Pioneer 2

MOBILE ROBOTS

with Pioneer 2 Operating System Servers

OPERATION MANUAL

THE PIONEER 2 MOBILE ROBOT

ActivMEDIA INCORPORATED

*Activ*MEDIA Robotics, A Div. of *Activ*MEDIA Incorporated
46 Concord Street, Peterborough, NH 03458, USA •
www.ActivMedia.com
603-924-9100 • Fax 603-924-2184

Pioneer 2 Mobile Robot – Operation Manual, January 1999

Table of Contents

Congratulations

on your purchase of this Pioneer 2 Mobile Robot, and welcome to the rapidly growing community of researchers, developers, and enthusiasts of Pioneer Mobile Robots .

This ***Pioneer 2 Operation Manual*** provides both the general and technical details you need to operate your Pioneer 2 Mobile Robot and to begin developing your own robotics hardware and software projects.

We encourage you to also use the companion resources that are packaged with your Pioneer Mobile Robot:

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

Pioneer Package

Our experienced robotics-manufacturing staff put your Pioneer 2 Mobile Robot and accessories through a "burn in" period and carefully tested them before shipping the robot to you. Our care extends beyond.

In addition to the companion resources listed above, we warranty the robot and our manufactured accessories against mechanical, electronic, and labor defects for one year. Third-party accessories are warranted by their manufacturers, typically for 90 days.)

All of these precautions ensure that you have many years to enjoy your new Pioneer 2 Mobile Robot. Even though we've made every effort to make your Pioneer package complete, please check the components again after you unpack them from the shipping crate.

Basic Components (all shipments)

- One fully assembled Pioneer 2 Mobile Robot
- Battery charger (some contain power receptacle and 220VAC adapters)
- 3.5-inch floppy disks containing licensed copies of Pioneer 2 software and documentation for Windows 95/NT platform
- Hex wrenches and assorted screws
- Replacement fuse
- Set of manuals
- Registration and Account Sheet

Optional Components and Attachments (partial list)

- Onboard EBX computer with PC104+ bus, hard-drive and other accessories
- Radio and/or Ethernet Modems
  - One mounted inside Pioneer 2 (antenna detached for shipping)
  - Companion radio (antenna also detached for shipping)
- Supplementary and replacement batteries
- 3-Battery Simultaneous Charge Station (110/220 VAC) Gripper

Fast-Track Color Tracking System

- Range-Finding StereoCam System
- Pan-Tilt-Zoom Surveillance System
- Custom Vision System
- Range-finding Laser
- GPS System
- Compass
- Bumper Ring
- Serial cables for external connections

User-Supplied Components / System Requirements

- Client computer: 486-class or later PC with Microsoft Windows 95-98/NT, FreeBSD, or Linux operating system; Power-PC Macintosh with System 7.5 or later; or any UNIX workstation
- One RS-232-compatible serial port

- `Four megabytes of available hard-disk storage`

Additional Resources

Every new Pioneer customer gets three additional and valuable resources:

1. A private account on our Internet server for downloading Pioneer software, updates, and manuals
2. Access to the private ***pioneer-users*** newsgroup
3. Direct access to the Pioneer technical support team

Pioneer Software

We maintain a 24-hour, seven-day per week Web server where customers can obtain Pioneer software and support materials:

http://robots.activmedia.com

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

Pioneer Newsgroup

We maintain an E-mail-based newsgroup through which Pioneer owners share ideas, software, and questions about the robot. To sign up, send an E-mail message to our automated newsgroup server:

