ActivMEDIA INCORPORATED

Pioneer

Mobile Robots

with Pioneer Server Operating System Software

Operation Manual

2nd Edition

Pioneer Mobile Robot Operation Manual, Edition 2, January 1998

Contents

Congratulations

on your purchase and welcome to the rapidly growing community of researchers, developers, and enthusiasts of the Pioneer Mobile Robot. This *Pioneer Operation Manual* provides the general and technical details you will need to operate your robot and to begin developing your own robotics hardware and software projects.

We encourage you to also use these companion resources that come with your Pioneer:

- Saphira, Pioneer Application Interface, and P-LOGO software and manuals
- Personal Account for the `http://robots.activmedia.com` Internet server
- Pioneer-user support newsgroups

Pioneer Package

Our experienced robotics manufacturing staff put your Pioneer Mobile Robot and accessories through a "burn-in" period and carefully tested them before we shipped the robot to you. Our care extends beyond: Besides the companion resources listed above that bring the whole community of Pioneer to you, the manufacturer warranties the robot against mechanical and electronic parts and labor defects for one year. All of these precautions ensure that you have many years to enjoy your new Pioneer Mobile Robot.

Even though we've made every effort to make your Pioneer package complete, please check the components once again after you unpack it from the shipping crate.

Basic Components (all shipments)

- One assembled Pioneer Robot
- Battery charger (some contain power receptacle and 220VAC adapters)
- Floppy 3.5" disk containing licensed copies of Saphira, PAI, and P-LOGO software for Windows95/NT platform
- Two hex wrenches
- Set of manuals
- Registration & Account Sheet

Optional Components and Attachments

- Serial cables for external connections
- Radio modems (DIP switches preset at factory)
    - One mounted inside Pioneer Console (antenna detached for shipping)
    - Companion radio modem (antenna also detached for shipping)
    - Serial DB9 female to DB9 male cable with 25-pin adapter
    - Power supply (120 VAC to 9 VDC; 500 ma) for external radio modem
    - Power receptacle and 220VAC adapters where applicable
    - Radio modem manual
- Replacement battery
- Gripper & Experimenter's Module
- Fast-Track Vision System
- Stereo Vision System
- Wireless Video System
- Experimenter's I/O Module
- Manipulation Bumper and Contact Sensor

User Supplied Components

- Computer: 486-class or better PC with Microsoft Windows95/NT, FreeBSD, or Linux operating system; Power-PC Macintosh with System 7.5 or later; or any UNIX workstation
- One RS232-compatible serial port
- Four megabytes of available disk-storage

Additional Resources

Every new Pioneer customer gets three additional and valuable resources: a private account on our Internet server for download of Pioneer software, updates, and manuals; access to the private `pioneer-users` newsgroup; and direct access to the Pioneer technical support team.

Pioneer Software

We have a World Wide Web server connected full-time to the global Internet where customers may obtain Pioneer software and support materials:

`http://robots.activmedia.com`

Some areas of the Pioneer website are restricted to licensed customers. To gain access, use the username and password that are written on the **`Pioneer Registration & Account Sheet`** that accompanied your robot and this manual.

Pioneer Newsgroup

We maintain a special email-based newsgroup for Pioneer owners to share ideas, software, and questions about the robot. To sign up, send an email message to our automated newsgroup server:

> `To:` **pioneer-users-request@activmedia.com**
> `From:` <***your return email address goes here***>
> `Subject:` <choose one commands:)
> **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**>

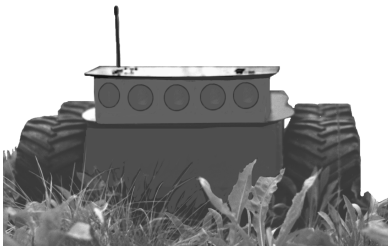
Access to the pioneer-users newslist is limited to subscribers, so your address is safe from spam. However, the list currently is unmoderated, so please confine your comments and inquiries to that concerning Pioneer operation and programming.

Support

Have a problem? Can't find the answer in this or any of the accompanying manuals? Or know a way that we might improve Pioneer? Share your thoughts and questions directly with us:

`pioneer-support@activmedia.com`

Your message goes to our Pioneer technical support team who will help you directly or point you to where you may find help. Because this is a support option, not a general-interest newsgroup like `pioneer-user`, we must reserve the option to reply only to questions about bugs or problems with Pioneer. (See also "Maintenance & Repair" Chapter 9.)



What is Pioneer?

Pioneer actually is two robots: the two-wheel drive Pioneer and the four-wheel drive Pioneer AT. Both are small, intelligent mobile robots developed by Kurt Konolige of SRI International and Grinnell More of Real World Interface, Inc. They are truly off-the-shelf, *plug-and-play*, intelligent mobile robots, containing all of the basic components for sensing and navigation in a real-world environment, including battery power, drive motors and wheels, position/speed encoders, and ultrasonic range finders—all managed via an onboard MC68HC11-based microcontroller (Figures 2-1). Your Pioneer also has a variety of expansion I/O ports for optional and custom attachments, and an RS232 serial port for communication between the robot and other computers.

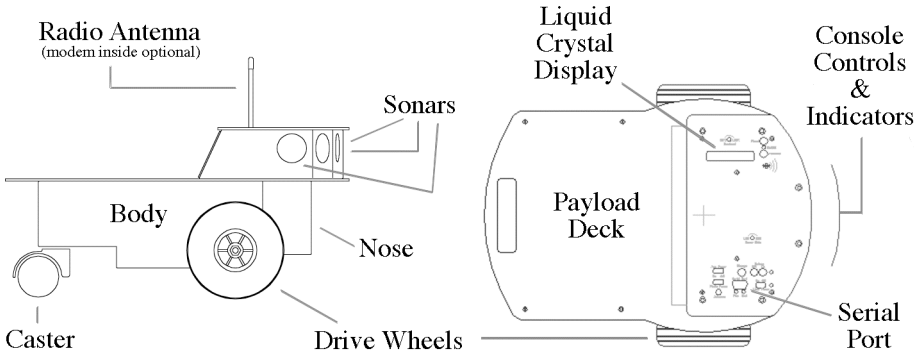


Figure 2-1. Basic components of the Pioneer 1.

And Pioneer comes with a variety of software packages for development and operation of your Pioneer-based robotics programs for personal, research, entertainment, and commercial applications.

The Drives Make the Difference

Except for the drive system and hinged Deck, there is virtually no operational difference between the Pioneer 1 and Pioneer AT ("all-terrain") Mobile Robots. The sonar sensor array and microcontrollers are the same. The accessories available for the Pioneer 1 also work with the Pioneer AT. And applications developed for the Pioneer 1 will work with little or no translation with the Pioneer AT, and vice versa.

Pioneer 1 is the original design. Intended for mostly indoor, hard and flat surface operation, the robot has solid rubber tires and a simple, two-wheel differential, reversible drive system with a rear caster for balance (Figure 2-1).

Intended for operation in uneven indoors and out-of-doors environments, including loose, rough terrain, the Pioneer AT has a powerful and stable four-wheel design with high-traction tires (Figure 2-2). Each of the four AT wheels is powered by a reversible DC motor; each side electronically and physically linked for evenly applied translational and rotational power and speeds. Functionally and programmatically identical with the Pioneer 1, the Pioneer AT has position/speed encoders in its two front wheels and uses differential "skid" steering.

Also, unlike the Pioneer 1, the Pioneer AT features a hinged Deck. Two latches hold the Deck securely in place, but are easily released with fingers for tool-less access to the Body components. In particular, the AT's hinged Deck lets you access and "hot-swap" its battery, also tool-lessly.

Modes of Operation

You may operate Pioneer in one of three modes: self-test, standalone, or server mode.
The basic Pioneer comes equipped with a 32K flash-programmable read-only memory (PROM) on its microcontroller. The PROM contains a self-test software routine that exercises the robot's drive, sonar, I/O, and processing systems (see the "Self Tests" Chapter 5 for details).

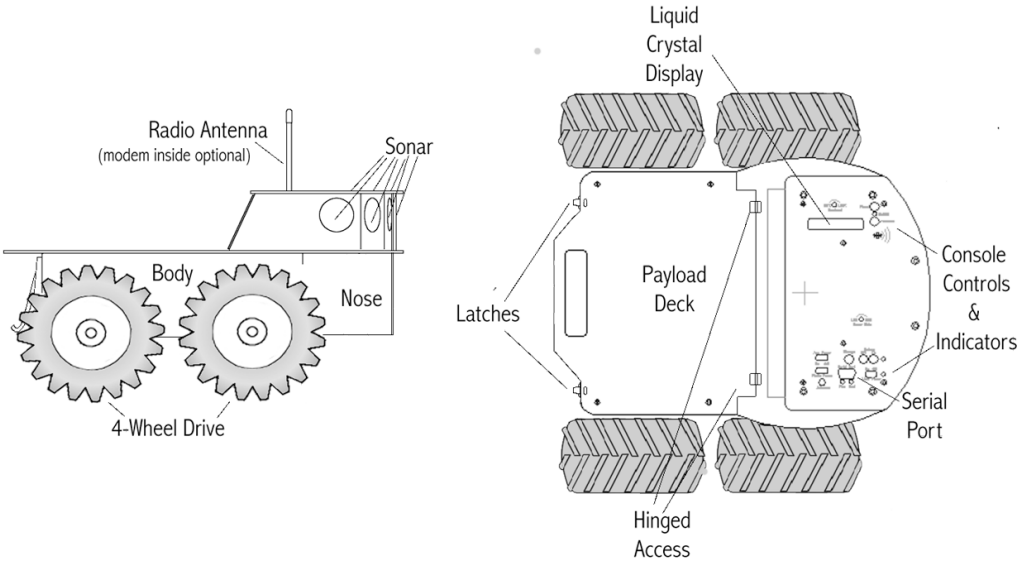


Figure 2-2. Basic components of the Pioneer AT.

The Pioneer PROM also contains the Pioneer Server Operating System (PSOS), which in conjunction with client software like Saphira or the Pioneer Application Interface (PAI) running on a user-supplied computer, lets you take advantage of modern client/server technologies for developing and performing advanced robot tasks (see the "Pioneer Server Operating System" Chapter 6 for more details). Most users run the Pioneer as a server, since it gives them quick and easy access to the robotics functionality through high-level applications software on a familiar host computer.

For experiments in microcontroller-level operation of robotics functions, you may reprogram the onboard PROM or replace it with a 32K static random-access memory (SRAM) chip that can store your custom 68HC11-based software for direct and standalone operation of Pioneer.

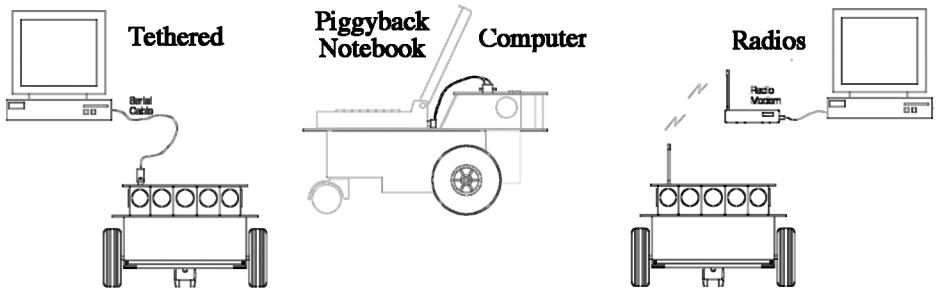


Figure 2-3. Serial communication options between Pioneer and external computers.

Regardless of the mode of operation, the Pioneer Mobile Robot supports basic serial communications between its microcontroller and a client computer via a standard RS232 port. The user-supplied basestation may be a desktop workstation or a piggyback laptop computer connected to the robot through a serial cable or over a radio modem link (Figure 2-3). The serial tether saves power and is much more immune from interference. Radio modems, on the other hand, are much more convenient for autonomous operation, and let more than one computer manage the robot's actions. You may still use a serial tether cable, even though your Pioneer contains a radio modem—just not simultaneously.

Server Mode

Pioneer comes configured as a robot server, both physically and functionally. In this mode, the Pioneer's microcontroller with PROM-based Pioneer Server Operating Systems (PSOS) software manages the low-level functions of the mobile robot, including firing the sonar sensors and retrieving echo signals, running the drive system, collecting position and speed information from the drive encoders, and so on.

The client—software running on either a single basestation, a network of workstations, or on a laptop computer riding piggyback on the robot—performs the high-level mobile robotics functions for the Pioneer. With Pioneer, we provide several client software packages, including the highly advanced suite called Saphira that was developed by Kurt Konolige and his colleagues at SRI International. As Pioneer's companion client software, Saphira handles such compute-intensive tasks as sensor interpretation, navigation planning, and mapping, as well as a variety of artificial intelligence and fuzzy logic-based robotics behaviors.

Saphira also implements a simple robotics-command language—Colbert—for rapid testing, prototyping, and development of applications. See the *Saphira Software Manual* that accompanied your robot for details.

Client Software

Currently available Pioneer client software—for the computing platform of your choice—includes:

- The Saphira client development suite with Colbert
- Pioneer simulator
- Pioneer LOGO (P-LOGO)
- Pioneer Application Interface (PAI)

The Windows95/NT versions of the software with manuals come with the Pioneer. Other versions and all updates for supported computing platforms are available to registered customers for download from our software website (see "Congratulations" Chapter 1):

`http://robots.activmedia.com`

Currently supported systems include most UNIX, Apple Macintosh, Sun Microsystems' SunOS and Solaris, Linux, Silicon Graphics' IRIX, and Microsoft's 32-bit Windows systems.

Saphira comes with a demonstration program that provides for manual (keyboard or joystick) and automatic drive control of the Pioneer, as well as the command-line interactive language, Colbert. The program also lets you enable several built-in robot behaviors, including collision avoidance, features recognition, and self-navigation. See the "Quick Start" Chapter 4 for details.

PAI is a C-language based development suite that works with Saphira. It lets software developers gain closer access to the low-level details of Pioneer server control. See the *PAI Manual* for details.

Pioneer LOGO is a version of UCB-LOGO from the University of California Berkeley, which we've extended to include Pioneer commands. It provides direct, interactive control of the robot through a familiar programming language. See the *P-LOGO Manual* for details.

An important benefit of Pioneer's client/server architecture is that different robot servers can be run using the same high-level client. For example, the included Pioneer simulator runs on the host machine and acts just like the physical robot, so that developers may conveniently perfect their application software, then run it without modification on the physical robot. Several clients also may share responsibility for controlling a single mobile server, leading to experiments in distributed communication, planning, and control.

Standalone Mode

By replacing the server software on the robot, either by reprogramming the onboard PROM or by physically replacing it with 32K of static RAM, you may download your own robot applications through the robot's serial port and run them autonomously without communication with an external computer. In standalone mode, you have direct access to the MC68HC11 microcontroller, leading to focused study of sensors and motor controls and development of dedicated mobile robotics tasks for the Pioneer.

We recommend (but currently do not support) that you create programs for the standalone Pioneer with Interactive C natively, then cross-compile for the MC68HC11 microprocessor and download them to the robot. Available from Newton Labs, Inc. (`http://www.newtonlabs.com`), Interactive C's software development environment has an accompanying library with an extensive collection of software for multitasking, timing operations, and digital and analog I/O in support of your standalone Pioneer software development efforts.

Specifications & Controls

Pioneer robots may be small, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and more expensive machines. Pioneer's modest size, for example, lends itself for navigation in tight quarters or cluttered spaces like a classroom full of desks and students where even adult humans have difficulty traversing. At the same time, the Pioneer server with its Saphira software client is fully capable of mapping its environment, complete with features like doorways and hallways.

