0-255).

`setheading|seth` *`degrees`*

Rotates the robot to the heading in degrees relative to its starting position.

`setmovevel` *`velocity`*

Sets the velocity for the moving (forward or backward) robot in centimeters per second.

`setpulsewidth` *`pulse`*

Sets the width of the PTU pulse to the specified pulse in milliseconds.

`setsonarsafetydist` `distance`

Sets the distance in centimeters at which objects will initiate collision avoidance.

`setvel` *`velocity`*

Tells the robot to move continuously at the specified speed in centimeters per second. Only stopmoving will stop a setvel moving robot.

`sonardist` *`sonar`*

Reports distance in centimeters that the specified sonar (0-6; 7 is enabled, but not working) detects to the nearest object.

`sonarsafetydist`

Reports the current sonar distance to nearby objects that will effect collision avoidance; in centimeters.

`shutdownrobot`

Disconnects P-LOGO from the connected Pioneer or simulator.

`stalledp`

Reports whether (true) or not (false) the motors are stalled. Only a moving robot can be stalled.

`startrobot` *`devicename`*

Connects P-LOGO to the Pioneer robot through `"serial1`, `"serial2`, or through another specified port .

`startrobot` `"simulator`

Connects P-LOGO to the Pioneer simulator.

`starttimer` `pin`

Starts the microsecond timer and trigger an event on the specified pin.

`stopmoving`

Stops a `setvel` moving robot.

`voltage`

Reports the robot's current battery charge in volts.

`xcor|ycor`

Reports the current position of the robot; ±X or ±Y centimeters away from the starting position (0,0).

`xmodbutton`

Reports if the pushbutton switch on the Experimenter's Module is pressed ("true) or not ("false).

`xmodled` `"right|"left|"side "on|"off`

Turns on or off the specified LED on the Experimenter's Module.

`xmodswitch`

Reports if the slide switch on the Experimenter's Module is `"up` or `"down`

Index

Warranty & Liabilities

The developers and marketers of P-LOGO, PAI and Saphira software shall bear no liabilities for operation and use with any robot or any accompanying software except that covered by the warranty and period. The developers and marketers shall not be held responsible for any injury to persons or property involving the P-LOGO, PAI, or Saphira software in any way. They shall bear no responsibilities or liabilities for any operation or application of the software, or for support of any of those activities. And under no circumstances will the developers, marketers, or manufacturers of P-LOGO, PAI, or Saphira take responsibility for or support any special or custom modification to the software.

ActivMEDIA
INCORPORATED

182 Hancock Road
Route 202 North
Peterborough, NH 03458
(603) 924-9100
(603) 924-2184 fax
http://www.activmedia.com/robots