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

Our SmartList-based listserver will respond automatically. After you subscribe, send your E-mail comments, suggestions, and questions intended for the worldwide community of Pioneer users:

```
To: pioneer-users@activmedia.com
From: <your return e-mail address goes here>
Subject: <something of interest to 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 issues concerning the operation and programming of Pioneer.

Support

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

pioneer-support@activmedia.com

Your message goes directly to the Pioneer technical support team. There a staff member will help you or point you to a place where you can find help.

Because this is a support option, not a general-interest newsgroup like **pioneer-users**, we reserve the option to reply only to questions about problems with Pioneer.

(See also Chapter 8, *Maintenance & Repair, for details*.)

What Is Pioneer?

Pioneer is a family of mobile robots, both two-wheel and four-wheel-drive. They are all small, intelligent robots, whose architecture was originally developed by Kurt Konolige, Ph.D., of SRI International.

These are truly off-the-shelf, "plug and play" 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 integrated sensors and accessories. They are all managed via an onboard microcontroller and mobile-robot server software.

Your Pioneer 2 also has a variety of expansion power and I/O ports for optional, custom attachments. It includes a complete addressable I/O bus for up to 16 devices and two RS-232 serial ports, one for communication between the robot server and a client computer, and the other serial port for accessory systems, such as the PTZ Robotic Camera.

With the onboard computer, Pioneer 2 has four additional RS-232 ports, a PCI bus and space for up to five PC104+ accessory cards. With Ethernet-ready onboard autonomy, the Pioneer becomes a plug-and-play agent for multi-intelligence work.

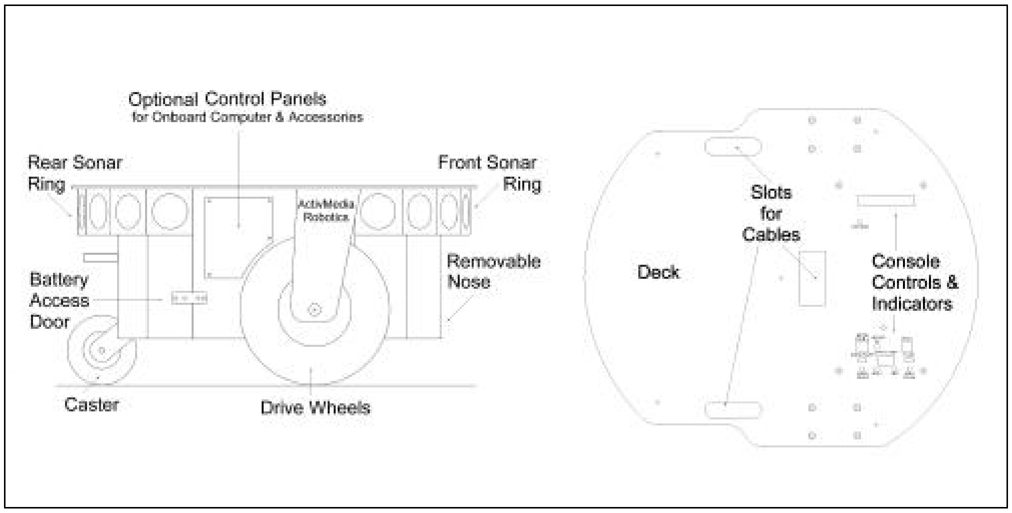


Figure 2-1. The Basic Components of Pioneer 2: Diagram Labels Two Views

Client Software

Pioneer 2 comes with a variety of mobile robotics development 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)
- Ayllu subsumption-like system, with multi-agent extensions

The Windows 95/NT versions of complimentary versions of software with manuals arrive with Pioneer 2. Other versions and all updates for supported computing platforms are available to password-registered customers for download from our software website (see *Congratulations*, Chapter 1):

http://robots.activmedia.com

Currently supported client operating systems include most UNIX, Apple Macintosh, SunOS and Solaris from Sun Microsystems, Linux, Silicon Graphics' IRIX, and Microsoft's Win32 (32-bit) Windows.

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

Pioneer Application Interface (PAI) is a C-language development suite that works with Saphira. It lets software developers gain closer control of the low-level details of Pioneer server. 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.

Ayllu is a tool for development of behavior-based control systems for mobile robots. It extends subsumption-style message passing to the multi-robot domain, provides for a wide variety of behavior-arbitration techniques, and allows a great deal of run-time system flexibility. This includes a dynamic reconfiguration of behavior structure and redistribution of tasks across a group of robots as determined by either task constraints or changing availability of resources. See the *Ayllu Programmer's 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 robot, so that developers may conveniently perfect their application software, then run it without modification on the robot. Several clients also may share responsibility for controlling a single mobile server, which permits experimentation in distributed communication, planning, and control.

The Pioneer Legacy

Pioneer 1 was the original design. It introduced a 68HC11-based microcontroller and the Pioneer Server Operating System (PSOS) software.

Intended mostly for indoor use on hard, flat surfaces, the Pioneer 1 has solid rubber tires and a two-wheel differential, reversible drive system with a rear caster for balance. The Pioneer 1 came standard with seven sonar range finders (two side-facing and five forward-facing) and integrated wheel encoders.

Software-wise, the Pioneer 1 initially served as a platform for SRI International's AI/fuzzy logic-based Saphira robotics applications development, but it wasn't long before it's open architecture became the popular platform for the development of a variety of alternative robotics software environments.

Many developers created software that interfaced directly with PSOS. Others extended the capabilities of Saphira (PAI and P-LOGO are two good examples), while others have implemented alternative robotics-control architectures, such as the subsumption-like Ayllu.

Functionally and programmatically identical to the Pioneer 1, the four-wheel drive, skid-steering Pioneer AT was introduced for operation in uneven indoor and outdoor environments, including loose, rough terrain. Each side of the Pioneer AT is electronically and physically linked for evenly applied translational and rotational power and speeds.

Except for the drive system, there are virtually no operational differences between the Pioneer AT and the Pioneer 1: The integrated sonar arrays and microcontrollers are the same. The accessories available for the Pioneer 1 also work with the Pioneer AT. Further, applications developed for the Pioneer 1 work with little or no porting to the Pioneer AT, and vice versa.

The newest generation of Pioneer Mobile Robots – including the Pioneer 2 CE, DX, and AT models – improves upon the Pioneer 1-AT legacy while retaining its many important features. Indeed in most respects, particularly with applications software, Pioneer 2 works identically with Pioneer 1 models.

Sporting a more holonomic body, larger wheels and stronger motors for better indoor performance, the Pioneer 2-DX and CE models, like Pioneer 1, are two-wheel, differential-drive mobile robots. (The least expensive CE model comes with fewer options and expansion capabilities than the 2-DX.) Similarly powered by four motors, the Pioneer 2-AT (but not its original AT) has independent encoders / drivers, as well as a stall-detection system, for each of its four drive motors.

All Pioneer 2 models use a high-performance 20 MHz Siemens 88C166-based microcontroller, with independent motor/power and sonar-controller boards for a versatile operating environment. The controller has two RS232-standard communications ports and an expansion bus to support the many accessories available for Pioneer, as well as your own custom attachments. And the Pioneer 2 comes with 9,850 ticks-per-revolution encoders on each and every wheel for finer odometry and translational and rotational speeds control. The Pioneer 2 also supports a full complement of eight sonars (front and rear) for nearly seamless object detection.

Software-wise, the Pioneer 2 is upwardly compatible with Pioneer 1: The Pioneer 2 Operating System (P2OS) software extends – but does not replace – the original PSOS. This means that even programs that interface at the lowest communication levels will work with *both* Pioneer 1-AT and with Pioneer 2 platforms. This also means that the higher level clients, such as Saphira, PAI, Ayllu, P-LOGO, and others, will work with P2OS and the host Pioneer 2 platform just as they had worked with Pioneer 1-AT. Of course, you will have to extend your client software, as we have done with Saphira, Ayllu and others, in order to take full advantage of P2OS.

To the relief of those who have invested years in developing software for Pioneer 1, Pioneer 2 truly does combine the best of the new mobile robot technologies and capabilities with the tried-and-true architecture and popularity of the original Pioneers.

Modes of Operation

You may operate Pioneer 2 in one of three modes:

- Self-test
- Stand-alone
- Server mode

The Pioneer 2 controller comes with 32K flash-programmable, read-only memory (flash-PROM) as part of its Siemens 88C166 microprocessor, and an additional 256K of dynamic RAM: 288K total memory space for your stand-alone robotics programs.

But we don't recommend that you start learning C166 programming. Rather, the robot comes to you installed with the latest P2OS robotics server software.

In conjunction with client software, such as Saphira, Ayllu, and PAI running on the onboard PC or a user-supplied computer, P2OS lets you take advantage of modern client / server and robot-control technologies. They enable the Pioneer 2 to perform advanced robot tasks. (See Chapter 6, *Pioneer 2 Operating System*, for details.)

Most users run Pioneer in this server mode, because it gives them quick, easy access to the robotics functionality while working in high-level software on a familiar host computer.

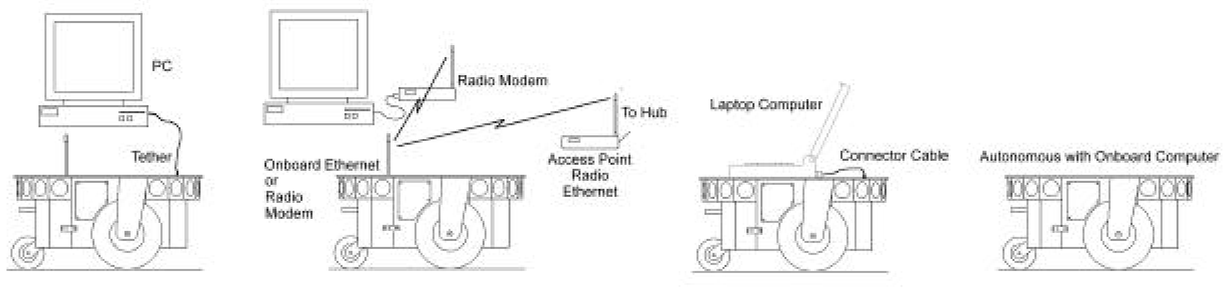

PC
Tether
Radio Modem
To Hub
Onboard Ethernet or Radio Modem
Access Point Radio Ethernet
Laptop Computer
Connector Cable
Autonomous with Onboard Computer

Figure 2-2. Serial communication options between Pioneer and an external computer

For experiments in microcontroller-level operation of robotics functions, you may reprogram the onboard PROM and RAM for direct and stand-alone operation of Pioneer 2. We supply the means to download, but not the microcontroller's programming software, for you to work in stand-alone mode.

In fact, the download utilities we provide for you to reprogram the 88C166-based P2 controller's PROM and RAM also are used to update and upgrade your P2OS. We typically provide the upgrades free for download from our website, so be sure to sign up for:

`pioneer-users` newslist