Physical Characteristics

See Appendix E for a complete listing and comparison of Pioneer 1's and Pioneer AT's physical and operational specifications.

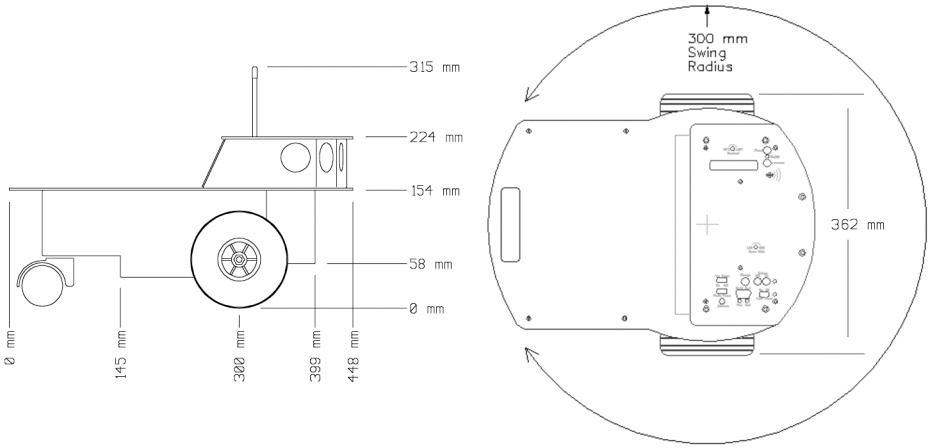


Figure 3-1. Pioneer 1 physical dimensions and swing radius.

The Pioneer Mobile Robots are lightweight (Pioneer weighs only 17 lbs. with battery, for example), yet their strong body materials and construction makes them virtually indestructible and lets them carry a significant payload of up to 10 lbs. (4.5 kg). The lightweight and solid construction also makes Pioneer extraordinary easy to transport—made even easier by a built-in handle at the rear of the main body. And the Pioneer is assembled with Allen hex screws for easy access to interior components and attachment of accessories.

Main Components

The Pioneer is comprised of four main sections: Console, Deck, Body, and Nose.

Console.

The Pioneer Console is the "head" of the robot. It sits on top of the main Deck on the front of the robot, mounted by six hex screws, and houses the Pioneer microcontroller, which is attached to the top plate by six hex screws. You'll also find the Pioneer's controls and indicators mounted to and accessed through the top of the Console. The optional radio modem sits inside and the Pioneer video cameras sit atop the Console, too.

The Pioneer's sonar sensors are attached to the interior front and sides of the Console, but nearly flush with its surface—one on each side and five facing forward at 15-degree intervals.

A rear panel on the Console may be removed (two hex screws) for modest access to the Console internal contents and connectors. There also are special versions of the rear panel for accessory power and other connectors for Pioneer accessories.

Deck.

The Deck of Pioneer is specially designed to carry a notebook computer. Unlike other mobile robot designs, a notebook computer on the Pioneer deck is fairly well protected, sitting low and mostly within the sensing region of the Pioneer's sonars.

The Pioneer AT's deck is hinged and latched for easy, tool-less access to the body contents, including the battery. Latches at the rear of the Deck hold it securely in place, and are easily released by finger. Once opened, the AT's Body contents are easily viewed and modified.

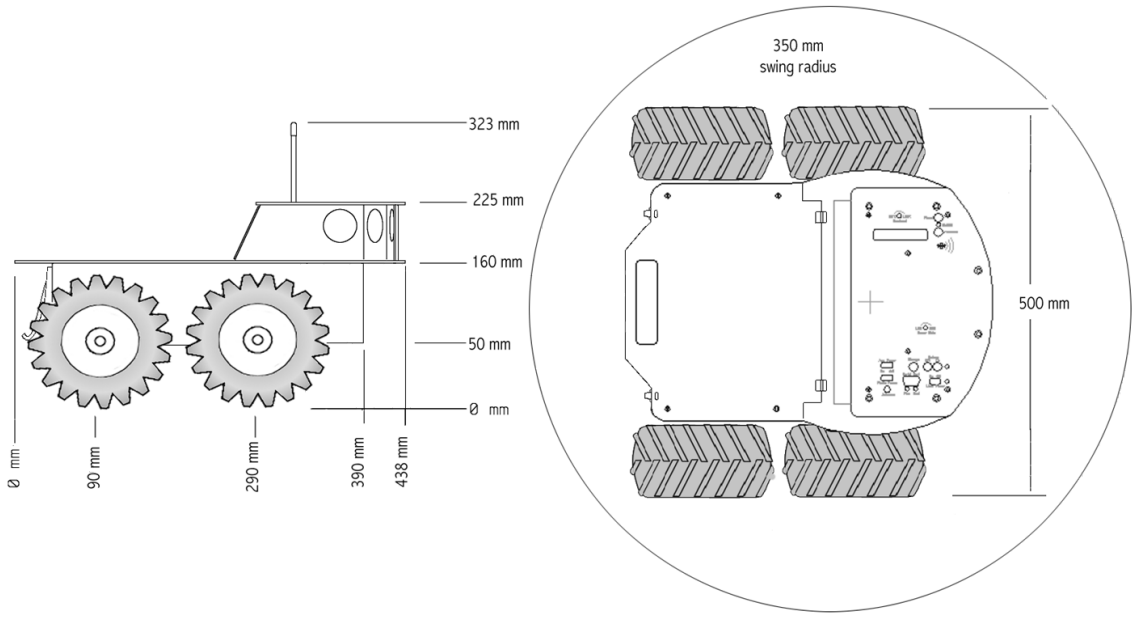


Figure 3-2. Pioneer AT physical dimensions and swing radius.

Body.

The Body of Pioneer contains its battery and the drive motors (two for Pioneer and four for Pioneer AT; front motors with encoders). There is sufficient room in the Body to contain a variety of robotics accessories, such as the Fast-Track Vision System.

Nose.

The Nose of Pioneer is empty, so it too may house your custom accessories and sensors. The Gripper, Experimenter's I/O Module, and Bump Manipulator accessories, for instance, each replace the Nose.

Motors and Position Encoders

The Pioneer drive systems are powered by reversible-DC motors. The differential drive lets Pioneer literally turn in place. The front drive motors each includes a high-resolution (100 ticks per revolution) optical quadrature shaft encoder for precise position and speed sensing.

Sonars

Pioneer has seven ultrasonic sonar transducers to provide object detection, range information, and environment features recognition. The sonar positions are fixed on the Console—one on each side and five forward facing. Separated by 15-degrees each, the five sonars provide 75 degrees of nearly seamless forward sensing; the respective side sonars are useful for corridor walls and doorways detection.

Sonar firing rate is 25 Hz (40 milliseconds per sonar) and sensitivity ranges from six inches (10cm) to over 16 feet (five meters). (Objects closer than six inches are detected, but their range is given as six inches.) Sonar firing pattern is software controllable. An eighth sonar port is available for user accessories.

Power

A single 12 VDC, seven A-hr sealed lead/acid cell (the AT also supports a 12 A-hr battery) supplies ample power for Pioneer's drives, electronics, and accessories. Typical intermittent operation of the motors gives six or more hours of Pioneer 1 use, or two to three hours of AT autonomous use per charge. If you don't activate the motors, Pioneer's electronics will run for several days on a single charge.

The robot's electronics also may run continuously while recharging when externally powered by the power adapter that comes with the Pioneer package. Typical recharge time varies on the battery's discharge state; it is roughly equal to three hours per volt.

The Pioneer AT's battery fits into a special cradle for easy replacement. Simply open the Deck, disconnect the wire harness on top of the battery, lift out the spent battery, drop in its replacement, reconnect the wire harness, close the Deck lid and tighten its clasps—all tool-lessly. The robot need not be powered down to replace the AT's battery: To "hot-swap" the battery, simply attach the charger during the operation.

Electronics

Except for the drive motor position encoders and custom or optional accessories, all of Pioneer's electronics reside on a single board mounted under the top plate of the Console. The control switches and indicators fit through the top plate of the Console.

The microprocessor is a 16 MHz Motorola MC68HC11F1 running a four megahertz bus. Onboard ports include:

- two high-power, reversible motor drivers
- two position-encoder inputs
- eight multiplexed sonar outputs
- 8 digital input ports (+5 VDC; logic-only)
- 8 digital output ports (+5 VDC; logic-only)
- one 8-bit analog-to-digital input port (0-5 VDC at 10 Hz)
- one digital timer output (1 microsecond resolution)
- one digital timer input (1 microsecond resolution; 65 microseconds duration)
- one RS232 serial port
- one switched 12 VDC auxiliary power port

All the ports, except those for the motors, encoders, and seven of the sonars, are available to the user for Pioneer accessory hardware. Port connector pinouts and electronic details are in the Appendix A. The Experimenter's I/O Module (also part of the Gripper and Bump Manipulator accessories) provides additional I/O support for the Pioneer.

User Controls, Ports, and Indicators

Liquid Crystal Display

Information about the robot appears on a 16-character by two-line liquid crystal display (LCD) on the left side of the Pioneer Console.

In Self-Test mode, the LCD shows the status and readings from the various tests and acquired sensor readings. In server mode, the display shows the state of communication with the host computer. In both the self-test and server modes, the LCD continuously displays the battery voltage and a blinking "heartbeat" asterisk ("*") as the last six characters in the second line of text.

There is a STATUS LED (green) that will blink during normal operation of the robot.

CONTRAST

A small, contrast-adjustment potentiometer for the LCD is inset next to the display. Make sure the `MAIN POWER` switch is ON and the battery is fully charged, then turn the adjustment screw with a small, flat-bladed screwdriver in the indicated direction (`DARK` or `LIGHT`) until the characters are easily visible on the display.

RESET and FUNCTION

The RESET and FUNCTION push-button switches affect the microcontroller's logic systems. Just as if you cycle the MAIN POWER, reset puts the microcontroller into its start-up state and disables the drive motors.

Unlike power off, reset preserves the RAM contents, so any user programs downloaded in standalone mode will be restarted.

Reset disrupts serial communications, effectively breaking any link to an offboard computer.

The FUNCTION button is under software control. For instance, when in self-test mode, pressing the FUNCTION button skips to the next self-test routine. By pressing and holding the FUNCTION button during reset, the microcontroller board can be placed in a special download mode for reprogramming the onboard flash-PROM (see “Updating and Reconfiguring PSOS” Chapter 7) or SRAM (see “Standalone Pioneer” Chapter 8).

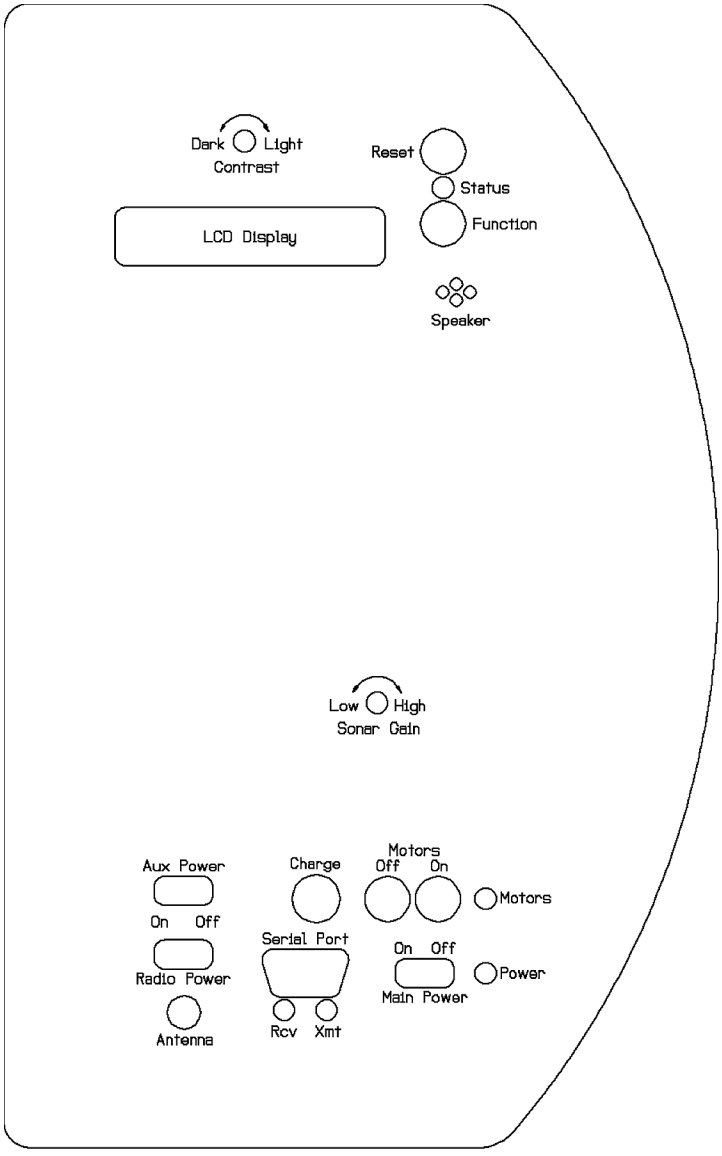


Figure 3-3. Pioneer’s Controls and Indicators

SERIAL PORT

Pioneer's serial port is a standard DB9 receptacle for RS232-compatible serial data communication between the Pioneer microcontroller and an external computer (see Appendix A for pinouts and cabling connections). Two adjacent amber LEDs are lit during actual data activity transmitted from (XMT) or received by (RCV) the Pioneer.

> Remove connectors from the Serial Port when using the optional radio modems.

CHARGE

The two-contact CHARGE socket is where you plug in the cable end of the battery charger that accompanies the robot. The charger itself must be plugged into a standard 110 VAC wall socket or into a voltage adapter.

The Pioneer's battery should be maintained in a charged state slightly above 13 VDC, as indicated on the robot's LCD (see above). We recommend you recharge the battery when it falls below 11 VDC, even though the robot may continue to operate below 10 VDC—the lead/acid battery can be damaged if severely discharged.

You may continue to operate the robot while charging, although it will lengthen the recharge time. And use the charger when "hot-swapping" the Pioneer AT's battery. The battery charger may be kept plugged into the robot for up to 24 hours without damage.

> Disengage the motors when recharging the robot.

RADIO POWER

The RADIO POWER slide switch enables (ON) and disables (OFF) power to the radio modem (optional equipment) mounted inside the Pioneer Console. Power to the modem also is controlled by the MAIN POWER switch.

The RADIO POWER switch does not affect the Serial Port functions directly, but you must switch the radio modem's power off if you use the Serial Port to connect a piggy-back laptop or other external computer to the robot.

AUX POWER

The AUX POWER slide switch controls the accessory power line on the microcontroller board. Accessories attached to the Pioneer Power Backpanel power sockets or the vision systems typically get power through the AUX POWER switch, for instance. In particular, the Pioneer AT's special motor amplifier hidden inside the robot is powered through the AUX POWER switch. Switch ON the AUX POWER as well as enable the white MOTORS ON button to drive the Pioneer AT.

SONAR GAIN

The sonar sensors are preset at the factory. You may adjust their general sensitivity and range with the SONAR GAIN control. Using a flat-bladed screwdriver, turn the gain control counterclockwise towards LOW to make the sonars less sensitive to external noise and false echoes. Low sonar gain settings reduce their ability to see small objects and under some circumstances, that's good. For instance, attenuate the sonar gain if you are operating in a noisy environment or on an uneven or highly reflective flooring—a heavy shag carpet, for example. If the sonars are too sensitive, they will "see" the carpet immediately ahead of the robot as an obstacle.

Increase the sensitivity of the sonars by turning the adjustment screw clockwise in the HIGH direction, making them more likely to see small objects or objects at a greater distance. For instance, increase the sonar gain if you are operating in a relatively quiet and open environment with a smooth floor surface.

> The Pioneer AT's AUX POWER switch, as well as the MOTORS switch, control power to its drive system. Both must be ON to move the robot.

Safety Watchdogs and Configuration

Pioneer's onboard server software contains a communications watchdog that will halt motion if communications between a client and the server are disrupted for a certain time interval, nominally two seconds (`watchdog`). The robot will automatically resume activity, including motion, as soon as communications are restored.

Also, Pioneer's server software contains a monitor that limits the drive current, to 1.5 amperes or less by default. If the drive current exceeds the limit (`stallmax`), motor power is cut off for about five seconds (`stallwait`), at the end of which time motor power automatically switches back `ON` and motion continues under server control.

Both these "failsafe" mechanisms help ensure that the robot will not damage objects or be electrically damaged during operation. You may reconfigure the communications, drive current, and stall-wait values to suit your Pioneer's application: See "Updating and Reconfiguring PSOS" Chapter 7.

Quick Start

The Pioneer comes fully assembled and ready for action. This chapter describes how to operate the intelligent mobile robot with the Saphira demonstration software. For more details about programming your Pioneer with Saphira, PAI, or P-LOGO, see their respective programming manuals.

In Saphira demonstration mode, the Pioneer requires a serial communication link to a user-supplied client computer. The serial link may be either a cable tether from the robot's 9-pin serial connector to a basestation or piggyback laptop computer's serial port, or to an optional radio modem pair—one inside Pioneer and its companion connected to the serial port of a basestation computer (see Figure 2-3).

Preparative Assembly

The Pioneer comes fully *plug-and-play* pre-assembled with its battery fully charged for out-of-the-box operation. The only assembly you may need to perform is to attach the optional radio modem's antenna which was intentionally left unattached for shipping. Simply insert the antenna through the hole in the Pioneer's Console plate and screw it in tightly. (See Figure 3-3 for antenna location.)

Radio Modem Antenna Installation (optional)

The host radio modem also needs its antenna screwed tightly into place, its external power supply connected, and the serial cable attached to both the modem and one of the Saphira client host computer's serial ports. All other settings have been configured at the factory and should not be changed (settings listed in Appendix B). See the radio modem manual for additional details.

Serial Connections

If you don't have radio modems or don't intend to use them for the robot-computer communication link, insert the 9-pin male connector on one end of the serial cable into the robot's serial port and the other end into one of the serial ports in the client computer. The cable may be a long tether to a basestation or a short link to a laptop computer riding piggyback on the Pioneer.

Saphira Client Installation

The Saphira client-development software, including the Saphira demonstration program and Pioneer simulator, as well as a variety of other Pioneer-related software, including PAI and P-LOGO, each come distributed as a compressed archive of directories and files. We include a 3.5-inch, 1.4 MB floppy diskette containing these software configured and compiled for Windows95/NT systems. Pioneer owners should obtain other Saphira and Pioneer-related software versions and updates for other platforms from our website (see Chapter 2 for details.):

`http://robots.activmedia.com`

The decompressed Saphira software typically requires four megabytes of hard-disk drive storage space. Once decompressed, the various Saphira libraries and executables may be found inside the Saphira directory.

Follow the instructions in the `README` file that accompanies the software for your platform to install the Saphira software. For instance, the Windows95/NT archive is self-extracting program; simply double-click the "`.EXE`" icon and follow the extraction program's instructions.

Saphira Client Startup

To start the Saphira client demonstration program, first locate the executable program: It's inside the `bin` directory that is in the top-level Saphira directory—typically `C:\saphira\verxx\bin` or `saphira/verxx/bin`. The demonstration program is named `saphira` or `saphira.exe`. For instance,

with the mouse, double-click the `saphira.exe` icon inside the C:saphira\ver61\bin folder on your Windows95 or NT desktop, or navigate (`cd`) to inside the Saphira directory on your UNIX or Linux machine and type `saphira, or ./saphira` to execute the Saphira client software there.

If properly installed and executed, a Saphira main window should open and appear on your graphics screen. Otherwise, diagnostic error dialogs will inform you of the problem. See the *Saphira Software Manual* for more details.

Pioneer Cold Start Up

Place your Pioneer on the floor in an open space. Slide the `MAIN POWER` switch to `ON`. The red power indicator lamp should light and the green STATUS lamp next to the LCD should begin blinking. You should also hear a low buzz, then "beep, beep" from the onboard speaker. During this initialization phase, the LCD on the control panel also should display a brief banner, then settle into the message:

```
SYNC G PSOS #.##
 C0 A      vv.vV*
```

(asterisk flashes on and off) telling us the robot is ready to open a connection with a client. The "#" characters actually display the Pioneer Server Operating System (PSOS; the aforementioned "server PROM") version number; "4.5", for example. And the voltage ("vv.vV") reading will be the actual battery charge, in volts.

The same initialization sequence occurs whenever you press the red `RESET` button. The drive motors automatically disengage when the Pioneer initially powers up or on `reset`.

Radio Modem Start Up

If you own and wish to use the optional radio modems for Pioneer client-server communications, make sure the power and serial cables are all properly connected to the external radio modem and to an available serial port on the client computer.

Slide the three-position switch on the external radio modem to `Send` (Figure 4-1). The radio modem's red `Pwr/Batt` lamp should light; the dispositions of the other indicator lamps depend on the status of the communication link with the robot, as explained below.

On the Pioneer, remove any connectors from the serial port on the Console and slide the `RADIO POWER` switch to `ON`. If the radio modem is installed and operating, both of the amber Serial Port indicator lamps (`RCV` and `XMT`) should light.

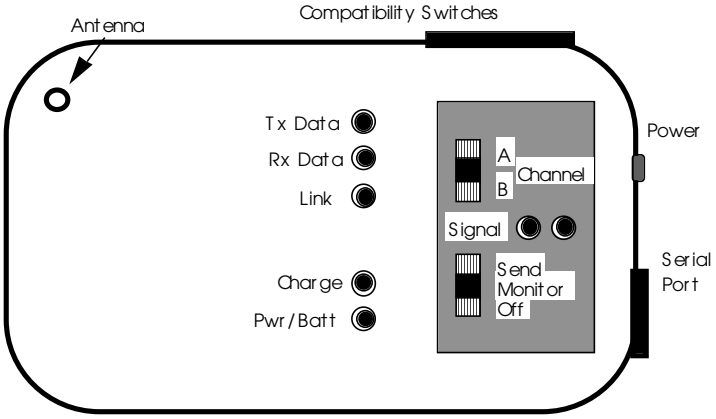


Figure 4-1. External radio modem controls and connections

The green `Link` and red `Signal` lamps on the external modem should now light. If not, try switching to the alternate channel, `A` or `B`, on the external modem. Or bring the robot closer to the basestation computer and companion radio modem. The `Signal` lamps will flicker and fade as the robot begins to exceed normal operating distances of nearly 100 feet in an electrically cluttered space or up to 500 feet in the open.

If difficulties persist, check the external modem DIP switch configuration settings shown in the Appendix B.

Starting Saphira Client/Pioneer Server Communications

When first started up, `reset`, or after completing the self-test, the Pioneer enters a quiet state awaiting a communication link with a client computer. To establish a connection with the Saphira client, pull down the `Connect` menu of the Saphira `main` window and engage the appropriate serial port: `/dev/ttyx` on a Sun workstation, the `modem` port on a Macintosh, one of the `COM` ports (1-4) on a PC, or any of the alternative serial ports that hosts the robot-to-computer connection (see Figure 4-2).

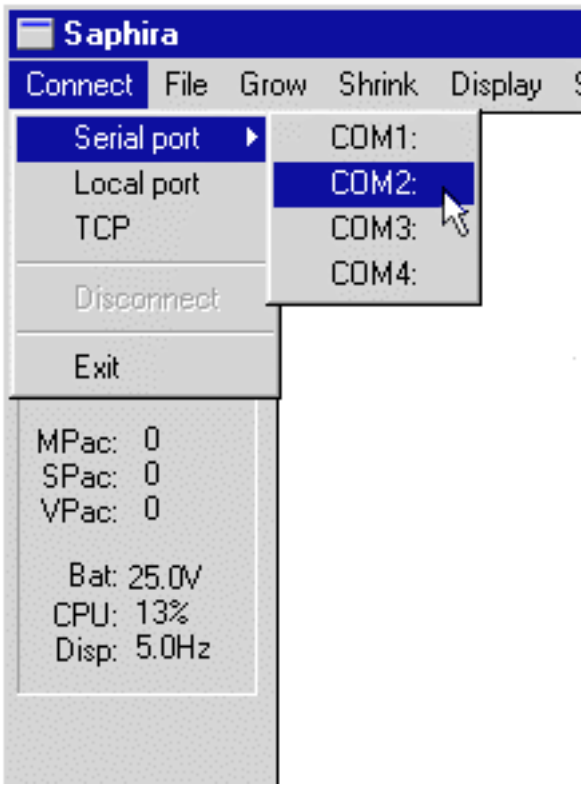


Figure 4.2 Connecting Windows95 Saphira with Pioneer

The Saphira client establishes communication by sending three synchronization packets to the robot server. The Pioneer server responds by echoing each synchronization packet. You may follow this sequence on the Pioneer's Console LCD and in the Saphira client's display window. The synchronization packets also appear sequentially next to the word `SYNC` on the top line of the LCD display on the robot, as they are received and echoed by the communications server. If these numbers do not appear, the communication line is down or the client is malfunctioning, and after a short time Pioneer will return to its waiting state. The `SYNC` number "3" is a special error code meaning there is a noisy communication line which must be ameliorated before operating Saphira.

A Successful Connection

Once synchronization is successfully negotiated, the Saphira client requests the Pioneer server to open and initiate its activities, including sonar polling, position integration, and so on. You should hear the Pioneer's sonars fire with a distinctive and repetitive clicking sound, and the drive motors should engage (nearly inaudible whine), although the robot should not move. If not, check the `MOTORS` LED on the Console for the status of the drive. Press the white `MOTORS ON` pushbutton to enable the drives. Also, with the Pioneer AT, put the backpanel motors switch to ON.

The external radio modem's `TX` Data and `RX` Data indicator LEDs should blink rhythmically. Similarly, the amber `Serial Port` indicator LEDs on the robot's Console should blink indicating Saphira communication.

The Pioneer LCD also should display something like the following message, indicating the Saphira connection is active and operating. (Except for the status word and battery voltage, the other values are for factory testing purposes. Please ignore.)

```
Con 00000  00000
*C0 A     vv.vV*
```

The Pioneer server may be opened by only one client at a time. Trying to communicate with Pioneer with two or more clients will hopelessly confuse it. However, while communication is restricted to a single client, the Pioneer may be controlled by several, cooperating clients on a network. And up to ten separate robots may operate in the same area with separately configured radio modems.

Operating the Saphira Client

Once communications between the Saphira client and the Pioneer server are established, the robot automatically becomes responsive and intelligent. For example, although you may drive it manually towards an obstacle, Pioneer will not crash because it detects and actively avoids collisions. Collision avoidance is just one of the many mobile robot behaviors demonstrated by the Saphira application and available in the Saphira C libraries. Again, this section is meant to get you started with Pioneer 1. Please read the *Saphira Software Manual* for details.

The main window of the Saphira client is a graphical representation of what the Saphira-enabled Pioneer "sees" through its sonars and deciphers about its physical space. For instance in Figure 4-3, Pioneer (center octagon) has identified a corridor and several doors. Notice the small dots, which are recent sonar reflections. The long lines through the sonar reflections are the calculated corridor's geometry. The rectangles dead ahead of the robot represent an obstacle "detected and of interest" to Pioneer. One of Saphira's behaviors, by the way, is to have Pioneer seek and traverse the center of a corridor.

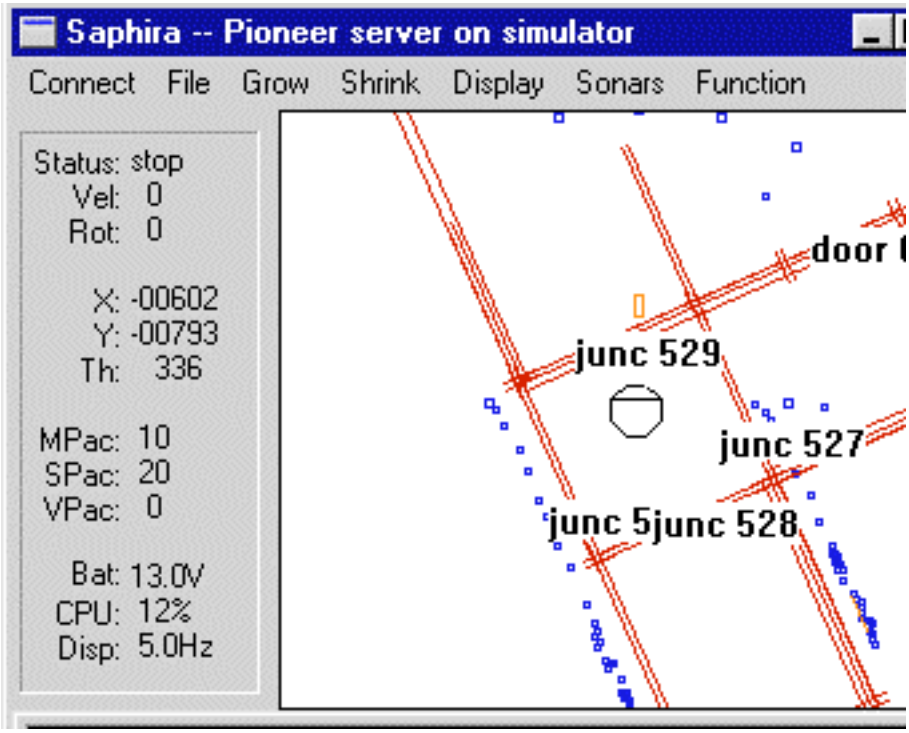


Figure 4-3. Pioneer Saphira client main window display

You may enable and disable Saphira behaviors for Pioneer by selecting or deselecting them from menu items in the Saphira client and from the client keyboard. These include manual drive operation and disabling/enabling obstacle avoidance and constant velocity behaviors (Table 4-1).

Each keypress moves the robot forward or backward faster or slower and incrementally changes its direction. For instance, when turning, it is often useful to push the left- or right-turn key rapidly several times in a row, because the turn increment is small.

Engage the Pioneer's motors (white MOTORS button and AT's backpanel switch) or the robot just won't move, no matter how excited you get.

Other behaviors and controls are found in the Saphira client's various pulldown menus, as described in detail in the *Saphira Software Manual*.

Table 4-1. Saphira client keyboard controlled behaviors

KEY	ACTION
i, ↑	Increment forward velocity
m, ↓	Decrement forward velocity
j, ←	Incremental left turn
l, →	Incremental right turn
k, *space*	All stop
g	Constant Velocity on/OFF
c	Obstacle Avoidance ON/off

Disconnecting Serial Communications

(intentionally or unintentionally)

When finished with the client/server connection, simply draw down with the mouse to the `Disconnect` item in the `Connect` menu of the Saphira client application (Figure 4-4).