That's where we notify our customers of the upgrades, as well as where we provide access to Pioneer users worldwide.

Finally, we provide some test programs that exercise Pioneer 2 microcontroller hardware and software. We examine that mode in some detail in Chapter 5, *Self-Tests*.

Specifications & Controls

Pioneer robots may be smaller than most, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and more expensive machines. For example, Pioneer's modest size lends itself to navigation in tight quarters or cluttered spaces, such as classrooms, laboratories and small offices (see (Figure 3-1).

At the same time, the powerful Pioneer server, with its Saphira software client, is fully capable of mapping its environment, finding its way home and performing other sophisticated path planning.

Physical Characteristics

For a complete comparison of Pioneer 2's physical and operational specifications, see Appendix D.

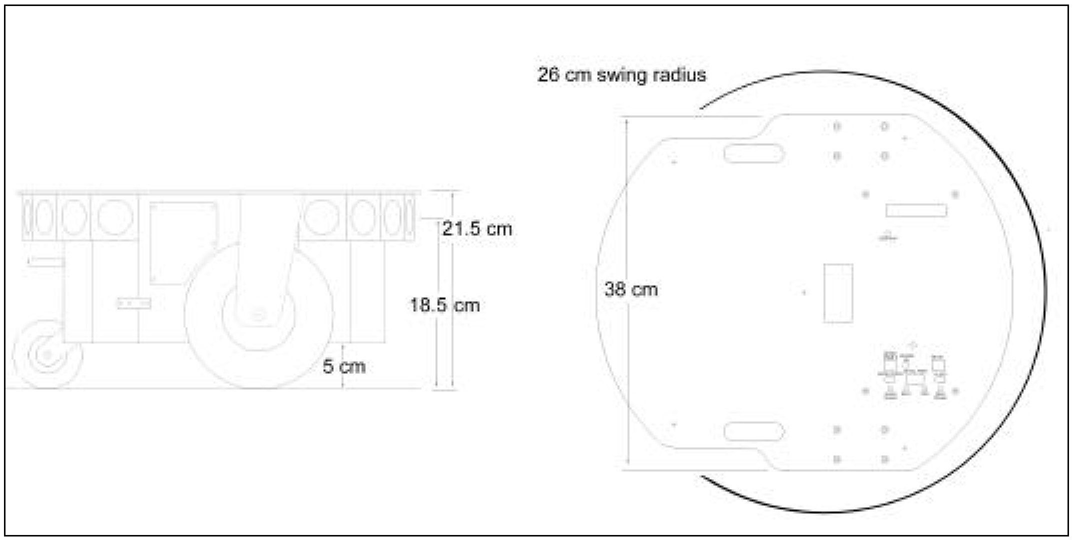


Figure 3-1. The Pioneer 2's physical dimensions and swing radius in centimeters

Weighing only 9 Kg (20 pounds with one battery), the basic Pioneer 2 Mobile Robot is lightweight, but its strong aluminum body materials, solid construction, and strut-reinforced wheels make it virtually indestructible.

These characteristics also permit the Pioneer 2 to carry a payload as heavy as 23 Kg (50 lbs.), yet it is lightweight enough that it is extraordinarily easy to transport – a task made even easier by its built-in handle. And because Pioneer's main components are assembled with Allen hex screws (metric sizes; tools included with robot), you have quick access to interior components and can attach accessories with a minimum of effort.

Main Components

The Pioneer 2 is composed of six main sections:

- Console
- Deck
- Body
- Nose
- Two Sonar Arrays

Console, Deck and Accessories

The Pioneer 2 Console and Deck are actually one piece – the top plate of the robot. The Deck is simply the flat surface for mounting projects and accessories, such as the PTZ Camera or a laser range finder. Feed-through slots behind the wheels let you conveniently route cables to the accessory side panels, and a removable plug in the middle of the Deck gives you convenient access to the interior of the robot.

Balance your robot's payload:
In general, you should try to center the robot's load over the drive wheels. If you must add a heavy accessory to the edge of a Pioneer 2-DX or 2-CE deck, counterbalance the weight with a heavy object on the opposite end of the deck. A full complement of batteries helps balance the robot, too.

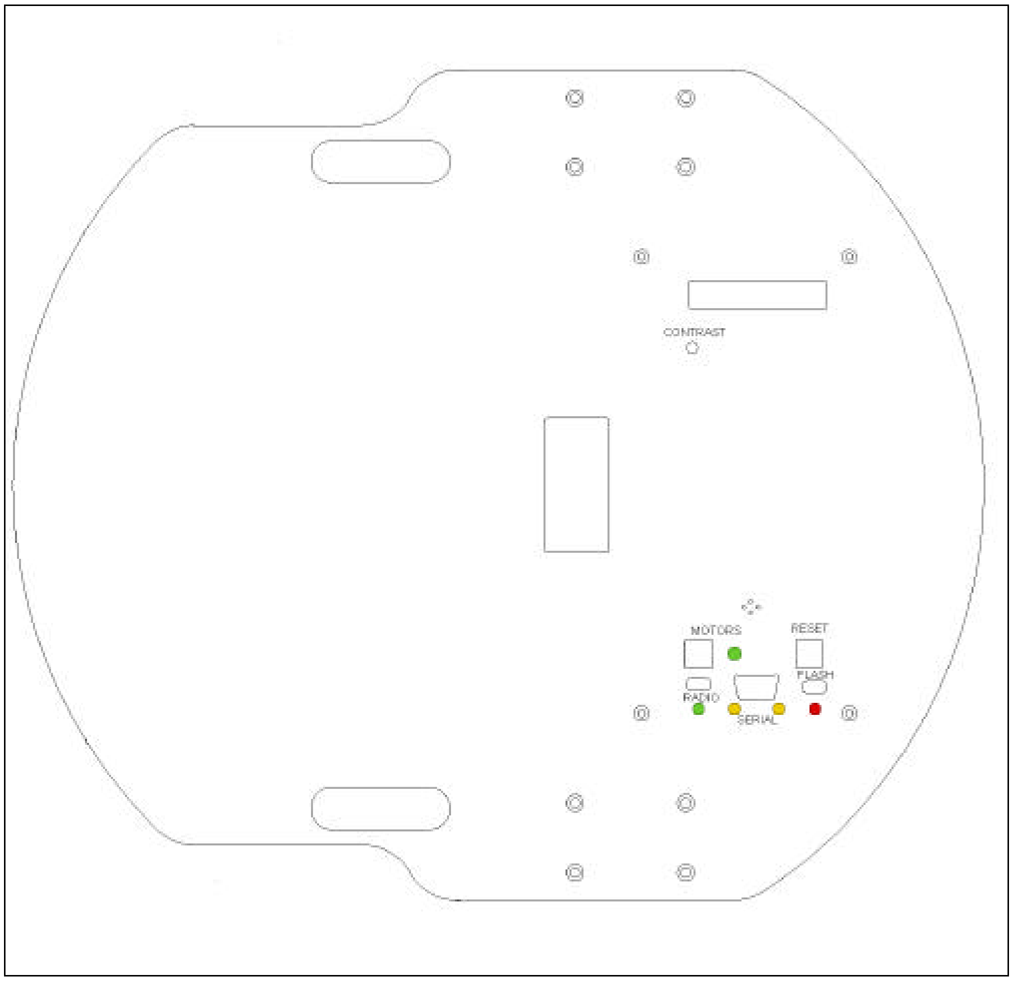


Figure 3-2. Pioneer 2's Console and Deck

The Console consists of a liquid-crystal display (LCD), MOTORS and RESET control buttons and indicators, and an RS232-compatible serial port at the 9-pin DSUB connector on the front and top of the Deck.

Attached directly underneath the Console is the Pioneer 2 microcontroller. (Operational details are provided in the next chapter, *Quick Start*.)

Body

Pioneer 2's Body houses its batteries, drive motors, electronics, and other standard components, including the forward and rear sonar arrays. The Body also has sufficient room, with power and signal connectors, to support a variety of robotics accessories inside, including the Fast-Track Color Tracking System, radio modems and radio Ethernet, onboard computer, and so on.

On the DX and AT models, the Pioneer 2 has a hinged rear door for easy access to the batteries. Accordingly, you may quickly hot-swap and replace any of up to three batteries.

Also at the rear of the Body is a convenient carry handle for easy transport of the robot.

Nose

The Nose of Pioneer 2 is empty, except when equipped with an onboard PC. The Nose is readily removable: Simply remove one screw from under the front sonar array, and a second screw from the bottom of the Pioneer 2, then slide the Nose away. This provides a quick and easy way to get at the PC104+ stack of accessory boards of the onboard PC, as well as to the sonar gain adjustment for the front sonar array (see Sonar Gain below). The Nose also is an ideal place for you to attach your own custom accessories and sensors.

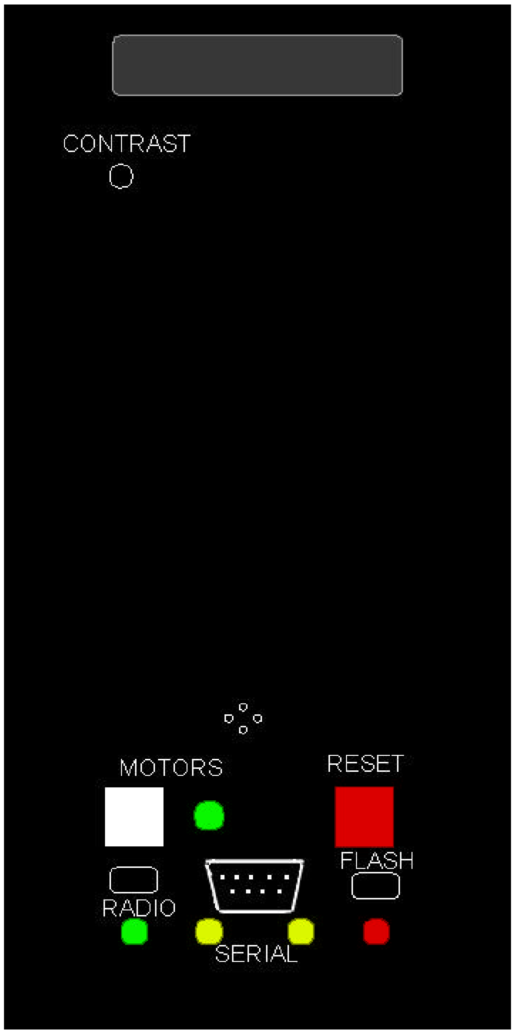


Figure 3-3. Pioneer 2 Console

Sonar Arrays

Pioneer 2 supports up to two sonar range-finder arrays. One array, affixed under the front of the Deck and atop the Nose, provides forward- and side-range sensing. The other, an optional sonar array is attached just beneath the rear Deck and provides rearward as well as side sensing.

Motors and Position Encoders

Pioneer 2's drive system uses pulse-width modulated (PWM), reversible-DC motors. Each drive motor includes a high-resolution optical quadrature shaft encoder that provides 9,850 ticks per wheel revolution (19 ticks per millimeter) for precise position and speed sensing and advanced dead-reckoning.

Sonars

Pioneer 2 supports both front and rear sonar arrays, each with eight transducers that provide object detection and range information for features recognition, as well as navigation around obstacles. The sonar positions are fixed in both arrays: one on each side, and six facing outward at 20-degree intervals, 17 cm high, providing more than 180 degrees of nearly seamless sensing, and all-around sensing when using both forward and rear arrays.

The sonar firing rate is 25 Hz (40 milliseconds per sonar) and sensitivity ranges from 10 cm (6 inches) to more than five meters (16 feet). (Objects closer than 10 cm are not detected.) You may control the sonar's firing pattern through software; the default is left-to-right in sequence for the forward array and right-to-left on the rear. One sonar from each array "ping" simultaneously.

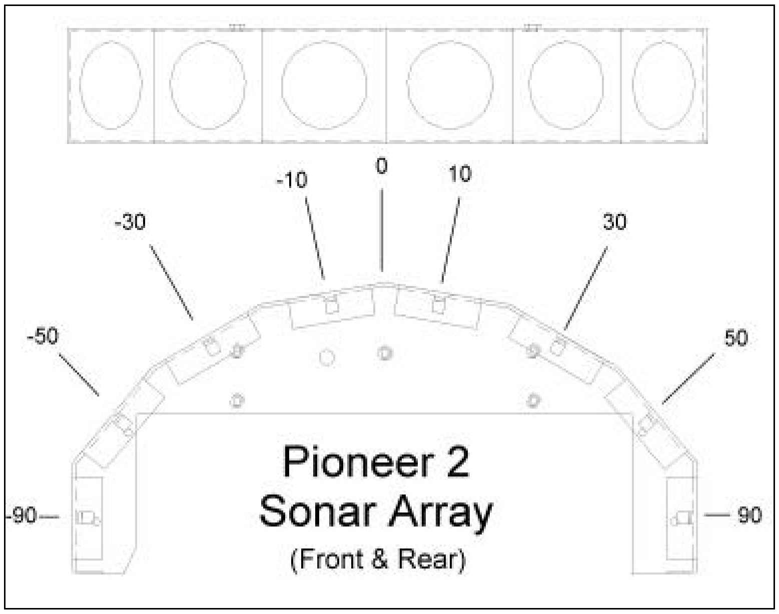


Figure 3-4. Pioneer 2 sonar array

Both the forward and rear sonar sensor arrays are preset at the factory. However, you may adjust their general sensitivity and range to accommodate differing Pioneer 2 operating environments.

The sonar gain control is on the underside of the sonar driver board, which is attached to the floor of each sonar module. Remove the Nose to access the front sonar module. Unlatch and open Pioneer 2's back door to access the rear sonar module.

From the underside, locate a hole near the front of the sonar module through which you can see the cap of the sonar-gain adjustment potentiometer. Using a flat-blade screwdriver, turn the gain control counterclockwise to make the sonar less sensitive to external noise and false echoes.

Low sonar-gain settings reduce the robots' ability to see small objects; under some circumstances, that is desirable. For instance, attenuate the sonar if you are operating in a noisy environment or on 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 sonar array by turning the gain-adjustment screw clockwise, making the sonar 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.

Batteries and Power

The Pioneer 2 CE has a single, twelve-volt direct-current (12 VDC), seven ampere-hour (84 watt-hour) sealed lead / acid battery which supplies ample power for Pioneer 2's drives, electronics, and accessories.

Typical intermittent operation of the motors gives six or more hours of Pioneer 2-CE use. If you don't use the motors, Pioneer's native electronics will run for several days on a single charge.

The Pioneer 2 DX and AT models may contain up to three hot-swappable, seven ampere-hour, 12 VDC sealed lead / acid batteries (total of 252 watt-hours), accessible though a hinged and latched back door.

Similar to the CE model, the battery life of the Pioneer 2 DX and AT, of course, depends on the configuration of accessories and motor activity. Pioneer 2-AT charge life typically ranges from four to eight hours. Pioneer 2-DX can run for more than a day.

Balance the batteries in your DX or AT.

IMPORTANT: Batteries have a significant impact on the balance and operation of your robot. For the Pioneer 2-DX and 2-AT, under most conditions, we recommend operating with three batteries. (See "Deck: Balancing Robot Loads," above.)

All Pioneer 2 models come with a battery recharger and power port for continuous operation of the robot's onboard systems while recharging. Typical recharge time varies according to the battery's discharge state; it is roughly equal to three hours per volt. The optional Charge Cube allows simultaneous recharge of three swappable batteries outside the robot.

With the optional high-Speed Charger, recharge time is greatly reduced.

Electronics

The Pioneer 2's standard electronics reside on two main boards: The microcontroller is mounted under the Console Deck, and the power / motor controller board is mounted on the back of the battery box inside the robot. A Main Power switch at the back of the robot controls power for the entire system. Processor control switches and indicators fit through the Deck Console.

Power/Motor Board

Inside the robot, mounted to back of the battery box, is the Pioneer 2 Motor-Power board. It supplies both the 12 VDC and five VDC power requirements of Pioneer 2's standard systems. Additionally, it has user-accessible five- and 12 VDC connectors, which supply 1–2.5 amperes of power for accessories, depending on configuration. The Motor-Power board also contains the motor-driver electronics, as controlled by the microcontroller.

The standard Motor-Power board has a 12-pin user-power connector that supports four sets of five- and 12 VDC power ports (total one ampere) for custom accessories. An optional computer-power section to the board supplies up to seven amperes at five VDC power for an onboard PC. The power board includes special low-power and power-down circuit that lets you gently shut down the computer without direct connection through a keyboard or monitor. (See Appendix B in this manual and the *Computer Tech Note*s pamphlet that accompany the Pioneer 2 for details.)

Microcontroller