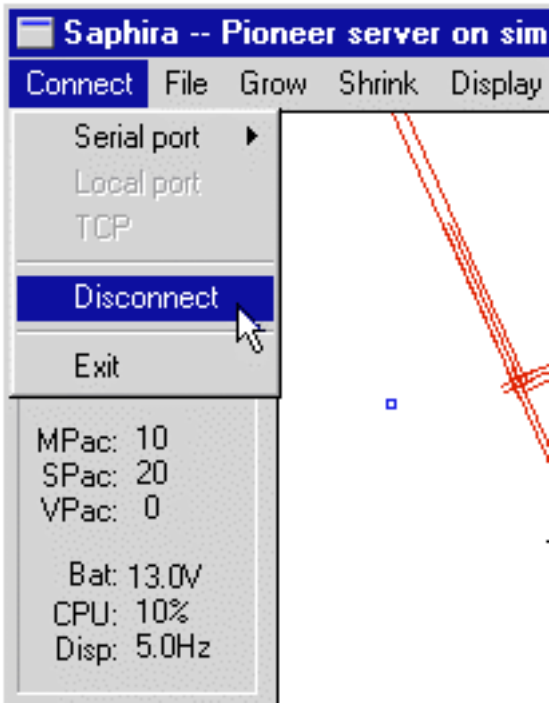


Figure 4-4. Gracefully disconnect the Saphira client from the Pioneer server.

Or shut down the application altogether. The Pioneer automatically will disengage its drive motors and stop moving, and its sonars should stop firing. Its `LCD` also should return to the waiting state message. You may now slide the Pioneer's `MAIN POWER` switch to `OFF`.

Problems?

If for some reason Saphira client-Pioneer server communications are disrupted, but both the client and server remain active, you may revive the connection with little effort: If you are using radio modems, first

check and see if the robot is out of range. Simply pick up the robot and move it closer to the external radio modem until the Signal lamps on the external modem brighten indicating the robot is back within reasonable range of operation. The connection should resume. Or, if either or both of the radio modems were inadvertently switched off, simply switch them back ON (Send) to resume client/server communications.

If either or both the client or server are somehow disabled during a session, for instance if you inadvertently switch off the robot's MAIN POWER or press the RESET button, you need to restart the connection. Switching the MAIN POWER OFF and then back on or pressing the RESET button puts the robot server back to its pre-communications wait state, ready to accept client connections again. If the Saphira client application is still active, simply draw down on the Connect menu to the Disconnect item. Otherwise, restart the application and reconnect the Saphira client with the Pioneer server.

Self Tests

The Pioneer server PROM comes with a series of short test routines for the drive motors, sonars, I/O, and onboard processor. To run it, start up the robot and enable the drive motors by pressing the white MOTORS ON button. (Switch the AT's AUX POWER on, as well.)

> Place Pioneer on the floor and have everyone step back *before* engaging the Self-Test Mode.

The green Motors ON lamp should light. (The red MOTORS OFF button immediately disengages the drive motors for safety. The red RESET button also disengages the motors when pressed.)

Press the black FUNCTION button once. The LCD message should now display:

```
BREAK, Boot to r
eturn     vv.vV*
```

indicating that Pioneer has entered self-test mode. You may press the RESET button at any time to disable the self-test, and you may press the FUNCTION button to skip past any one of the self-tests.

Motors Test

The first self-test exercises the Pioneer's drive motors, during which time the robot is not at all conscious of bystanders. Please have everyone step back and remove any obstacles from within a circle of four to five feet diameter around the Pioneer. When ready, press the black FUNCTION button once again.

The motors self-test begins by engaging the right drive wheel, first forward, then in reverse, each to complete a partial turn counterclockwise, then clockwise. Similarly, the left wheel engages, first forward, then reverse to complete partial turns, first clockwise, then counterclockwise. The LCD display, although difficult to read while the robot is in motion, shows the applied power and encoded position for each motor, left (L) and right (R), according to the scheme:

```
R ppp  L ppp
+ddd -ddd vvv.V*
```

where "ppp" is the encoded position value and "ddd" is the applied drive power forward (+) or reverse (-). The last value is, of course, the battery voltage.

Use the drive test to ensure that the drive and encoder cables are in their correct sockets on the microcontroller board. They're the same size and have the same number of pins, so they can be mistakenly switched right for left and left for right.

Sonar Test

Pioneer's self-test series automatically continues on to the next test by individually firing and reading echoes from its seven ultrasonic sonar transducers. You should hear the distinctive clicking sound as the sonars fire for about five seconds each in order from left to right beginning with the one on the left side nearest the LCD. The display tells you the sonar number under test and the number (hexadecimal notation) of microseconds, up to a maximum of 7000, it takes to receive an echo from some nearby object, such as your hand.

For example, this might be the LCD message if you held your hand a few inches away from the center sonar during its test period:

```
Range:  03 0A57
          12.4V*
```

The sonars are numbered 0 through 6. The self-test also fires a phantom eighth sonar (sonar #7); the extra one may be added to a custom Pioneer.

Digin Test

A third self-test lets you examine the values of the eight digital input ports (ID0-7) that come with the Pioneer. The state of each input port is mapped into a single byte, which value gets displayed in the LCD as a 2-digit hexadecimal value, "ii":

```
Digin:  ii  BOOT
 to skip  vv.vV*
```

The basic Pioneer comes with no attachments to the I/O ports. Their normal state is digital On, hence the LCD reading should be "FF". The Digin self-test becomes useful, of course, when you have equipment attached to one or more of the digital input ports, such as a Gripper and the Experimenter's Module. See the Appendix A, "I/O Ports and Connections", for the location of the digital input ports (ID0-7) on the Pioneer microcontroller.

Digout Test

Press the black FUNCTION button to exit the Digin self-test and activate the Digout self-test. Like Digin, the self-test software maps the current state of each the eight digital output ports (OD0-7) that come with the Pioneer robot into a single byte and displays the result in the LCD as a 2-digit hexadecimal value "oo":

```
Digout: oo  BOOT
 to skip  vv.vV*
```

The Digout self-test currently cycles the output bits automatically, alternatively setting and resetting alternate ports, so that the display cycles between oo="AA" and oo="55".

See the Appendix A, "I/O Ports and Connections" for the location of the digital output ports (ID0-7) on the Pioneer microcontroller.

PSU Test

Press the black FUNCTION button to exit the Digin self-test and activate the PSU self-test. The PSU self-test exercises the ODT output port, alternating the duty cycle so that an attached servo will toggle between two positions. The LCD displays x=7 and x=5 for the two positions, receptively:

```
PSU:    0x00BOOT
 to skip  vv.vV*
```

Analog Test

Press the black FUNCTION button on the Console once again to enable the Analog test. The value of the current reading at the analog-to-digital input port on the Pioneer microcontroller gets displayed as a two-digit hexadecimal value "aa":

```
Analog: aa  BOOT
 to skip  vv.vV*
```

The byte value corresponds linearly in 256 increments to an applied 0-5 VDC analog input ("FF" = 5 VDC, for example).

See the Appendix A, "I/O Ports and Connections" for the location of the analog-to-digital port (A/D) on the Pioneer microcontroller.

Processor Test

Press the black FUNCTION button once again to enable the final Pioneer self-test, which exercises the microcontroller processor. The LCD shows the percentage of idle CPU time, relative to a 68HC11 running at eight megahertz. The Pioneer's CPU runs twice as fast—at 16 megahertz—so its relative CPU idle time will typically exceed 100 percent.

Pioneer Server Operating System

In Pioneer's client/server model, the robot server works to manage all the low-level details of the robot's systems, including operating the drives, firing the sonars and collecting echoes, and so on, on command from and reporting to a separate client application, such as Saphira. With Pioneer, the server software is the Pioneer Server Operating System (PSOS; see Figure 6-1).

High-level robotics applications developers do not need to know many details about a particular robot server, because the client typically insulates them from this lowest level of control. Some of you, however, may want to write your own robotics control and reactive planning programs, or just would like to have a closer programming relationship with your robot. This chapter explains how to communicate with your robot via the PSOS client/server interface. The functions and commands, of course, are supported in the Saphira and PAI C libraries that came with your Pioneer.

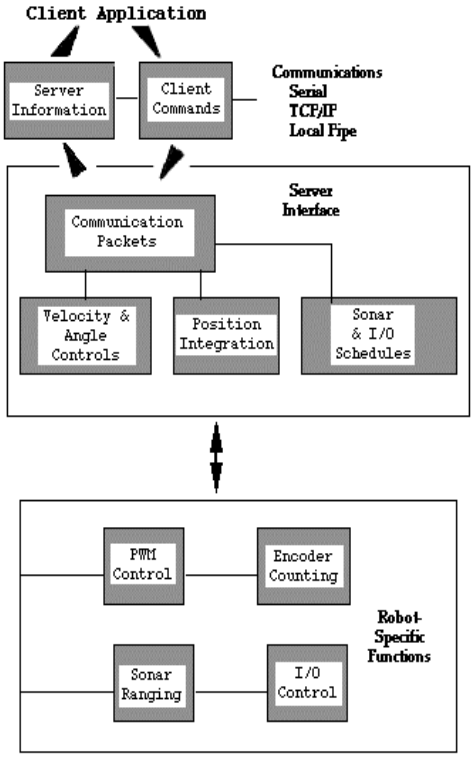


Figure 6-1 Saphira client-robot server architecture

Communication Packet Protocol

PSOS communicates with a client application using a special packet protocol. It is a bit stream consisting of four main elements (Table 6-1): a two-byte header, a one-byte count of the number of data and checksum bytes in the packet, a client command including arguments or a server information data block, and ending with a two-byte checksum.

Table 6-1 Main elements of PSOS communication packet protocol

Component	Bytes	Value	Description
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of subsequent data bytes plus checksum; must be < 200 total bytes long
Data	N	command or SIB	Client command or server information block (SIB; discussed in subsequent sections)
Checksum	2	computed	Packet integrity checksum

Packet Data Types

Packetized client commands and server information blocks use several data types, as defined in Table 6-2. There is no convention for sign; each packet type is interpreted idiosyncratically by the receiver. Negative integers are sign-extended.

Table 6-2 PSOS Communication Packet Data Types

Data Type	Byte Count	Byte Order
Integer	2	b_0 low byte; b_1 high byte
Word	4	b_0 low byte; b_3 high byte
String	up to ~200, length-prefixed	b_0 length of string; b_1 first byte of string
String	unlimited null-terminated	b_0 first byte of string; 0 (null) last byte

Packet Checksum

Calculate the communication packet checksum by successively adding data byte pairs (high byte first) to the running checksum (initially zero), disregarding sign and overflow. If there is an odd number of data bytes, the last byte is XOR-ed in to the low-order byte of the checksum.

NOTE: The checksum word is placed at the end of the packet with its bytes in the reverse order of that used for arguments and data; that is, b_0 is the high byte, and b_1 is the low byte.

Use the following C-code fragment in your client applications to compute a checksum:

```
int
calc_chksum(unsigned char *ptr) /* ptr is array of bytes, first is data count
*/
{
  int n;
  int c = 0;
  n = *(ptr++);
  n -= 2;                       /* don't use chksum word */
  while (n > 1) {
    c += (*(ptr)<<8) | *(ptr+1);
    c = c & 0xffff;
    n -= 2;
    ptr += 2;
  }
  if (n > 0) c = c ^ (int)*(ptr++);
  return(c);
}
```

Packet Errors

Currently, PSOS ignores a client command packet whose byte count exceeds 200 or has an erroneous checksum. The client should similarly ignore erroneous server information packets.

PSOS does not acknowledge receipt of a command packet nor does it have any facility to handle client acknowledgment of a server information packet. Hence, Pioneer client/server communication is as reliable as the physical communication link. A cable tether between the robot and client computer, such as a piggyback laptop, is very reliable links; radio modem-mediated communication is much less reliable.

Accordingly, when designing client applications that may use radio modems, do not expect to receive every information packet intact, nor have every command accepted by the server.

The design decision to provide an unacknowledged packet interface is a consequence of the realtime nature of the client/server interaction. Simply retransmitting server information blocks or command packets would result in antiquated data not at all useful for a reactive client or server.

For some operations, however, the data do not decay as rapidly: some commands are not overly time-sensitive, such as those that perform housekeeping functions like changing the sonar polling sequence. It would be useful to have a reliable packet protocol for these operations, and we are considering this for a future release of Saphira server interface.

In the meantime, the Saphira client/server interface provides a simple means for dealing with ignored command packets: Most of the client commands alter state variables in the server. By examining those values in the server information packet, client software may detect ignored commands and reissue them until achieving the correct state.

Client Commands

PSOS implements a structured command format for receiving and responding to directions from a client for control and operation of the robot or its simulator. You may send PSOS commands to the robot at a maximum rate of once every 100 milliseconds. The client must send a command at least once every two seconds or so (see "Updating and Reconfiguring PSOS" Chapter 7); otherwise, the server will stop the robot's onboard drives.

Table 6-3. PSOS client command communication packet

Component	Bytes	Value	Description
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of command bytes plus checksum; must be < 200 total bytes long
Command Number	1	0 - 255	Client command number; see Table 4-4
Arg Type (optional)	1	0x3B or 0x1B or 0x2B	Data type of command argument, if included: (*sfARGINT*) positive integer (*sfARGNINT*) negative int or absolute value (*sfARGSTR*) string, null-terminated
Argument (optional)	N	data	Command argument; integer or null-terminated string
Checksum	2	computed	Packet integrity checksum

The PSOS command is comprised of a one-byte command number optionally followed by, if required by the command, a one-byte description of the argument type and the argument. To work, of course, the client command and its optional argument must be included as the data component of a client communication packet (Table 6-3; also see earlier sections of this chapter).

Client Command Argument Types

There are three different types of PSOS client command arguments: positive integers two bytes long, negative integers two bytes long, and strings of up to 195 characters long (200-byte limit on packets) terminated with a 0 (`NULL`). Byte order is least-significant byte first. Negative integers are transmitted as their absolute value (unlike information packets, which use sign extension for negative integers; see below). The argument is either an integer, a string, or nothing, depending on the command.

Saphira Client Command Support

Saphira fully supports client commands with useful library functions. Prototypes can be found in `handler/include/saphira.h and saphira.pro`. See the *Saphira Software Manual* for details.

Server Information Packets

PSOS automatically sends a packet of information over the communication port back to the client every 100 milliseconds. The PSOS server information packet informs the client about a number of the robot's operating parameters and readings, using the order and data types shown in Table 6-5.

Table 6-4. PSOS 4.6 supported client commands

Command Name	Number	Argument Value(s)	Description
sfSYNC0	0	none	Start connection; PSOS echoes these synchronization commands back to client.
sfSYNC1	1	none	
sfSYNC2	2	none	
sfCOMPULSE	0	none	Communication pulse
sfCOMOPEN	1	none	Open the motor controller
sfCOMCLOSE	2	none	Close PSOS-client connection
sfCOMPOLLING	3	string sequence of numbers 0-7	Set sonar polling sequence
sfCOMSETO	7	none	Set server origin
sfCOMVEL	11	signed int mm/sec	Forward (+) or reverse (-) velocity (all wheels)
sfCOMHEAD	12	unsigned int degrees	Turn to absolute heading 0-360 degrees
sfCOMDHEAD	13	signed int degrees	Turn heading +-255 degrees
sfCOMRVEL	21	signed int degrees/sec	Set rotational velocity +- 255 degrees/sec
sfCOMVEL2	32	2 bytes 4*mm/sec	Set wheel velocities independently +- 4mm/sec
sfCOMDIGOUT	30	integer bits 0-7	Set digital output bits
sfCOMTIMER	31	integer PIN 0-7	Initiate user input timer, triggering an event with specified PIN
sfCOMGRIPPER	33	integer 0, 1, 4, 5	Sets gripper state
sfCOMPTUPOS	41	integer 1-2000 ms	Set pulse-width for position servo control.
sfCOMSTEP	64	none	Single-step mode (simulator only)

Table 6-5. PSOS server information data packet (minimum contents)

Name	Data Type	Description
Header	int	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2; must be less than 201 (0xC9)
Status	byte = 0x3S; where S =	Motors status
	sfSTATUSNOPOWER	Motors power off
	sfSTATUSSTOPPED	Motors stopped
	sfSTATUSMOVING	Robot moving
Xpos	unsigned int (15 ls-bits)	Wheel-encoder integrated coordinates; platform-dependent units—multiply by *DistConvFactor* in the parameter file to convert to mm; roll-over ~ 3 m
Ypos	unsigned int (15 ls-bits)	
Th pos	signed int	Orientation in platform-dependent units—multiply by `AngleConvFactor` for degrees.
L vel	signed int	Wheel velocities (respective Left and Right) in platform-dependent units—multiply by `VelConvFactor` to convert to mm/sec.
R vel	signed int	
Battery	byte	Battery charge in tenths of volts
Bumpers	2 bytes - L and R	Motor stall indicators
Bumpers	unsigned int	
Control	signed int	Setpoint of the server's angular position servo—multiply by `AngleConvFactor` for degrees
PTU	unsigned int	Pulse width of position servo
Say	byte	verbal/sound clues
Sonar readings	byte	Number of new sonar readings included in information packet; readings follow:
Sonar num	byte	Sonar number
Sonar range	unsigned int	Sonar reading—multiply by *RangeConvFactor* for mm
... *rest of the sonar readings ...*		
Input timer	unsigned int	User input timer reading
User Analog	byte	User analog input reading
User Input	byte	User digital input pins
User Output	byte	User digital output pins
Checksum	int	Checksum (see previous section)

In future versions, PSOS server information packets may contain additional, appended data fields. To remain compatible, have your client application accept the entire data packet, even though it may use only a few selected fields.

Start Up and Shut Down

Before exerting any control, a client application must first establish a connection to the Pioneer server via its RS-232 serial link. Over that established communication link, the client then sends commands to and receives back operating information from the server.

Synchronization—sfCOMSYNC

When first started, the Pioneer robot is in a "wait" state; PSOS listens for communication packets over its designated port. To establish a connection, the client application sends a series of three synchronization packets through the host communication port—`sfSYNC0, sfSYNC1, and sfSYNC2,` in succession.

PSOS responds to each client command, forming a succession of identical synchronization packets. The client should listen for the returned packets and only issue the next synchronization packet after it has received the echo.

Autoconfiguration

PSOS (versions 4.1 and later) automatically sends configuration information back to the client in the last sync packet (`sfSYNC2`). After the sync byte, there are three `NULL`-terminated strings that comprise the robot's name, class, and subclass. You may uniquely name your Pioneer and set the subclass variable with a special configuration tool. See the next Chapter for details.

The Pioneer class string is simply "Pioneer". The subclass is configurable and should be set to the Pioneer platform you have: "PionAT" for the Pioneer AT; and "Pion1m" or "Pion1x" for Pioneer 1s with newer or older motors, respectively.

Clients may use these identifying parameters to self-configure their own operating parameters. The Saphira client software, for instance, loads a special Pioneer parameters file found in the Saphira `params` directory.

Opening the Servers—`sfCOMOPEN`

Once the communication link is established, the client should then send the `sfCOMOPEN` command, which causes the Pioneer to perform some housekeeping functions, start its sonar and motor controllers (among other things), listen for client commands, and to begin transmitting server information.

Keeping the Beat—`sfCOMPULSE`

As mentioned earlier, a PSOS safety watchdog expects that the Pioneer server receives at least one communication packet from the client every two seconds. Otherwise, it assumes the client/server connection is broken and shuts down the robot's motors. If your client application will be otherwise distracted for some time, periodically issue the `sfCOMPULSE` client command to let the server know you are indeed alive and well. If the robot shuts down due to lack of communications traffic, it will revive upon receipt of a client command and automatically accelerate to the last-specified speed at the current heading.

Closing the Connection—`sfCOMCLOSE`

To close a connection and reset PSOS to the wait state, simply issue the client `sfCOMCLOSE` command.

Movement Commands

As of PSOS 4.2 and later, the robot server accepts several different types of motion commands: You can set the turn angle or velocity, and the forward/back velocity; or, you can control the side wheel velocities independently. Table 6-6 summarizes the command modes available.

The robot server automatically switches to the required motion control mode when it receives one of these commands. For example, if it is in two-wheel velocity mode, and it is sent an `sfCOMHEAD` command, it abandons two-wheel velocity mode and starts controlling the heading and velocity of the robot.

Table 6-6. Server motion command types

Rotation	Translation
`sfCOMHEAD`	absolute heading
`sfCOMDHEAD`	differential heading from control pt
`sfCOMVEL`	forward/back velocity
`sfCOMRVEL`	rotational velocity
`sfCOMVEL2`	left and right wheel velocities

PSOS will try to make the robot achieve the desired velocity and heading as soon as the commands are received, using its internal (de)acceleration managers. Check your robot's operation manual to find its absolute maximum achievable motion and rotational velocities.

Pioneer in Motion

When PSOS receives a velocity command, it accelerates the Pioneer robot at a constant rate set internally to the speed you provided as the argument for `sfCOMVEL`. Rotational headings are achieved by a trapezoidal velocity function (Figure 6-2). This function is re-computed each time a new heading command is received, making on-the-fly orientation changes possible.

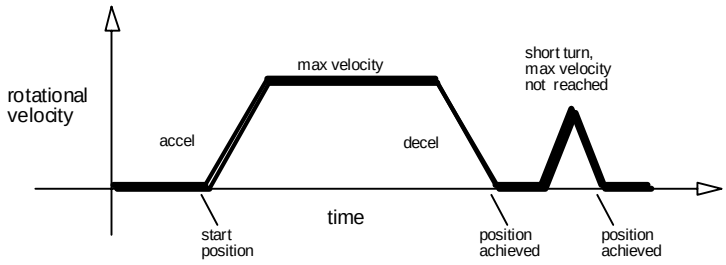


Figure 6-2 Trapezoidal turning velocity profile

Position Integration

Pioneer keeps track of its position and orientation based on dead-reckoning from wheel motion, which is an *internal coordinate position*.

Registration between external and internal coordinates deteriorates rapidly with movement, due to gearbox play, wheel imbalance and slippage, and many other real-world factors. You can rely on the dead-reckoning ability of the robot for just a short range—on the order of several meter and one revolution, depending on the surface (carpets tend to be worse than hard floors).

Also, moving either too fast or too slow tends to exacerbate the absolute position errors. Accordingly, consider the robot's dead-reckoning capability as a means of tying together sensor readings taken over a short period of time, not as a method of keeping the robot on course with respect to a global map.

The orientation commands `sfCOMHEAD` and `sfCOMDHEAD` turn the robot with respect to its internal dead-reckoned angle (Figure 6-3). On startup, the robot is at the origin (0,0), pointing towards the positive x-axis at 0 degrees. Absolute angles vary between 0 and 360 degrees. As the robot moves, it will update this internal position based on dead-reckoning. The x,y position is always positive, and rolls over at about 3,000 millimeters. So, if the is at position (400,2900) and moves +400 millimeters along the y-axis and -600 millimeters along the x-axis, its new position will be (2800, 300).

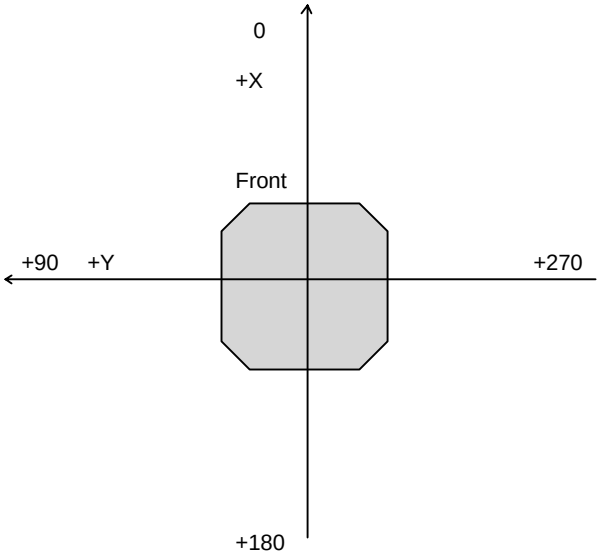


Figure 6-3 PSOS internal coordinate system

Sonars

When opened by the appropriate client command (see `sfCOMOPEN` above), PSOS automatically coordinates and begins firing the robot sonars in a pre-defined default sequence, and sends the results to the client via the server information packet.

Use the `sfCOMPOLLING` command to change the polling sequence of the sonars. Its argument is a null-terminated string of bytes at most 12 bytes long. Each byte is 1 + sonar number.

For example, the string

```
"\001\002\001\006"
```

starts the sonar polling sequence 0, 1, 0, 5. Note that sonar numbers can be repeated. If the string is empty, all sonars are turned off.

I/O

Your Pioneer comes with a number of I/O ports that you may use and which get used by some of Pioneer's accessories. See Appendix A for port locations on the Pioneer microcontroller. All ports are supported in PSOS. The digital value of an applied voltage (0-5VDC) at the analog port, for instance, automatically appears in the Server Information Packet under PSOS control.

Three other client commands give you control over the digital output, timer, and PTU ports.

sfCOMDIGOUT

The revised PSOS 4.2 (and later) sfCOMDIGOUT command number 30 gives you control over the eight digital output ports (OD0-7). The command needs a two-byte argument: The high byte is a mask of those output bits you want to change, and the low byte is the bit pattern for the bits states. This way, you don't need to know the current state of the port(s) you want to change, just the state you want them to be. And you don't change the state of any of the other ports.

For example, to selectively change the ports OD0-3 without altering the states of ports OD4-7, use a mask of 0x0F. Accordingly, a sfCOMDIGOUT argument mask of 0x0F and pattern of 0x55 would set (on; 5VDC) the even output ports (OD0, and 2) and reset (0) the odd ports (OD1, and 3) while leaving ports OD4-7 untouched.

sfCOMTIMER

`See Table 6.2.`

sfCOMPTUPOS

The timer output port (ODT) may be used to control an RC servo. With versions of PSOS predating 4.5, a command packet with sfCOMPTUPOS (command number 41) and an integer argument set the RC servo angle. The effect is particular to a given type of RC servo, but usually a value of 1000 μsec corresponds approximately to a neutral position.

With PSOS 4.5 and later (to better support the multiplexed RC electronics of the Experimenter's I/O accessory), sfCOMPTUPOS is takes a two byte arguments. The first is a servo address (0-4); the second is a pulse width, in 10 μsec increments. Without an Experimenter's Module (or Gripper) attached, PSOS ignores the RC address and will simply affect the Nose Port ODT pin 15.

sfCOMGRIPPER

PSOS 4.2 and later versions fully supports operation of the Gripper accessory for Pioneer through a set of state functions. Please consult the *Gripper & Experimenter's Module Manual* for details.

Updating & Reconfiguring PSOS

Beginning with version 4.3, you not only may update the Pioneer Server Operating System (PSOS) software without physically replacing the ROM hardware on its microcontroller, but you may also reconfigure the identity and operating parameters of your Pioneer Mobile Robot.

The new `psosdl.exe` program lets you download a fresh psos image (psos46.s19, for example) onto the Pioneer microcontroller. Similarly over the serial communication interface with the robot, the new `psoscf` program lets you customize some of Pioneer's identifying and operating parameters, which beginning with PSOS version 4.3, are separately stored in the MC68HC11 microprocessor's memory, including:

- ✓ Name of the Pioneer
- ✓ Pioneer serial number
- ✓ Robot type
- ✓ Motor type
- ✓ PID motor control parameters
- ✓ Number of encoder ticks per wheel
- ✓ Presence of Experimenter's Board/Gripper
- ✓ Serial communication baud rate
- ✓ Maximum integrated current before stalling
- ✓ Motor halt time after stall
- ✓ Communication watchdog timer

Please note at the outset that the `psoscf` program that came with PSOS versions 4.3. 4.4, or 4.5 are *not* compatible or sufficient for use with PSOS 4.6 or later versions. Please make sure you are using the proper version.

Where to Get PSOS Software

If you haven't already collected the latest version of PSOS with the `psosdl` and `psoscf` programs along with this document, select and download the software for your client's computing environment from our webserver:

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

Alternatively, the latest versions of `psosdl(.exe)` and `psoscf(.exe)`, as well as the latest PSOS image, come included with the Saphira software distributions, also available to Pioneer customers from:

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

Be sure to download Saphira 6.1d or later versions to get PSOS 4.6 (or later). Also be aware that the Saphira URL contains license-only versions of the robotics development software, so be prepared to enter the special access username and password that accompanied your Pioneer.

Flash PROM and Rev D MICROCONTROLLER REQUIRED

For you to reprogram your Pioneer to a new PSOS via the `psosdl` software, your Pioneer *must* have a Flash PROM (not UV-erasable PROM) installed on its microcontroller. And, you must have a Version 1.3 (Rev D or later) microcontroller. How do you know if you do have these things or not?

All Pioneer AT owners have the proper equipment, so skip ahead.

If your Pioneer 1 has a serial number greater than P1D-8254, you have the right stuff and can go to the next section. Those who received Pioneers after February, 1997, probably have the correct microcontroller, but may or may not have the flash PROM. And those who received their Pioneer 1 before February, 1997 may have to physically alter their microcontroller if not exchange the board altogether.

Unless you are sure you have the flash PROM onboard, please follow these steps to discover which configuration you may have before contacting us:

Do You Have a Flash PROM?

Step 1: Check the PSOS version number that appears in the Pioneer 1 Console LCD while the robot server is waiting for a client connection. If it says "PSOS 4.3" or later, then your robot came installed with the flash PROM and all you may need to do is reprogram PSOS to the latest version (instructions below). If the LCD says "PSOS 4.2", you probably do not have the flash PROM, but a few did, so go on with the tests. Any earlier PSOS version and you surely don't have a flash PROM onboard. In fact, if you still operate with PSOS 3.x, you need to contact us and upgrade your Pioneer microcontroller altogether.

Step 2: If you are still unsure, go ahead and attempt to download a new PSOS with `psosdl` (details below). If you have a flash PROM onboard and follow the downloading steps carefully, the new PSOS will download correctly and update your Pioneer. Otherwise, the program cannot and therefore will not in any way alter the current PSOS and the `psosdl` program will "error-out" when attempting to download the new PSOS. Note that having a flash PROM is not necessarily sufficient to get the download to work: Some microcontrollers need to be physically modified to accept the flash PROM.

Step 3: You may examine the microcontroller board. Follow the directions in the next Chapter for exchanging the EPROM for SRAM. If the label imprinted on the board just above the LCD display says "Pioneer Version 1.3", you have the right board and you need to exchange the EPROM for a flash PROM. If you have an earlier board version, you do not have a flash PROM onboard (unless you or a colleague put it there) and you will have to make a small modification to the board to get it to work with a flash PROM. See following sections and Appendix D for details.

Obtaining a Flash PROM

You have three options should you need to replace the PSOS EPROM with a flash PROM chip:

Option 1: Do it all yourself. Buy an `ATMEL AT29C256-12PC` (or equivalent) flash PROM, install it, and program the Pioneer with the new PSOS and configuration parameters. This is probably the fastest and easiest way.

Option 2: Have us do it all. Make arrangements to pay for and ship your Pioneer round-trip to us, and we'll install the flash PROM and perform all the upgrades and configurations. This is the slowest and most expensive way, but one that ensures you have a working robot in the end.

Option 3: Take the middle road. Contact us, tell us your Pioneer serial number so we can make sure we get the right parameters, and we'll mail a flash PROM with PSOS to you for free (special/faster shipping extra by special arrangements).

Contact pioneer-support@activmedia.com to obtain a flash PROM for your Pioneer, if you decide not to do it yourself. Specify your Pioneer's serial number and mailing address. Or fax the information to (603) 924-2184. Use the pioneer-support email to report problems and gain assistance, as well.