Pioneer 2's microcontroller uses a 20 MHz Siemens 88C166 microprocessor with integrated 32K flash-PROM. The microcontroller also has 256K of dynamic RAM, two RS232-compatible serial ports, several digital and analog-to-digital I/O user-accessible ports, and an eight-bit expansion bus. (See Appendix A for port details.)

All the ports, except those for the motors, encoders, and sonar, are available to the user for Pioneer 2 accessory hardware, which you may control through the P2OS. Port connector pinouts and electronic details appear in the Appendices.

Sonar Board

Associated with each sonar array – forward and rear – is a sonar multiplexer / firing board. Wire leads to the individual sonar plug into a 16-pin connector on the board. A 10-conductor signal cable connects the sonar board with the microcontroller.

Controls, Ports, and Indicators

Main Power, Fuse, and Indicator

A single slide-switch on the rear left panel of the Pioneer 2 controls power to the entire robot and all its integrated accessories. Up is ON; down is OFF. A red LED on the Console indicates Main Power.

Inside, on the top right side of the battery box (accessible through the hinged back door of the Pioneer 2 DX and AT) is the Main Power Fuse. It is an automotive-type (spade terminals) 15A (DX, CE) or 30A (AT) fuse designed for tool-less replacement. To the left of the fuse, on the same board, is the power relay, which isolates the high-ampere draw of the robot system from the Main Power Switch.

Recharge/Power Port

Below the Main Power Switch is the battery Recharger port. It provides 12 VDC power to the robot's electronics, motors, and accessories, even without batteries.

You should maintain Pioneer’s battery in a charged state above 11 VDC, as indicated on the robot’s LCD. We recommend recharging the battery when it falls below 11 VDC, even though the robot may continue to operate below 10 VDC. The microcontroller will sound a warning when the battery voltage falls below that set level (see Chapter 7, *Updating and Reconfiguring P2OS*), and the optional computer power circuitry will automatically shut down the onboard PC. Discharging the batteries to below 10 VDC damages them.

Disengage the motors when recharging the robot.

You may continue to operate Pioneer while charging its batteries, although that will lengthen the recharge time.

If you have only one battery onboard, plug Pioneer into the charger to “hot-swap” the exhausted battery for a fresh one. To “hot-swap” two or three batteries, exchange each exhausted battery one at a time for a fresh one, leaving the remaining battery(ies) in place to supply power to the robot.

The Pioneer 2 charger also may be used with the optional Pioneer 2 Charge Cube.

Liquid-Crystal Display

When powered on, information about the robot's state and connections appears on a 16-character by two-line liquid-crystal display (LCD) on the left side of Pioneer 2’s Console.

When under control of the P2OS server, for example, the display shows the state of communication with the client computer, along with the battery voltage and a blinking "heartbeat" asterisk (*) as the last six characters in the second line of text.

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 well charged. Then, using a small, flat-blade screwdriver, turn the adjustment screw to darken or lighten the screen so that the characters are clearly visible under your lighting conditions.

RESET and MOTORS

The RESET (red) and MOTORS (white) push-button switches on the Pioneer 2 Console affect the microcontroller’s logic and motor driver systems. Both are under software control.

When pressed alone, RESET puts the microcontroller into its start-up state, disrupting any running program or client connection. It also disables the drive motors – just as if you cycle Main Power. But, unlike a cold-power restart, RESET preserves the contents of the Pioneer 2 microcontroller's RAM, so any user programs downloaded in stand-alone mode get restarted.

The MOTORS pushbutton is *NOT* a power switch – it does not directly control power to the motors. P2OS does and your stand-alone software must.

The MOTORS button and its associated green LED are under software control. Normally, Pioneer 2's motors are disabled when not connected to a client, or when not running the self-test or a stand-alone program on the controller. When running P2OS and connected with a client, the motors remain disabled (LED flashes) until you press and release the MOTORS button.

The green LED should light continuously when the motors are enabled and blink on / off when disengaged. Pressing and releasing the white MOTORS button enables / disables the motors when connected with a P2OS client.

Under P2OS control when *not* connected with a client, pressing and releasing the MOTORS button alone puts Pioneer 2 into a self-test mode that exercises the robot's drive, controller, and I/O systems. (See Chapter 5, *Self-Tests,* for details.)

Pressing and holding the MOTORS button in combination with pressing and releasing the RESET button puts the microcontroller board into a special download mode for reprogramming the onboard flash PROM and RAM. See *Pioneering 2 Operating System* and *Updating & Reconfiguring P2OS*, Chapters 6 and 7.

SERIAL

Pioneer 2's microcontroller has two serial ports and three connectors. One connector, a standard 9-pin D-SUB receptacle, is located on the Console and is for direct RS232-compatible serial data communication between the microcontroller and an client computer. SERIAL shares its three-line transmit, receive, and ground connections with one of the two serial ports that are inside the robot. (See Appendix A for pinouts and cabling connections.)

Two amber LEDs on either side of the Serial Port are lit during actual data activity transmitted from (`TX`) or received by (`RC`) the controller.

Remove any tether or laptop connection from the Console Serial Port when using the optional radio modems.

RADIO

The RADIO slide switch on Pioneer 2's Console controls power to the optional radio modem or Ethernet radio. The RADIO switch does not affect the SERIAL port functions directly, but you must switch a radio modem's power off if you use the Console Serial Port to connect a piggyback laptop or another external computer to the robot. (The radio modem gets connected to the microcontroller via the internal, shared serial port.)

FLASH

A slide switch labeled FLASH is recessed into the Console. It write-protects the flash PROM-stored P2OS software and your Pioneer 2's operating parameters (see *Updating and Reconfiguring P2OS*, Chapter 7). When switched forward, toward the front of the robot, FLASH is enabled for writing. (The P2OS utilities that accompany your robot will warn you if FLASH is disabled.)

Safety Watchdogs and Configuration

Pioneer's standard onboard software, P2OS, contains a communications watchdog that will halt motion if communications between a client computer and the server are disrupted for a set 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 stall monitor. If the drive exerts a PWM pulse that equals or exceeds a configurable level and the wheels fail to turn (`stallval`), motor power is cut off for a configurable amount of time (`stallwait`). When the time elapses, 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 Chapter 7, *Updating & Reconfiguring P2OS*, for details.)

Quick Start

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

The Pioneer 2 P2OS servers require a serial communication link to a client, including Saphira.
The serial link may be:

- A tether from the robot's 9-pin serial connector on the Console to a basestation computer
- A piggyback laptop cabled to the robot's 9-pin serial connector on the Console
- An optional radio modem pair – one inside Pioneer and its companion connected to the serial port of a basestation computer
- An integrated onboard PC wired internally to a serial port

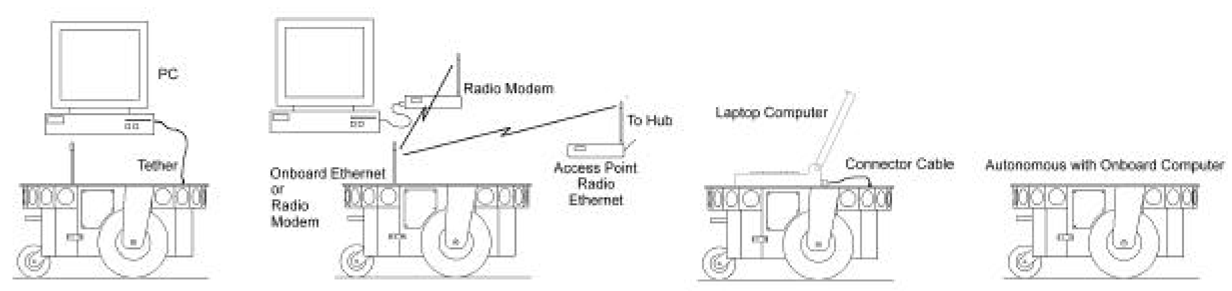

PC
Tether
Radio Modem
To Hub
Onboard Ethernet or Radio Modem
Access Point Radio Ethernet
Laptop Computer
Connector Cable
Autonomous with Onboard Computer

Figure 4-1. Pioneer 2's P2OS servers need a serial link with a client

Preparative Assembly

Out of the box, Pioneer comes fully assembled, with its batteries installed and fully charged – ready to drive right out of the box. However, you may need to attach an antenna or plug in an accessory that we intentionally leave unattached so as to prevent damage during shipping. Consult any Tech Notes and accessory assembly manuals that may accompany your Pioneer 2 for final assembly details.

Saphira Client Installation

The Saphira client-development software, including the Saphira demonstration program and Pioneer simulator, as well as other Pioneer-related software including Ayllu, PAI, and P-LOGO, come distributed as compressed archives of software. We include 3.5-inch, 1.4-megabyte floppy diskettes containing these software configured and compiled for Windows 95/98/NT systems. Pioneer owners should obtain other Saphira and Pioneer-related software versions and updates for other platforms from our support website (see Chapter 1, *Congratulations*, for details.) You will need the ID and password from your Registration Sheet in order to access portions of our customer website:

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

When installed, Saphira typically requires four megabytes (4 MB) of hard-disk space. After uncompressing them, you may find the various Saphira libraries and executables inside the Saphira directory.

To install the Saphira software, follow the instructions in the `README` file that accompanies the software for your platform. For instance, the Windows 95/98/NT archive is a self-extracting archive; simply double-click on its `exe` icon and follow the extraction program's instructions.

Saphira Client Start-Up

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 on the `saphira.exe` icon inside