CAVEAT: To make Option 3 as "bullet-proof/plug-n-play" as we can, the flash PROM we'll send to you for free will *not* contain PSOS version 4.3 or later. Rather, it will contain PSOS 4.2, which will work fine with your Pioneer after simple replacement of the existing PROM. Updating it by software to a new PSOS may require that you physically modify the board, and that you at least once set your Pioneer's operating parameters. Making you do the upgrade to the new PSOS reinforces that necessity that may otherwise be forgotten, in which case your Pioneer will live up to its ancestor's name, Erratic.

Installing the PSOS Software Utilities

The new PSOS software comes as a separate package and as bundled with Saphira version 6.1 and later. If you use the Saphira-included software, follow the procedures for installing Saphira on a host computer, as described in detail in the Saphira documentation. Make sure to set the `SAPHIRA` environment variable to the top-level directory of the Saphira distribution (`/home/saphira/.ver61`, for example), and find the PSOS-related software in the `$(SAPHIRA)/pioneer/psos` directory.

For the individual PSOS software packages found at the Pioneer software website, download the version that matches your basestation computer's environment: `linux-ps46.tgz` for Linux, the various Unix (`sun-ps46.tgz`, for example for SunOS), or MS Windows95/NT (`win32-ps46.EXE`). We distribute these as compressed archives containing all the programs and accessory files you need to perform the PSOS upgrade and to set your Pioneer's configuration parameters.

The `win32-ps46.EXE` distribution is a self-extracting, self-installing package: Simply follow the onscreen directions. For the Linux/Unix versions, uncompress and untar them with the respective system utilities. For example, with the Linux version:

```
% tar –zxvf linux-ps44.tgz
```

The command creates a `psos/` directory in the current path, containing the PSOS software.

Updating PSOS

Use the psosdl(.exe) program to download a fresh copy of PSOS to the Pioneer microcontroller's flash PROM.

Step 1. Serial Connection from Computer to Pioneer

Connect the Pioneer to your host computer through their respective serial ports (see the "Quick Start" Chapter 4). We recommend a direct tether from the computer to the 9-pin serial port on top of the Pioneer Console, since it is the most reliable path for communication. In fact, if you have a Fast-Track Vision System installed, you *must* use a direct tether to that top Console serial port: a tether to the serial port on the backpanel nor the radios will work.

If you have, but do not use the radio modems for serial communications, make sure to switch their power off (RADIO POWER switch); they will otherwise interfere with the direct connection. In all cases, turn he AUX POWER OFF.

Step 2: Put Pioneer into Boot Mode.

Start up or reset your Pioneer. After it has finished initializing, place it into its special Boot mode by pressing and holding the black FUNCTION button, then pressing and releasing the red RESET button. The robot should not actually reset; if it does, you didn't hold the FUNCTION button down properly. Try again.

In Boot mode, the Pioneer's amber serial-port RCV LED should light up; the XMT amber LED should go out; and the green Status LED should stop flashing on. If not, check your connections and settings and try again, making sure you hold the FUNCTION button down before pressing RESET.

STEP 3: Start up psosdl.

With Linux/Unix systems, enter the psos directory and execute psosdl with the following parameters:

```
% psosdl PSOS-IMAGE <–b bootfile> <-c comm-port>
```

The PSOS-IMAGE filename is required and may include a direct or relative path. The file psos46.s19, for example, is the current PSOS version 4.5 image in the psos/ folder, or in the $(SAPHIRA)/pioneer/psos folder.

The bootfile and comm-port parameters are optional: The former is the temporary software Pioneer uses to manage the download (currently bootload.s19). The last, optional parameter, lets you specify the serial communication port that connects psosdl with the Pioneer microcontroller. For Linux/Unix systems, the default is /dev/cua0.

For Win32 systems too, you've got to pass the required PSOS-IMAGE parameter to the psosdl.exe program, so you cannot simply double-click the program icon from the Windows desktop. Rather, you may "drag and drop" the PSOS-IMAGE icon (psos46.s19, for example) onto the psosdl icon with the mouse to successfully launch the program, but only if you also plan to use the default bootfile (currently bootload.s19) and the COM1 serial port.

Otherwise, and in any case, you may execute psosdl.exe and pass it the proper parameters from the MS-DOS Prompt program, normally found in the Programs section of the Start menu.

For example, when using psosdl.exe from the win32-ps46.EXE distribution whose files you've saved in the C:\psos directory, type in the DOS console window:

```
> cd c:\psos
> psosdl psos46.s19 –c com2
```

to perform the download of PSOS version 4.5 through the COM2 serial port on your PC.

Once started, psosdl(.exe) loads PSOS-IMAGE and bootfile, and then waits for you to press the Enter (Return) key to continue or the "q" key to quit. Don't continue unless and until you have completed Step 2.

STEP 4: Download the BOOTFILE image.

Press the Return (Enter) key to continue the psosdl(.exe) program from Step 2. It then downloads the bootfile program to the robot processor's EEPROM. Don't be confused—this should work with all Pioneer microcontrollers, including those without a flash PROM. If successful, the message "boot loader" will appear on the Pioneer's LCD, and the program automatically will go on to download the PSOS image.

If the bootfile download is unsuccessful, no message will appear on the Pioneer LCD, and the psosdl program will tell you the problem and prompt you to either fix the connection or faulty Boot mode problem and continue, or to simply quit the program. If you select to continue, but haven't really solved the

problem, psosdl will fall into an endless loop of unsuccessful attempts to download the software. In this case, use Control-C to stop the program and start over from Step 1.

In some cases, the program will download correctly, but then print a misleading message suggesting that not all the data got through, like "Sent 576 bytes, received 374." Ignore the error, and press the "c" key to continue.

STEP 5: Download the PSOS-IMAGE file.

Once psosdl has downloaded the bootfile program to the Pioneer's microprocessor's EEPROM, it automatically begins to download the PSOS-IMAGE file you had selected in Step 3 (psos46.s19, for example). As it performs each download segment, psosdl places a sequence of periods (".") in the console display. And, too, the serial communication LEDs (on the radio modems especially) should flash as the data gets transmitted from the computer to the robot.

The program will tell you if the download was successful or not. If so, simply reboot the robot and go on to set it's operating parameters (see next section). Otherwise, try the download from Step 1 again, to be sure you performed each step correctly. If this is your second or third time through, you either do not have a flash PROM onboard, or your microcontroller needs to be modified.

See the Appendix D for details on how to modify certain older (pre-February, 1997) Pioneer microcontrollers so that they will accept a flash PROM.

Pioneer Configuration Parameters

Beginning with version 4.3, certain PSOS parameters get stored in Pioneer's microprocessor's EEPROM separate from PSOS, and remain unaffected by changes to PSOS itself. For all Pioneers shipped before November 1, 1997, none of these parameters are set or valid for proper operation with PSOS 4.3 or later. Accordingly, you *must* set your Pioneer parameters *at least once* after updating PSOS.

In addition, please note that if you are upgrading PSOS 4.3 through PSOS 4.5 to PSOS 4.6 or later, you must run the newer psoscf program to set some newer control parameters.

STEPS 1 and 2: (Same as Steps 1 & 2 above).

Perform identical Steps 1 and 2 as described in detail in the previous *Updating PSOS* section to make the serial connection with a host computer and place Pioneer into Boot mode.

STEP 3: Start up the PSOS configuration program

The psoscf (psoscf.exe for Win32) program is the way you view and change your Pioneer's identity and operating parameters. Like psosdl, find it in the $(SAPHIRA)/pioneer/psos directory of your Saphira 6.x distribution, or in the psos/ folder if you have downloaded the update distribution separately:

```
% psoscf <-c comm-port>
```

The program accepts a single, optional parameter—the communication port name (comm-port), which specification is identical to the psosdl program. The port /dev/cua0 (or /dev/ttyS0) in Linux/Unix systems and COM1 are the default serial ports psoscf and psosdl use if none is given from the command line. Accordingly, if you use COM1 on your Win32 PC, you may launch the psoscf.exe icon directly from the Windows desktop, rather than execute the program from the MS-DOS Prompt console.

STEP 4: Setting and Resetting the Configuration Parameters

Once properly started, psoscf retrieves from the Pioneer and displays its current identifying and operating parameters. Tables 7-1 and 7-2 summarize the command names and default values for the Pioneer 1 (Table 7-1) and Pioneer AT (Table 7-2) in columns 1 and 2, respectively. A description of the command and parameter value is in column 3 of the Table 7-1.

For Pioneer 1s built before November, 1997, these parameters initially will not be set to anything near a proper value.

Change a parameter by typing the parameter command name at the psoscf console prompt, followed by a space and the new value. You may edit the command with the delete or erase key and retyping the line. Press Return (Enter) to execute the command. Values may be entered in decimal or hexadecimal form. Decimal is the default; use the 0xNNNN form for hexadecimal numbers.

For example, to change the watchdog timeout to be four seconds, type:

```
> watchdog 200     or
> watchdog 0x0C80
```

Besides displaying all the parameter values with the `show` command, you can view a parameter by typing its command name alone.

Table 7-1. Configuration commands (psoscf), default values and descriptions for Pioneer's identifying and operating parameters (PSOS 4.6).

Command	Default	Description
serial	FFFF	Serial number of this Pioneer board, in hexadecimal. Pioneers built before November, 1997 will not have a preset serial number. If you do change the value, please use the serial number that came with your robot.
subclass	Pion1m	String identifies Pioneer type and gets included on sfSYNC2 return packet for use by client. Set to PionAT for the Pioneer AT; set to Pion1m for a Pioneer 1 with new motors (see "motors" below); set to Pion1x for a Pioneer 1 with the older motors.
revcount	40000	The decimal number of encoder ticks for a 360 degree revolution of the robot. The default value is for Pioneer 1s built after October, 1996 (old PSOS version number contained suffix "m") and 25600 for those built before that date (no PSOS version number suffix). We encourage you to follow the directions in a following "Turn Calibration" section to set this parameter to a number that best reflects the characteristics of your robot in a particular environment.
motor	1	Set this to 1 if your robot was built after October, 1996 (contains new motors; old PSOS version number contained suffix "m"). Otherwise, set this parameter to 0.
experiment	0	Set to 1 if an experimenter's board is present, 0 if not. This parameter is needed only if your Pioneer is equipped with an Experimenter's Board, but not a Gripper.
name	nobody	Unique name you may give your Pioneer. Besides its ownership value, this parameter gets passed to a connecting client as the first argument in the SYNC 2 packet, so is therefore useful to differentiate amongst multiple Pioneers. Maximum of 20 characters; no intervening spaces, please.
baud	9600	Set to 0 for 9,600 baud; 1 for 19,200 baud.
stallmax	16384	Maximum integrated current before the Pioneer 1 motors stall (AT has no stall feedback). Default is the typical value; higher values up to 64000 means a higher current stall threshold useful for Pioneers carrying heavy payloads and climbing steep slopes.
stallwait	4	Pioneer 1 halt time after stall, in 500ms increments.
watchdog	100	Specifies the amount of time, in 20ms increments, a Pioneer will wait for a client command before halting all motor activity. Default equals 2 seconds; don't set below 500ms when using a Saphira client.
pparam	10	Proportional control parameter controls the basic responsiveness of the drive system. Values range from 5 to 20; lower values give a slower, less-responsive system; higher values make the robot "zippier", but can lead to overshoot and oscillation. A heavily loaded robot (payload of 10 lbs) works well with a pparam of 16.
dparam	45	Differential control parameter dampens oscillation and overshoot. Values range from 20 to 60; increasing values better control oscillation and overshoot, but also make the robot's movements more sluggish.
iparam	10	Integral control parameter adjusts residual error in turning and velocity. Values range from 0 to 20; higher values make the robot correct increasingly smaller errors between its desired and actual angular position and speed.
show		Display the current identifying and operating parameters for Pioneer.
defaults 0/1		Resets values to default parameters; 0=Pioneer 1; 1=Pioneer AT.
quit		Stops the PSOS configuration program.

The defaults assume that your Pioneer robot was built after November, 1996. Check your old PSOS version number. If it contained the suffix "m", then you have the newer Pioneer, and the default motor value is correct. Otherwise, you must set the `motor` parameter to 0, and the `revcount` parameter to around 25600.

The critical operating parameters for Pioneer are `motor`, `revcount`, and the PID control parameters. Get them wrong and your robot won't run properly. Assuming the defaults for your AT, for example (configuration should have been preset at the factory, by the way, but you can change them), makes it think it's a Pioneer 1 and do all kinds of weird things.

The subclass parameter is important at least for Saphira. The client software uses the subclass to identify and load the proper parameters file which it uses to control the robot.

Table 7-2. Configuration parameters specific for Pioneer AT (PSOS 4.6)

Parameter	Should be:
serial	
subclass	PionAT
revcount	46000
motor	1
experiment	
name	
baud	
stallmax	Pioneer 1 only
stallwait	Pioneer 1 only
watchdog	
pparam	8
dparam	20
iparam	8
show	
defaults 1	
quit	

Turn Calibration

Individual Pioneers may differ in the number of wheel encoder counts registered per full turn. Moreover, the full-turn encoder count can change significantly depending on the environment where you run the robot. Here is a four-step calibration procedure you can perform to make Pioneer's turn registration more accurate:

STEP 1: Place the robot in the environment where it is to be calibrated, turn it on, and connect to it with a Saphira client.

STEP 2: Disable the motors (red `MOTORS OFF` button). Turn the robot by hand until the second number (top right) displayed in the Pioneer Console LCD reads "00000" or as close to it as possible. This number is the difference between the left and right wheel encoder counts, in hexadecimal notation.

STEP 3: Enable the motors (white MOTORS ON button; AUX POWER on for AT, too). Back at the Saphira main window, press and hold either the left or the right arrow key on the client's keyboard to rotate the robot a full 360 degrees in a continuous motion. If you overshoot, rotate the robot back until you get as close to 360 degrees as you can. DO NOT turn the robot by hand. The motors will resist you doing so, and you just aren't able to rotate the robot as accurately as the robot can do itself. Record that differential encoder count value (second number in the Pioneer LCD).

Note that the LCD difference number is limited to revcount: It will "turn over" at the current revcount. If you had chosen Pioneer AT's defaults with a revcount of 46000, for example, the LCD counter will go through this sequence (or the reverse) as you spin the robot: 45998, 45999, 0, 1, 2, and so on. To derive a proper number, either add the extra counts beyond the current revcount, or simply turn the robot 180 degrees and double the resulting count.

STEP 4: Start up the PSOS configuration utility (psoscf) and use it to set the revcount parameter to the decimal value you determined in Step 3. (See the previous section, "Setting Pioneer's Configuration", for operation of the psoscf utility.)

Standalone Pioneer

As described in the previous chapters, the Pioneer microcontroller's flash PROM is reprogrammable. Or you may replace the PROM with 32K of static RAM and download your own programs to the Pioneer microcontroller. This way, the robot performs all computations onboard for fully autonomous operation. Communication via the serial port with a host computer is still possible, and necessary for downloading programs. It can also be useful during runtime for coordinating a team of robots.

Standalone mode also lets you perform experiments in low-level sensing and control, such as feedback control of the motors. Although the MC68HC11 microcontroller is a slow computational device and memory is limited, it is possible to program some interesting behaviors, such as using ideas from subsumption architectures.

With this manual, we tell you how to convert Pioneer for standalone operation. We do not tell you details about the Pioneer systems or offer any help with your programming efforts. We are developing Pioneer standalone mode support kits and materials for running Interactive C native, a multitasking OS for the MC68HC11 originally developed at MIT. They are scheduled for release sometime in late 1996. So, stay tuned to the Pioneer-users newsgroups for announcements.

In the meantime, we suggest you collect and read the various texts and treatises on the programming the MC68HC11 microprocessor. A good starting point is Russ Hersch's excellent frequently-asked questions (FAQ) which he posts periodically on Usenet's `comp.robotics` newsgroup, which we also make available at the `http://robots.activmedia.com/docs` site.

Converting Pioneer to Standalone Mode

To convert the Pioneer into standalone mode, you must disassemble the Console and replace the server EPROM on the microcontroller card with a 32K static random-access memory (SRAM) chip. Use a low-power CMOS 62C256 SRAM package, rated at 120 nanosecond access time or better.

Console Disassembly

Make sure the MAIN POWER switch on the robot is OFF. If you have an onboard serial modem, unscrew and remove the antenna.

With one of the hex wrenches that came with your robot, remove the six outermost screws that secure the top plate to the Console—two on the front and two on each side. Gently lift the Console top plate to expose the microcontroller board, which is mounted to its underside, and the various wire harnesses attached to it. Being careful to not handle the attached microcontroller card, remove the wire connectors on both sides of the card from their sockets, freeing the Console top plate with the microcontroller card attached.

Lay the Console top plate face down, Pioneer microcontroller card up. Locate and remove the six hex screws which secure the microcontroller board to the Console top plate. Separate the parts (handle the microcontroller by its edges), and lay the microcontroller board face-up on a table.

EPROM/SRAM Replacement

Locate the EPROM/SRAM socket (Figure 8-1) and note that the Pin 1 locator notch on its case is towards the outer edge of the board—you'll want to similarly position the SRAM chip in a later step.

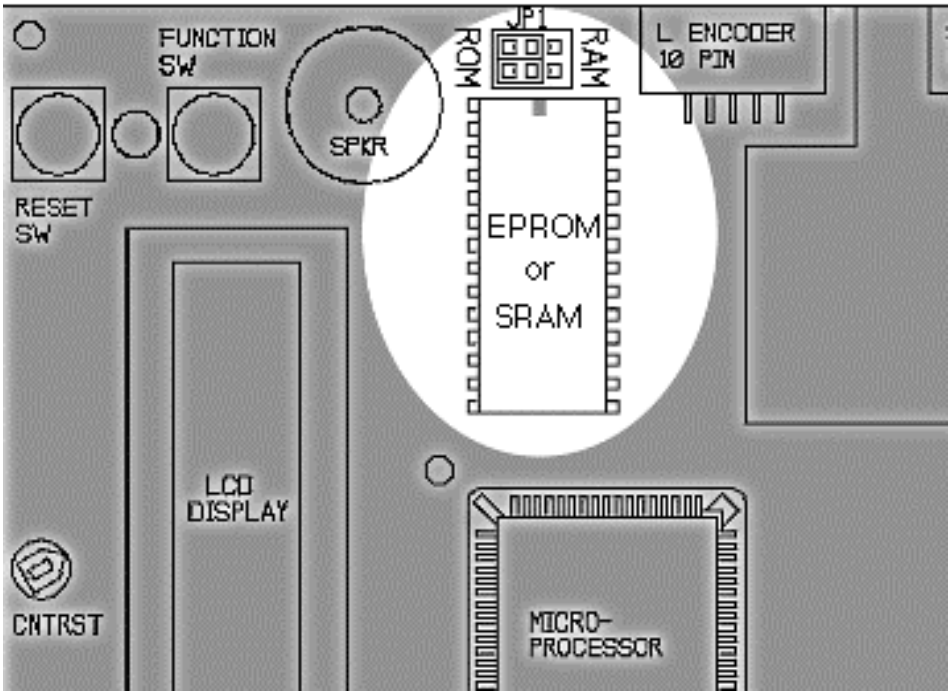


Figure 8-1. Location of the EPROM/SRAM on the Pioneer microcontroller board.

Pry or pull out the socketed chip. We'd prefer that you use a special "chip-puller" tool, but if one isn't handy, use a thin flatbladed screwdriver and gently pry up each end of the chip until it lifts out of the socket freely.

The EPROM and SRAM chips are sensitive to static electricity, so be very careful to handle them by their case, not touching the pins with your fingers. Store the unused chip on conductive foam. And be particularly careful not to bend any of the chip's 28 delicate connector pins.

Before installing the new chip, locate and move the two chip-select jumpers just at the head of the socket. Reposition them so that the jumpers connect the center and SRAM-marked side pins. Do not rotate the jumpers 90 degrees (Figure 8-2).

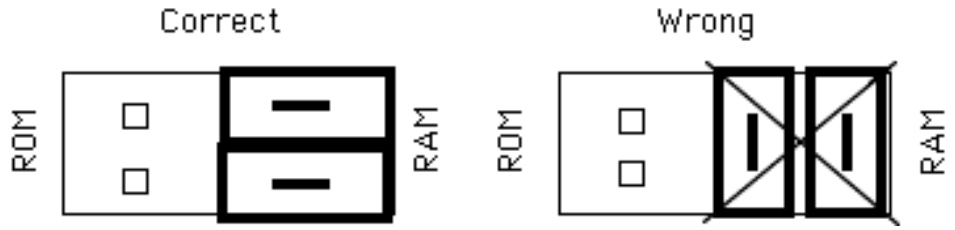


Figure 8-2. Proper standalone mode SRAM jumper positions.

Now install the SRAM chip. Make sure the pin-1 notch points towards the outer edge of the board and that all the pins slide into their respective sockets. Often the best technique is to align one side of pins slightly into their sockets, then gently press the chip from the other side until the other pins align and slide into their sockets.

Finally, gently seat the chip all the way into the socket. Your thumb is the best tool for the job.

> You can damage both the chip and the processor
> if you install the EPROM or SRAM backwards in the socket.

Reassembling the Console

Invert the microcontroller card and reinstall it on the Console top plate. Align the screw mounts and controls with their respective holes through the plate and replace the six hex screws which hold the microcontroller to the Console top plate.

Replace the various wire harnesses in their respective sockets. We find it's easiest to install the ones on the back of the board first, then the ones on the front. On the back—the edge that normally faces the rear of the robot when finally installed—the connectors and cables include the power harness and, if one is installed, a serial cable from the modem. On the front are at least three plugs and sockets: the two drive/encoder sockets (the longer of the two cables goes into the left drive socket—on the right side as you face the board) and the sonar cable between them. All of the sockets are keyed so you can't install the plug upside down. And simply match up the number of sockets to the same number of pins for each socket.

Lay the Console plate down onto the Console, align the screw mounts, and replace the six hex mounting screws. If you have a radio modem installed, replace its antenna.

Voila! You are finished.

Standalone Operation

We do not currently support Pioneer standalone operation. We do plan to provide extensive support for low-level operation and study of Pioneer's mobile and sensor systems in future Pioneer packages, using the Interactive C native (ICn) language. We recommend you obtain and use Interactive C from any of the various archives on the Internet. It is a subset of the C language with added functionality for multitasking and real-time control on the MC68HC11 processor. Programs written in this language are compiled on the host computer into pseudo-codes ("pcodes"), and downloaded to the robot where they run on a pcode interpreter. The language also supports a simple download program for sending compiled pcode programs to the robot.

Download Mode

To download your Pioneer executables into the onboard SRAM, we assume, of course, that you have connected the Pioneer to a basestation computer, either through a tether cable connected to the serial port or through the optional radio modems.

To communicate with the standalone Pioneer 1, whether via a download program, ICn, or another tool, you must first put the robot into standalone "download mode." (See Chapter 3, "Specifications and Controls", especially Figure 3-3, for details about the Pioneer controls.)

Switch the robot's MAIN POWER ON. You should *not* hear the distinctive "beep, beep" sounds characteristic of the Pioneer server EPROM software, nor should anything, except perhaps some gibberish, appear in the LCD. If you do see a message or hear the boot sounds of the Pioneer server, please review the beginning of this chapter and install the SRAM for standalone Pioneer operation.

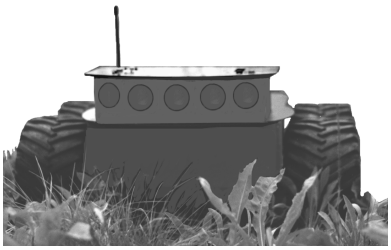
Press and hold down the black FUNCTION button next to the LCD. While still holding down the FUNCTION button, press and release the red RESET button. Now release the FUNCTION button. The amber XMT LED next to the serial port should momentarily light when the RESET button is first pressed, then extinguish within a second or so after you release the RESET button. The Pioneer is now in standalone download mode.

Repeat the sequence if you doubt its veracity—understandable, since there is no special indication for standalone download mode on the robot. The LCD, for example, displays nothing, except perhaps some gibberish.

Downloading MC68HC11 Programs

The `psosdl` software discussed in the "Updating and Configuring PSOS" Chapter 7 may be used to download your own Pioneer software besides PSOS. We don't otherwise support standalone mode for the Pioneer 1. We may someday. In the meantime, consult the Interactive C programming manuals for the MC68HC11 for software development and downloading details.

Once finished downloading the program, press RESET your Pioneer to execute it. (Do not turn off the power if you are using volatile SRAM). Also, be sure to activate the MOTORS (ON button) if your program uses the Pioneer drives.



Maintenance & Repair

The Pioneer is built to last a lifetime, and requires little maintenance.

Drive Lubrication

The drive motors and gearbox are sealed and self-lubricating, so you need not fuss with grease or oil. An occasional drop or two of oil on the axle bushings between the wheels and the case wouldn't hurt. And keep those axles clear of carpet or other strings that may wrap around and bind up the Pioneer's drive.

Pioneer Battery

Lead-acid batteries like the one in Pioneer last longest when kept fully charged. In fact, severe discharge is harmful to the battery, so be careful not to operate the robot if the battery voltage falls much below 11 VDC.

Charging the Battery

Insert the battery charger that comes with your Pioneer into a standard 110 VAC three-pronged wall power receptacle. (Some users may require a special power adapter.) Then insert the round plug at the end of the cable that is attached to the charger into the CHARGE socket on the Console. The LEDs on the charger indicate charge status, as marked on its case.

It takes less than 12 hours—oftentimes just a few hours, depending on the level of discharge—to fully charge the Pioneer battery with the accompanying charger (roughly, three hours per volt). And you may operate the robot while connected to the charger—with limited mobility, of course. Just don't leave the charger connected for much longer than 24 hours.

Alternative Battery Chargers

The center post of the Charger socket on the Pioneer Console is the positive (+) side of the battery; the case is the negative (-) side. If you choose to use an alternative battery charger for the Pioneer 1, be sure to connect positive to positive and negative to negative from charger to Pioneer.

An alternative AC to DC converter/battery charger for Pioneer should sustain between 0.75 and 1 A at around 13.75 to 14 VDC. It also should be voltage and current limited so that it cannot overcharge the battery.

Replacing the Pioneer 1 Battery

Expect years of service, but no battery lasts forever. Hence, we've made the one in Pioneer easy to replace. Make sure the MAIN POWER is OFF. Remove the ten hex screws which secure the deck plate to the main body of Pioneer 1—five on top on each side including the four that attach to the nose.

> Be careful to maintain the correct polarities when changing the battery or you could damage your robot.
> The terminals are marked: red, positive (+) and black, negative (-).
> The yellow wires on the power harness are (+); black wires are (-).

Once loosened, rotate the Deck plate 90 degrees clockwise to expose the battery which is mounted in the center inside of the body. There should be no need to disconnect any cables from the microcontroller board or drives.

Remove the two Phillips-head screws which secure the battery bracket to the floor of the case. Let loose the grounding strap lug and lift the bracket out of the robot.

Set the battery upright—terminals on top. First slip the positive (+) terminal off its post on the battery—it's the one marked in red and connected to the yellow wires of the power harness. Then slip off the negative (-) ground terminal from the battery. That way, you reduce the risk of an inadvertent short. Finally, lift the battery out of the robot case.

Inserting the new battery essentially is the reverse process for removal of the old one: Drop in the new battery, terminal posts up. Slide the slip-on terminals of the power harness onto their proper posts on the battery—negative side first.

Turn the battery on its side and secure with the bracket. Be sure to also secure the grounding cable lug under one of the bracket screws. Rotate the deck plate counterclockwise back into position and secure with the ten hex screws you removed earlier. Done.

Replacing the Pioneer AT Battery

If you thought replacing the Pioneer 1's battery is easy, replacing the Pioneer AT's battery is even easier: With your fingers, simply unlatch the two fasteners on the back of the AT and raise the Deck. Pry off the battery power connector near the battery terminals and lift the battery out of the Body. Place the fresh battery in the battery cradle, replace the power connector to the battery, close the Deck, and refasten the latches. Tool-lessly quick and simple. If that took you more than a minute, you need practice!

Tightening the AT's Drive Belts

Each side of Pioneer AT's drives are physically ganged by a drive belt. With use, particularly after climbing, the belts may stretch or loosen. We've given you the tools to tighten them yourself: Besides hands and fingers, all you need is the largest of the three hex wrenches that came with your robot (9/64" one).

Remove the hex screw from the center of the rear wheel—the one that holds the wheel to the axle. Gently pull the wheel off the axle.

Using the same hex wrench, now loosen, but do not remove, the four screws that hold the axle hub to the Body.

Open the Body lid by unclasping the latches and lifting from the rear. Inside, you'll see the motors and axles, as well as the drive belts on each side. With thumb against the back outside the Body, place your fingers over the rear motor whose mount you loosened in the previous step. Apply pressure on the belt by closing your hand. Tighten it, so that the belt is taut, but not rigid.

While still applying pressure on the belt to tighten it with one hand, use the other hand to retighten the axle hub mounting screws.

To remount the wheel, loosely align it on the axle with one hand and, with the other hand, guide the hex screw through the center hole—easiest if the screw is fit on the end of the hex wrench. Partially tighten the screw to loosely hold the wheel to the axle.

Grip your fingers around the face of the wheel, and while gently pushing it against the axle, rotate the wheel until you feel it's four "keys"—nubs on the other side of the wheel—grip and then seat into the axle.

Finally, tighten the wheel screw and you are done.

Factory Repairs

If you are having *hardware* problems with your Pioneer and, after reading this manual, you are satisfied that it needs repair, here's who to contact:

pioneer-support@activmedia.com
(603) 924-2184 fax
(800) 639-9481 (voice)

In the body of your email or fax message, describe the problem in as much detail as possible. Also include your name, email and mail addresses, as well as phone and fax numbers, and when and how we can best contact you (we will assume email is the best manner, unless otherwise notified).

We will try and resolve the problem through communication. If the robot must be returned to the factory for repair, obtain a shipping and repair authorization code and shipping details from us first. We are not responsible for shipping damage or loss.

Do not return a robot without contacting us first and obtaining *shipping and repair authorization.*

Appendix A: I/O Ports & Connections

This Appendix contains pinout and electrical specifications for the external and internal serial ports on the Pioneer, and for the two expansion sockets on the microcontroller board.

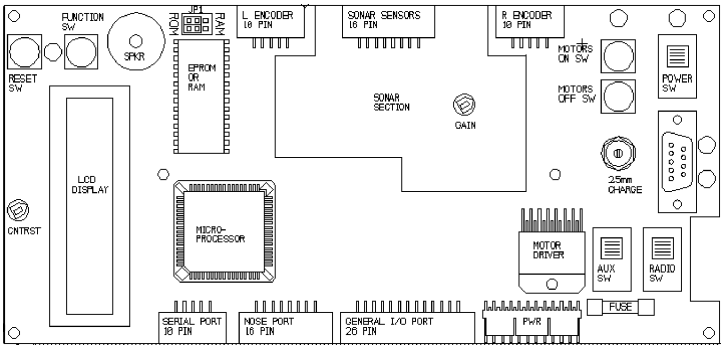


Figure A-1. Pioneer microcontroller ports and connectors.