```
C:\Saphira\ver61\bin
```

on your Windows 95, 98, or NT desktop, or navigate (`cd`) to the Saphira directory on your UNIX or Linux machine and type `saphira,` or `./saphira` to execute the Saphira client software there.

If the demonstration program is installed and executed properly, a Saphira main window should open and appear on your screen. Otherwise, diagnostic error dialog boxes will inform you of the problem. (Also see *Troubleshooting* below.)

Pioneer Cold Start-Up

Place your Pioneer 2 on the floor in an open space. Slide the Main Power switch up to `on`. The red power indicator lamp on the Console should light. After a short P2OS initialization phase in which the LDC displays the current P2OS version number, the LCD on the control panel then displays the current status, the Console serial port baud rate, your robot's serial number, and the battery charge, in volts. For example,

```
no conn   19.2kB
BBA-9999   13.8*
```

The asterisk following the battery voltage should be flashing.

The same P2OS initialization sequence occurs whenever you press the red RESET button. Unlike Pioneer 1, you cannot engage the drive motors until you have connected with a client.

RADIO ON

If you own radio modems for Pioneer client-server serial communications, switch on RADIO power.

Starting Saphira Client/Pioneer Server Communications

After it starts up, resets, or completes the self-test, P2OS enters a quiet state awaiting communication with a client computer.

To establish a connection with the Saphira client, pull down the `Connect` menu of Saphira's `main` window on your computer and engage the appropriate serial port: It's the one that you connected to the robot via a direct cable or through modems (Figure 4-1).

The port name varies by computer platform, such as `/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 on next page.)

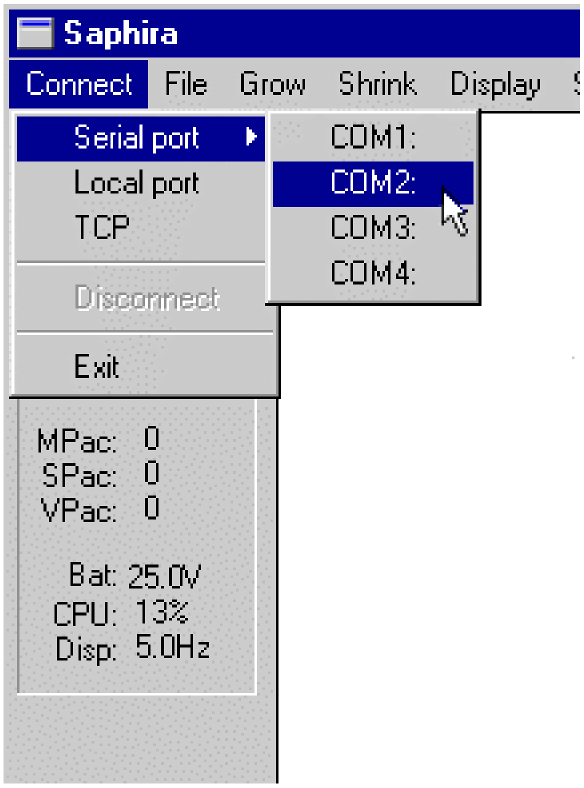


Figure 4-2. Connecting Saphira with Pioneer

The Saphira client initiates a connection with the P2OS server by exchanging three synchronization packets. You may monitor this process on Pioneer's Console LCD and in the Saphira client's Colbert interaction window.

As synchronization packets are received and echoed by the communications server, they appear sequentially next to the word `sync` on the top line of the LCD display on the robot. 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 indicating you have a noisy communication line; you must reduce the noise level before operating Saphira.

A Successful Connection

After Saphira negotiates a connection successfully, the client requests various P2OS servers to initiate their activities, including sonar polling, position integration, and so on. You should hear the Pioneer's sonar ping with a distinctive and repetitive clicking sound.

Press the white MOTORS pushbutton to enable the drive motors. The associated green LED should stop flashing and light continuously.

The amber SERIAL port indicator LEDs on the robot's Console should blink to indicate Saphira client / P2OS server communications. The Pioneer `LCD` also should display a message similar to the following:

```
Connected
P2OS 1.0       13.2*
```

Operating the Saphira Demo Client

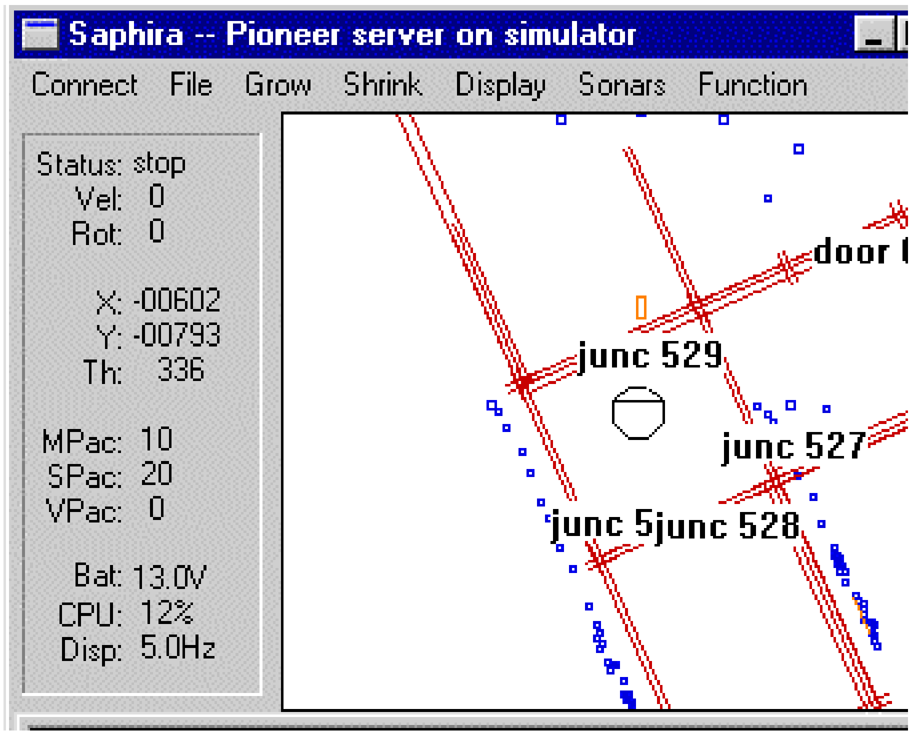
When communications between the Saphira client and the Pioneer server are established, the robot becomes responsive and intelligent. For example, although you may drive it manually toward an obstacle, Pioneer will not crash (unless its obstacle-avoidance behaviors have been disabled) because it can detect and actively avoids collisions.

Collision avoidance is just one of the many mobile robot behaviors available through Pioneer's suite of robotics application-development software. Remember, this section is meant to familiarize you with Pioneer. Please read the respective Pioneer software manuals for details.

Engage Pioneer 2’s motors (white Motor button) after connecting with a client or the robot won’t move, no matter how excited you get.

The main window of the Saphira client displays a sonar map built by Saphira as the Pioneer moves through space. Landmarks may be defined so that Saphira will classify certain sensor data patterns, for example, walls or openings.

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 directly 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.


Saphira -- Pioneer server on simulator
Connect File Grow Shrink Display Sonars Function
door
junc 529
junc 527
junc 5junc 528

Status: stop
Vel: 0
Rot: 0

X: -00602
Y: -00793
Th: 336

MPac: 10
SPac: 20
VPac: 0

Bat: 13.0V
CPU: 12%
Disp: 5.0Hz

Figure 4-3. The main window of Pioneer’s Saphira client

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).

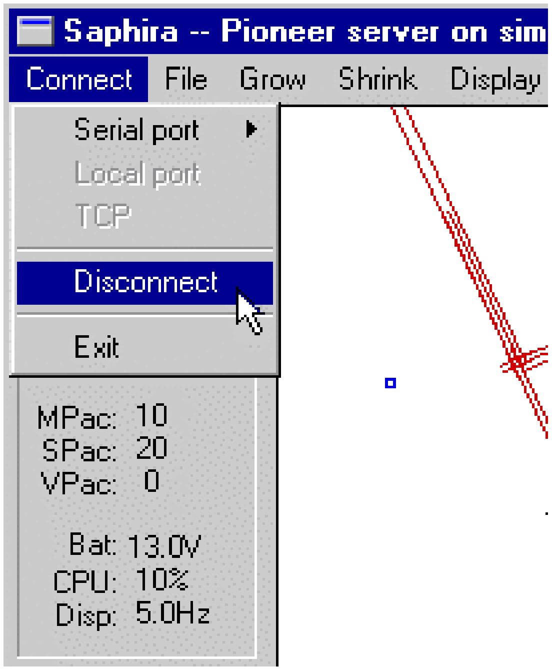
When manually joysticking the robot, each keypress moves the robot forward or backward faster or slower and incrementally changes its direction. For instance, when turning, it is often useful to push the left- or right-turn key rapidly several times in a row, because the turn increment is small.

Table 4-1. 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 you finish playing with Pioneer 2, pull-down the Saphira client's Connect menu with your mouse and choose the Disconnect option (Figure 4-4).



MPac: 10
SPac: 20
VPac: 0

Bat: 13.0V
CPU: 10%
Disp: 5.0Hz

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

The Pioneer will disengage its drive motors and stop moving automatically, and its sonar should stop firing. Its LCD also should return to the waiting-state message, and the motors should become disabled. You may now slide the Pioneer's Main Power switch to off.

Quickstart Troubleshooting

You must have a valid Saphira license to connect with Pioneer or any other robot. Unlicensed Saphira clients may only connect with the Pioneer simulator.

Licensed versions of Saphira come with the robot and can be downloaded by Pioneer customers from the *Activ*MEDIA Robotics support website.

Most problems occur when attempting to connect the Saphira client with the Pioneer server for the first time. Make sure you have Saphira properly installed and its related SAPHIRA and LD_LIBRARY_PATH (UNIX/Linux only) environment variables set. It's also a good idea to recheck that the serial cable is plugged into a working serial port on your computer.

UNIX and Linux users should be sure they have permission to read / write that port. On the server side, make sure your radio modem is on, if that is the connection route.

If you access the wrong port, the Saphira demonstration program will complain, "Error opening" the selected serial port. If the robot server isn't listening, or if the serial link is severed somewhere between the client and server (cable loose, or a modem off, for instance), the client will attempt to "Syncing 0" six times and fail with a "Connection refused." In that case, reset the robot and check your serial connections. For instance, if you are using the InfoWave radio modems, the DCD lamp on the host should light up. If it doesn't, it means it cannot find the one in the robot.

Once successfully connected, remember that the robot won't move unless its motors are engaged, and that you have to manually engage the motors just after you make a connection with a client; not beforehand.

If for some reason, communications sever between the Saphira client and Pioneer server, 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.

To test for range limits, simply pick up the robot and move it closer to the basestation radio modem. If the robot was out of range, the connection should resume. If not, check to make sure that radio modems were not inadvertently switched off.

Communications will also fail if the client and / or server is somehow disabled during a session. For instance, if you inadvertently switch off the robot’s Main Power or press the Reset button, you must restart the connection. Turning the Main Power switch off and then back ON or pressing the Reset button puts the robot server back to its wait state, ready to accept client connections again. If the Saphira client application is still active, simply open the Connect menu and choose the Disconnect option. Otherwise, restart the application and reconnect the Saphira client with the Pioneer server.

Self-Tests

P2OS comes with a series of short test routines for the Pioneer 2's drive motors and sonar. To run the self-tests, start up or reset the robot into its P2OS wait state (LCD "no conn"). You may press the Reset button at any time to disable the self-tests.

Place the Pioneer robot on the floor and have everyone step back
before engaging self-tests.

Motors Test

The first Self-Test exercises Pioneer 2's drive motors. During this test, the robot is not at all conscious of bystanders. Please have everyone step back and remove any obstacles from within a diameter of four to five feet around the Pioneer. When ready, press the white Motor button once.

The motors' self-test begins by engaging the left drive wheel, first forward, then in reverse, each to complete a partial turn clockwise, then counterclockwise. Similarly, the right wheel engages, first forward, then reverse to complete partial turns, first counter-clockclockwise, then clockwise.

The LCD, although difficult to read while the robot is in motion, displays a message for each test. For example:

```
Left forward
P2OS 1.0      13.2*
```

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

Sonar Test

Pioneer's self-test series automatically continues to the next test by individually firing and reading echoes from its sonar arrays.

You should hear the distinctive clicking sound as each sonar "pings" individually (one forward and one rear simultaneously) for about five seconds, each in order from left to right in front and right to left in the rear. The LCD tells you the sonar number and the number (in hexadecimal notation) of microseconds it takes to receive an echo from a nearby object, such as your hand.

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