Console Serial Port

Only three pins on the Console DB9 serial port connector are active: RS-232 compatible pins 2 (TxD), 3 (RxD), and 5 (signal ground). Table A-1 lists the cable connections for various host computers to this serial port. Except for DB-9-based straight-through cables like the one that comes with the radio modem, most common null-modem cables which cross-connect the transmit and receive lines work with the respective host and Pioneer 1.

Table A-1 Common serial cable connections to Pioneer

Platform & Connector	pin 2 (TxD)	pin 3 (RxD)	pin 5 (Gnd)
ComRAD radio DB9	2	3	5
Sun Sparcstation DB25	3	2	7
SGI Irix mini-DIN 8	5	3	4
PC COM*n*: DB9	2	3	5
PC COM*n*: DB25	3	2	7
Macintosh mini-DIN 8	3	5	4

The serial port is common RS232-compatible. The Pioneer operates at the following configuration; Pioneer software automatically configures the host computer's serial port for communication:

- 9,600 or 19,200 bits per second data rate
- Eight bits data
- One stop bit
- No parity
- No hardware handshaking (DTR)

Internal Serial Connector

The serial port connector on the Pioneer microcontroller supports both RS-232 serial communications, as well as the 68HC11's synchronous serial peripheral interface (Table A-2) . The latter serial interface's associated Peripheral Data Ports (PDA2-5) are supported in PSOS version 4.6 and later.

Table A-2. Internal Serial Port Connections

Pin #	**Label**		**Pin #**	**Label**
1	TxD		6	PD4
2	PD2		7	Gnd
3	RxD		8	PD5
4	PD3		9	No connection
5	Gnd		10	Vcc (5 VDC)

The Nose Expansion Port

At the rear of the Pioneer microcontroller card, next to the power connector, there is a 16-pin IDC socket we call the Pioneer "Nose" port (Figure A-1). It contains a number of I/O ports to the MC68HC11 microprocessor and support devices for custom and expansion hardware for the Pioneer robot. These include:

- 3 digital input (ID0-2)
- 3 digital output (OD0-2)
- 1 analog-to-digital input (A/D)
- 1 digital timer output (ODT)
- 6 signal ground (Gnd)
- 2 Vcc (+5 VDC)

The digital ports are biased to Vcc (+5 VDC); connect to Gnd to indicate input.

The A/D port produces an 8-bit digital representation of a 0 to +5 VDC analog input.

The ODT produces a high-precision (0 to +5 VDC; 1 microsecond resolution) waveform suitable for RC servo control projects.

The A/D and ODT ports also appear in the General I/O port connector (see below).

Note also that the Nose 16-pin and the General26-pin I/O connectors are numbered odd pins on top and even pins on the bottom; not top, left to right, then bottom left to right.

The pinouts for the Nose connector, for example, are:

1	3	5	7	9	11	13	15
2	4	6	8	10	12	14	16

> The Vcc ports supply *logic* power.
> They are not suitable for large currents, such as those drawn by RC servos.
> Use the Auxiliary 12V supply or Experimenter's I/O Module to power these devices.

Table A-3. Pioneer Nose Port Connections

Pin #	Label		Pin #	Label
1	ID0		9	OD0
2	Gnd		10	Vcc
3	ID1		11	OD1
4	Gnd		12	Vcc
5	ID2		13	OD2
6	Gnd		14	Gnd
7	A/D		15	ODT
8	Gnd		16	Gnd

The General I/O Expansion Port

At the back of the Pioneer microcontroller card, between the Nose Expansion and Serial Ports, there is a 26-pin IDC socket we call the Pioneer "General I/O" port (Figure A-1). It contains a number of I/O ports to the MC68HC11 microprocessor and support devices for custom and expansion hardware for the Pioneer robot. These include:

- 5 digital input (ID3-7)
- 3 digital output (OD3-7)
- 1 analog-to-digital input (A/D)
- 1 digital timer output (ODT)
- 1 digital timer input (IDT)
- 8 ground (Gnd)
- 2 Vcc (+5 VDC)

The digital ports are biased to Vcc (+5 VDC); connect to Gnd to indicate input.

The A/D port produces an 8-bit digital representation of a 0 to +5 VDC analog input.

The ODT produces a high-precision (0 to +5 VDC; 1 microsecond resolution) waveform suitable for servo control projects.

The IDT is a high-precision timer (one microsecond resolution; 65 microseconds duration). Like the digital ports, it is biased to Vcc, so connect it to Gnd to indicate input.

The A/D and ODT ports are shared by the Nose I/O port connector (see above).

Table A-4. General I/O Port Connections

Pin #	Label		Pin #	Label
1	A/D		14	Gnd
2	Gnd		15	ID3
3	IDT		16	Gnd
4	Gnd		17	OD7
5	—		18	Gnd
6	Vcc		19	OD6
7	ID7		20	Gnd
8	Vcc		21	OD5
9	ID6		22	Gnd
10	Gnd		23	OD4
11	ID5		24	Gnd
12	Gnd		25	OD3
13	ID4		26	Gnd

Appendix B: Radio Modem Configuration

The optional radio modem pair—one inside the Pioneer and an external modem for connection to a user-supplied computer—are a set; you usually cannot mix pairs. In fact, the modem manufacturer can deliver twelve different crystal frequencies around 900 kilohertz, so that up to twelve radio-modem equipped Pioneer robots can operate in the same general space.

The DIP switches on the side of both modems come preconfigured for operation with the Pioneer 1. These settings should not be changed (Table B-1).

Table B-1. Radio modem DIP switch settings

Switch	1	2	3	4	5	6	7	8	9	10	11	12
	d	u	u	d	u	d	d	d	d	u	u	u

d=down; u=up

Appendix C: Pioneer's Saphira Parameter Files

Saphira consults a special parameters file to determine various operating parameters for the robot server to which it connect. All the latest Pioneer 1s with PSOS 4.1 and later are described by the psos41x.p or psos41m.p parameter file found in $(SAPHIRA)/params. The difference is in the types of motors used in Pioneer 1s built before versus after November, 1996. The Pioneer AT is described in a single parameters file: pionat.p.

PION1X.p

```
;; Saphira parameters describes the Pioneer 1
;; Oldest motors ("x") version
;;
Class                   Pioneer
Subclass Pion1m.p
;;
AngleConvFactor         0.0061359   ; radians per encoder count diff (2PI/1024)
DistConvFactor          0.07979     ; 5in*PI / 5000 (mm/count)
VelConvFactor           3.9898      ; mm/sec per encoder count per 1/50 sec
RobotRadius              220.0      ; radius in mm
RobotDiagonal             90.0      ; half-height to diagonal of octagon
Holonomic               1           ; turns in own radius

;; These are for seven sonars: five front, two sides
;;
;; Sonar parameters
;;                      SonarNum N is number of sonars
;;                      SonarUnit I X Y TH is unit I (0 to N-1) description
;;                      X, Y are position of sonar in mm, TH is bearing in
degrees
;;
RangeConvFactor         0.1734    ; sonar range mm per 2 usec tick
;;
SonarNum 7
;;          #   x   y   th
;;--------------------------
SonarUnit  0  100 100  90
SonarUnit  1  120  80  30
SonarUnit  2  130  40  15
SonarUnit  3  130   0   0
SonarUnit  4  130 -40 -15
SonarUnit  5  120 -80 -30
SonarUnit  6  100 -100 -90
SonarUnit  7    0   0   0

;; Number of readings to keep in circular buffers
FrontBuffer 20
SideBuffer  40
```

PION1M.p

```
;;
;; Saphira parameters file describes Pioneer 1
;; New "m"otors version
;;
Class               Pioneer
Subclass Pion1m
;;
AngleConvFactor     0.0061359 ; radians per encoder count diff
(2PI/1024)
DistConvFactor      0.05066   ; 5in*PI / 7875 counts (mm/count)
VelConvFactor       2.5332            ; mm/sec / count  (DistConvFactor
* 50)
RobotRadius         220.0             ; radius in mm
RobotDiagonal        90.0             ; half-height to diagonal of LPS
display octagon
Holonomic           1                 ; turns in own radius
MaxRVelocity              100.0       ; degrees per second
MaxVelocity           400.0          ; mm per second
;;
;; These are for seven sonars: five front, two on sides
;;
;; Sonar parameters
;;                  SonarNum N is number of sonars
;;                  SonarUnit I X Y TH is unit I (0 to N-1) description
;;                  X, Y are position of sonar in mm, TH is bearing in
degrees
;;
RangeConvFactor     0.1734            ; sonar range mm per 2 usec tick
;;
SonarNum 7
;;          #   x   y   th
;;-------------------------
SonarUnit  0  100 100  90
SonarUnit  1  120  80  30
SonarUnit  2  130  40  15
SonarUnit  3  130   0   0
SonarUnit  4  130 -40 -15
SonarUnit  5  120 -80 -30
SonarUnit  6  100 -100 -90
SonarUnit  7    0   0   0

;; Number of readings to keep in circular buffers
FrontBuffer 20
SideBuffer  40
```

PIONAT.p

```
;;
;; Saphira parameters file describes Pioneer AT
;;
Class               Pioneer
Subclass PionAT
;;
AngleConvFactor    0.0061359     ; radians per encoder count diff
(2PI/1024)
DistConvFactor     0.04          ; 5in*PI / 7875 counts (mm/count)
VelConvFactor      2.5332        ; mm/sec / count  (DistConvFactor *
50)
RobotRadius         330.0        ; radius in mm
RobotDiagonal       120.0        ; half-height to diagonal of LPS
display octagon
Holonomic          1             ; turns in own radius
MaxRVelocity        100.0        ; degrees per second
MaxVelocity         400.0        ; mm per second

;; These are for seven sonars: five front, two on sides
;;
;; Sonar parameters
;;                 SonarNum N is number of sonars
;;                 SonarUnit I X Y TH is unit I (0 to N-1) description
;;                 X, Y are position of sonar in mm; TH is bearing in
degrees
;;
RangeConvFactor    0.1734        ; sonar range mm per 2 usec tick
;;
SonarNum 7
;;         #   x   y   th
;;-------------------------
SonarUnit  0  100 100  90
SonarUnit  1  120  80  30
SonarUnit  2  130  40  15
SonarUnit  3  130   0   0
SonarUnit  4  130 -40 -15
SonarUnit  5  120 -80 -30
SonarUnit  6  100 -100 -90
SonarUnit  7    0   0   0

;; Number of readings to keep in circular buffers
FrontBuffer 20
SideBuffer  40
```

Appendix D: Modifying Pioneer's Microcontroller v1.2

The download program `psosdl` will work *only* with Pioneer microcontrollers Version 1.2 and later. If you have a microcontroller that was manufactured prior to February, 1996 and are still running PSOS 3.x, you need to replace the microcontroller. Contact `pioneer-support@activmedia.com` for details. Microcontrollers Version 1.3 or later need no modifications.

With microcontrollers Version 1.2, `psosdl` still won't work, even though you have a flash PROM installed (you get "Board not responding" when attempting download of the new PSOS image). You will need to insert a short jumper onto the board to get it work.

Before you make any modifications, make sure you have a Version 1.2 microcontroller and that you have a flash PROM installed. For the latter information, read Chapter 7 "Updating and Configuring PSOS" earlier in this document. To determine which version microcontroller you have, remove it from the robot (see Chapter 8 "Standalone Pioneer" for details). The version number is printed on the board just above the LCD display.

On the Version 1.2 microcontroller, locate the RAM/ROM shunts beside the PROM (see Figure D-1).

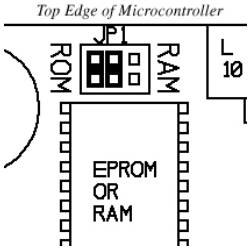


Figure D-1. Original RAM/ROM shunts on the Pioneer Microcontroller 1.2

Remove the top shunt and put a jumper across the top middle pin to the one on the bottom right, as shown in Figure D-2. Solder a short wire into place is best, but a good wire-wrap will work fine, too. Make sure the wire doesn't short the other pins, and leave the bottom shunt in place. Voila'!

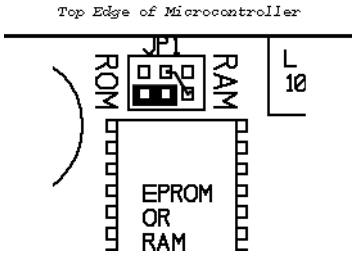


Figure D-2. Flash PROM-ready jumper and shunts on the Pioneer Microcontroller 1.2

Appendix E: Specifications

Physical Characteristic	Pioneer	Pioneer AT *"The Outlaw"*
Length	45cm	Same
Width	36cm	50cm
Height (body)	22cm	24cm
Height (antenna)	32cm	44cm
Body clearance	3.5cm	5.5cm
Weight	7.7kg	11.3kg
Payload	4.5kg	4.5kg
Construction		
Body	1.6mm CNC fabricated, painted aluminum	Same
Console & main deck	2.4mm CNC fabricated aluminum	Same
Assembly	Allen hex screws	Same
Power		
Battery	12V sealed, lead-acid	Same
Charge	84 watt-hr	84 or 144 watt-hr
Run time	8-10 hrs	2-3 hrs
Recharge time	8 hrs	Same
Mobility		
Drive wheels	2 solid rubber, plus rear balancing caster	4 soft rubber, with deep chevron tread
Wheel diam.	12.5cm	16cm
Wheel width	3.7cm	11cm
Steering	Differential	Same
Gear ratio	19.7:1	19.7:1
Pushing force	2kg	8kg
Swing radius	30cm	Same
Turn radius	0cm	Same
Translate speed max	0.6 m/sec	1.5 m/sec
Rotational speed max	150 degrees/sec	360 degrees/sec
Traversable step max	2cm	8.9cm
Traversable gap max	8.9cm	12.7cm
Traversable slope max	20% grade	80% grade
Traversable terrains	All wheelchair accessible	All surfaces
Sensors		
Ultrasonic sonars	7 total:	Same
	1 each side	Same
	5 forward @ 15° intervals	Same
Position encoders	100 ticks per revolution	Same
Electronics (basic onboard microcontroller)		
Processor	MC68HC11F1 (16 MHz)	Same
Position inputs	2	Same
Sonar inputs	8 (multiplexed)	Same
Digital I/O	16 logic ports	Same
A/D	1 @ 0-5 VDC	Same
Digital timer output	1 @ 1µsec resolution	Same

Physical Characteristic	Pioneer	Pioneer AT *"The Outlaw"*
Electronics (continued)		
Digital timer input	1 @ 1µsec resolution;	Same
	65µsec duration	Same
Comm port	1 RS-232 serial	Same
EEPROM	32 KB; PSOS-encoded software	Same
Power switches	1 main; 2 auxiliary	Same
Controls, Ports and Indicators (main console)		
LCD display	Systems status and messages	Same
Reset pushbutton	Warm reboot; yellow LED indicator	Same
Function pushbutton	Self-test	Same
Main power sw.	Robot power; 12VDC; red LED indicator	Same
Radio power sw.	Radio modem or other 12VDC	Same
Aux Power sw.	12VDC	Same
Motors pushbuttons	White enable, red disable; green LED indicator	Same
Speaker	Piezo	Same
Serial comm port	9-pin RS232 with RCV and XMT LED indicators	Same
Charger port	12VDC	Same

Index

Warranty & Liabilities

Your Pioneer is fully warrantied against defective parts or assembly for one year after it is shipped to you from the factory. This warranty explicitly *does not include* damage from shipping or from abuse or inappropriate operation, such as if the robot is allowed to tumble or fall off a ledge, or if it is overloaded with heavy objects.

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

Pioneer Mobile Robot Operation Manual, Edition 2, January 1998

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