```
FS:#3 034A
RS:#3 7D0012.4V*
```

The sonars are numbered 0 through 7 regardless of whether they are located front or rear. Absent echoes return at the maximum time of 0x7D00.

THE PIONEER 2 MOBILE ROBOT

OPERATING SYSTEM

Pioneer 2 Operating System

All Pioneers use a client / server robotics architecture developed by Dr. Kurt Konolige and others at SRI International, including the Pioneer 2 Operating System (P2OS; see Figure 6-1). In the model, the server works to manage all the low-level details of the robot's systems. These include operating the motors, firing the sonar, collecting sonar and motor encoder data, and so on – all on command from and reporting to a separate client application, such as Saphira.

With this client/server architecture, 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 Pioneer via the P2OS client/server interface. The same P2OS functions and commands are supported in the various client-programming libraries accompanying Pioneer.

Experienced Pioneer users can be assured that P2OS is directly upwardly compatible with PSOS, implementing all the same commands and information packets. P2OS, of course, extends the servers to add new functionality, improve performance, and provide additional information about the robot's state and sensing. Hence, P2OS-specific programs may not operate on Pioneer 1s or ATs.

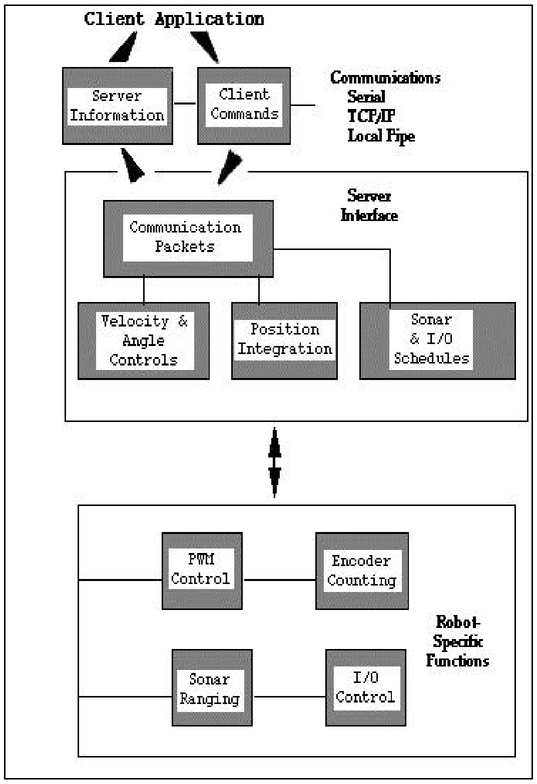


Figure 6-1. The architecture of Saphira's client-robot server

Communication Packet Protocol

P2OS communicates with a client application using a special packet protocols: command packets from client to server, and server information packets from the server to client. Both are bit streams consisting of four main elements (Table 6-1): a two-byte header, a one-byte count of the number of command/data bytes, the client command and its arguments or the server information data, and finally, 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, including checksum word, but not Byte Count. Maximum 200 total bytes.
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 below 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. P2OS Communication Packet Data Types

Data Type	Bytes	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

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 XORed 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, P2OS ignores a client command packet whose byte count exceeds 200 or has an erroneous checksum. The client should similarly ignore erroneous server information packets.

P2OS does not acknowledge receipt of a command packet nor does it have any facility to handle client acknowledgment of a server information packet. Consequently, Pioneer client/server communications are as reliable as the physical communication link. A cable tether between the robot and client computer, such as a piggyback laptop, provides 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 can you expect the server to accept every command.

Because of the real-time nature of the client / server interaction, we made a conscious decision to provide an unacknowledged packet interface. Re-transmitting server-information blocks or command packets would serve no useful purpose, because old data would be virtually useless in maintaining responsive client-server interaction.

For some operations, however, the data do not decay as rapidly: Some commands are not overly time-sensitive – for example, those that perform such housekeeping functions as 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 the 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

P2OS implements a structured command format for receiving and responding to directions from a client for control and operation of the robot or its simulator. The number of client commands per second you may send depends on the serial baud rate and average number of data bytes per command. Do note that the P2OS server may not be up to the task of managing a deluge of commands, so we recommend that you time commands with the internal processing clock of the server, settable to either 100 milliseconds or 50 milliseconds per cycle with Pioneer 2. You may exceed those rates; just do it sparingly.

The client must send a command at least once every two seconds or so (`watchdog` parameter; see *Updating & Reconfiguring P2OS*, Chapter 7); otherwise, the communication watchdog server will stop the robot's onboard drives.

Table 6-3. P2OS client command packet

Component	Bytes	Value	Description
Header	2	0xFA, 0xFB	Packet header; same for client and server
Byte Count	1	N + 2	Number of subsequent command bytes plus checksum, not including Byte Count. Maximum of 200 bytes.
Command Number	1	0 - 255	Client command number; see Table 4-4
Argument Type (command dependent)	1	0x3B or 0x1B or 0x2B	Required data type of command argument: positive integer (`sfARGINT`), negative integer or absolute value (`sfARGNINT`), string, null-terminated (`sfARGSTR`)
Argument (command dependent)	n	data	Command argument; integer or string
Checksum	2	computed	Packet integrity checksum

The P2OS 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 value.

Client Command Argument Types

There are three different types of P2OS client command arguments: positive integers two bytes long, negative integers two bytes long, and NULL-terminated strings consisting of as many as 196 characters. The 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 an integer, a string, or nothing, depending on the command.

Table 6-4. P2OS/PSOS command set

Command	#	Args	Description	PSOS	P2OS
			Before Client Connection		
SYNC0	0	none	Start connection; P2OS echoes synchronization commands back to client.	3.x	1.0
SYNC1	1	none			
SYNC2	2	none			
			After Established Connection		
PULSE	0	none	Client pulse resets server watchdog	3.x	1.0
OPEN	1	none	Starts the controller	3.x	1.0
CLOSE	2	none	Close server and client connection	3.x	1.0
POLLING	3	int, string	Set sonar polling sequence; *int* is number of string bytes	3.9	1.0
ENABLE	4	int	Enables/disables the motors	--	1.0
SETA	5	signed int	Resets translational acceleration parameter, if positive, or deceleration, if negative; in millimeters per second^2	--	1.0
SETV	6	int	Reset maximum translational velocity, in millimeters per second	4.8	1.0
SETO	7	none	Resets server to 0,0,0 origin	3.x	1.0
SETRV	10	int	Resets maximum rotational velocity in degrees per second	4.8	1.0
VEL	11	int	Move forward (+) or reverse (-) at millimeters per second	3.x	1.0
HEAD	12	int	Turn to absolute heading; 0-359 degrees	4.2	1.0
DHEAD	13	int	Turn relative to current heading; +- degrees	3.x	1.0
SAY	15	int, string	As many as 20 pairs of duration (20 ms increments) /tone (half-cycle) pairs; *int* is string length	4.2	1.0
RVEL	21	signed int	Rotate at +- degrees per second	4.2	1.0
DCHEAD	22	int	Colbert relative heading setpoint; +- degrees		
SETRA	23	int	Sets rotational (+)acceleration or (-)deceleration, in millimeters per second^2	--	1.0
SONAR	28	int	Enable/disable the sonar	--	1.0
STOP	29	none	Stops robot (motors remain enabled)	--	1.0
VEL2	32	int	Independent wheel velocities; lsb=right wheel; msb=left wheel; PSOS is in +-4mm/sec; P2OS in 2 cm/sec increments	4.1	1.0
STEP	64	none	Single-step mode (simulator only)	3.x	1.0

Saphira Client Command Support

Saphira fully supports P2OS client commands with useful library functions. You can find prototypes in `$(SAPHIRA)/handler/include/saphira.h` and `saphira.pro`. Saphira's P2OS command names have the prefix `sfCOM` followed by the command names listed in subsequent tables. (See the *Saphira Software Manual* for details.)

For example, to enable the motors from the Colbert interaction window on the Saphira client, type:

```
sfRobotComInt(sfCOMENABLE,1);
```

Or to have it play a tune (albeit rather tinny), type:

```
sfRobotComStrn(sfCOMSAY,"\1\6\2\105",4);
```

Server Information Packets

Like its PSOS predecessor, P2OS automatically sends a packet of information over the serial communication port back to the client every 100 milliseconds. The P2OS 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.

Unlike PSOS, however, P2OS supports several additional server information packet types, extending the capabilities of the Pioneer 2. See the following section for a description of the extended server information packet types available with P2OS version 1.0. And, a stored parameter lets you increase the communications rate to 20 per second (50ms cycle). (See Chapter 7, *Updating & Reconfiguring P2OS*.)

Table 6-5. Standard P2OS server information packet

Name	Data Type	Description
`Header`	integer	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 (0)`	Motors power off
	`sfSTATUSSTOPPED (1)`	Motors stopped
	`sfSTATUSMOVING (2)`	Robot moving
`Xpos`	unsigned integer (15 ls-bits)	Wheel-encoder integrated coordinates; platform-dependent units; multiply by *DistConvFactor* in the parameter file to convert to millimeters
`Ypos`	unsigned integer (15 ls-bits)	
`Th pos`	signed integer	Orientation in platform-dependent units—multiply by `AngleConvFactor` for degrees.
`L vel`	signed integer	Wheel velocities (respective Left and Right) in platform-dependent units—Multiply by `VelConvFactor` to convert to millimeters per second
`R vel`	signed integer	
`Battery`	byte	Battery charge in tenths of volts
`Bumpers`	integer	Motor stall indicators; left = msb
`Control`	signed integer	Setpoint of the server's angular position servo—multiply by `AngleConvFactor` for degrees
`PTU`	unsigned integer	Pulse width of position servo
`Compass`	byte	Compass heading in 2-degree units
`Sonar readings`	byte	Number of new sonar readings included in information packet; readings follow:
`Sonar number`	byte	Sonar number
`Sonar range`	unsigned integer	Sonar reading—multiply by `RangeConvFactor` for mm
…rest of the sonar readings…		
`Checksum`	integer	Checksum (see previous section)

`*Factors* refer to values in Saphira parameter file. See Appendix C.`

To be fully compatible with the original PSOS and supporting clients like Saphira, the standard P2OS server information packet will not change. Newer versions of the P2OS server information packet will be implemented as a different packet type.

P2OS Extended Packets

To be implemented…

Programming P2OS

As mentioned above, you may create your own P2OS interface, or use the convenience functions available through the various applications development software that comes with your Pioneer 2.

Synchronization—SYNC

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 operating information from the server.

When first started, the Pioneer 2 robot is in a "wait" state; P2OS listens for communication packets over its designated port. To establish a connection, the client application must send a series of three synchronization packets through the host communication port – `SYNC0, SYNC1, and SYNC2,` in succession, and retrieve the server responses.

P2OS 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

P2OS automatically sends configuration information back to the client in the last sync packet (`SYNC2`). 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 2 with a special configuration tool we provide with the robot. The class and subclass also are parameters stored in the robot's flash PROM, but are constants not changed by the user. (See Chapter 7, *Updating and Reconfiguring P2OS* for details.)

The Pioneer class string is simply `Pioneer`; the subclass depends on your robot model; P2DX or P2AT, for example. Clients may use these identifying parameters to self-configure their own operating parameters. The Saphira client software, for instance, loads a special, related Pioneer parameters file found in the Saphira `params` directory.

Opening the Servers—`OPEN`

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

Note that once connected, Pioneer 2's motors are disabled, regardless of their state when last connected. To enable the motors after starting a connection, you must either enable the motors manually (white MOTORS button) or send an ENABLE command with the integer argument 1.

Keeping the Beat—`PULSE`

As mentioned earlier, a P2OS safety watchdog expects the Pioneer server to receive at least one communication packet from the client every few seconds. Otherwise, it assumes the client/server connection is broken and shuts down the robot's motors.

It's good practice to send a `PULSE` to Pioneer just after opening the servers. And if your client application will be otherwise distracted for some time, periodically issue the `PULSE` 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—`CLOSE`

To close a connection, disable the motors and sonar, and reset P2OS to the wait state, simply issue the client `CLOSE` command.

Movement Commands

The P2OS server accepts several different motion commands (Table 6-6) of two mutually exclusive types: either direct wheel-velocity control or P2OS motor controls. The robot server automatically abandons any P2OS translational or rotational setpoints and switches to direct wheel-velocity control mode when it receives a SETVEL2 command. Any other motion command makes P2OS abandon direct wheel-velocity control. For example, if the P2OS is in two-wheel velocity mode and is given a `HEAD` command, it disables two-wheel velocity mode and starts controlling the heading and velocity of the robot.

When in P2OS motion control (recommended), translation and rotation operate independently. P2OS will try to make the robot achieve the desired velocity and heading as soon as the commands are received, using its internal acceleration and deceleration managers, which default values you may set (see Chapter 7, *Updating and ReconfiguringP2OS*) and change on the fly (SETRA and SETA).

Table 6-6. P2OS motion commands

Rotation	
`HEAD`	Absolute heading
`DHEAD, DCHEAD`	Differential heading from control point
`SETRA`	Rotational (de)acceleration to achieve setpoint
`SETRV`	Sets rotational velocity for Colbert turn and turnto commands
Translation	
`VEL`	Forward/back velocity
`SETA`	Translational (de)acceleration to achieve setpoint
`SETV`	Velocity for Colbert move command

Pioneer in Motion

When P2OS receives a P2OS server-translation or -rotation command, it accelerates the Pioneer robot at the SETA or SETRA rate you program, or the rates preset in the P2OS configuration parameters. 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.

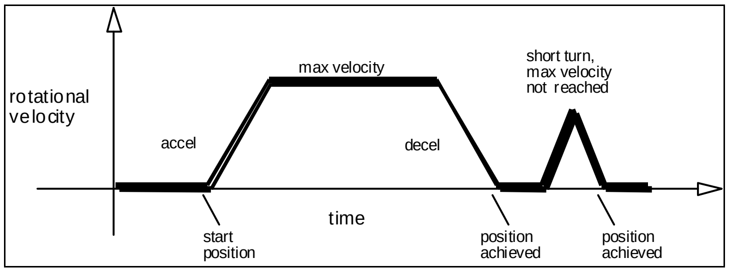


Figure 6-2. Pioneer’s trapezoidal turning velocity profile

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 meters 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 HEAD and DHEAD turn the robot with respect to its internal dead-reckoned angle (Figure 6-3). On start-up, the robot is at the origin (0,0), pointing toward 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 robot 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).

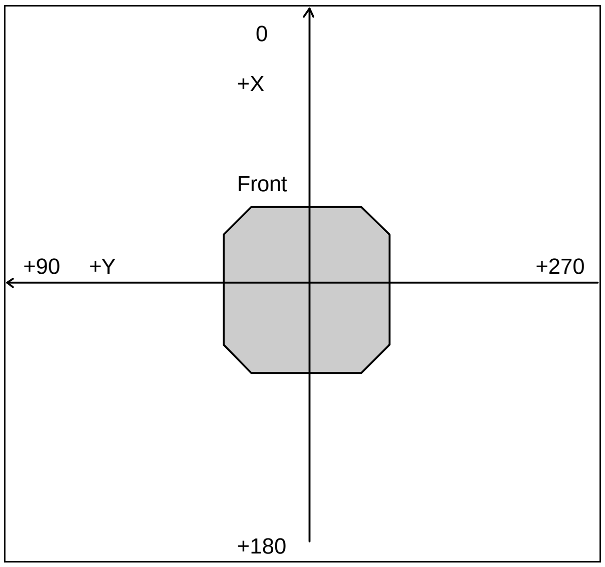


Figure 6-3. Internal coordinate system for P2OS

You may reset the internal coordinates to 0,0,0 with the SETO P2OS command.

Sonar

When opened by the appropriate client command (see "Opening the Servers," above), P2OS automatically begins firing Pioneer's sonar in the predefined default sequence: 1–8 and 9-16, simultaneously when both front and rear sonar are implemented. P2OS also begins sending the sonar ranging results to the client via the server-information packet. Use the SONARS command to enable (argument is "1") or disable (argument is "0") the sonar pinging.
For example:

```
  sfRobotComInt(sfCOMSONARS,0);  /* Stop the pinging when not needed */
```

Use the `POLLING` command to change the polling sequence of the sonar. Its argument is a null-terminated string of bytes (octal notation). Front sonar are numbered (octal) 1 through 10, and the rear sonar are numbers 11 through 100. You may repeat a sonar number and have it ping more than one per sequence.
For example:

```
  sfRobotComStrn(sfCOMPOLLING,"\1\2\1\2",4); /* ping only on the left side */
```

Note that if the string is empty, all sonar get turned off, even though you haven't disabled the sonar with the SONARS command.

Input / Output (I/O)

Your Pioneer 2 comes with a number of I/O ports that you may use for some of Pioneer's accessories and for your own custom attachments. None are yet enabled in P2OS. Soon, though. We promise.

See Appendix A for port locations on the Pioneer microcontroller.

Updating & Reconfiguring P2OS

The P2OS server software and its set of operating parameters get stored on the Pioneer 2 microcontroller's flash ROM. With special download and configuration utilities, you may change and update the flash ROM image without physically replacing any hardware.

Where to Get P2OS Software

Your P2OS comes preinstalled with the latest version of P2OS. Thereafter, stay tuned to the pioneer-users newsgroup, or periodically visit our support website to obtain the latest P2OS and related documentation:

`http://robots.activmedia.com`

Installing the P2OS Utilities

The P2OS utilities come with your robot or can be downloaded from the support website as a separate package. Install the version that matches your client computer’s environment. For example, use `linux-p2os1_0.tgz` for RedHat v4.x or 5.x Linux, `sun-p2os1_0.tgz` for SunOS, or `win32-p2os1_0.exe` with Microsoft Windows 95/NT/98. 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-p2os1_0.exe` distribution is a self-extracting, self-installing package: Simply follow the on-screen directions. For the Linux/UNIX versions, uncompress and untar them using the appropriate system utilities. For example, with the Linux version, the proper command is:

```
% tar –zxvf linux-p2os1_0.tgz
```

The command creates a `p2os/` directory in the current path and stores the P2OS software there.

Updating P2OS

Use the `p2osdl(.exe)` program to download a fresh copy of PSOS to the Pioneer microcontroller’s flash ROM.

Step 1. Serial Connection from Computer to Pioneer

Connect Pioneer 2 to your host computer through their respective serial ports. We recommend a direct tether from the computer to the 9-pin serial port on top of the Console, because 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; neither a tether to the serial port on the back panel 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; if you do not, they will interfere with the direct connection.

Step 2: Enable FLASH

Locate the FLASH switch on the Console. It's recessed. Use a flat-bladed screwdriver or other thin instrument to move the slide switch forward toward the front of the robot to enable FLASH writes.

Step 3: Put Pioneer into Boot Mode

Start up or reset your Pioneer 2. After it has finished initializing, place it in its special boot mode: While pressing and holding the white MOTORS button, press and release the red RESET button. Continue holding the MOTORS button down for three or more seconds, then release it.

The robot should not reset; if it does, you probably didn’t hold the MOTORS button down long enough. Try it again. And you'll notice that the "heartbeat" asterisk stops blinking in the LCD.

Step 4: Run p2osdl

With Linux/UNIX systems, enter the `p2os/` directory and execute `p2osdl`, using the following arguments:

```
% p2osdl p2os.hex <comm-port>
```

The pathname for the `p2os.hex` image file is required and may be a direct or relative path. The file `p2os1_0.hex`, for example, is the current P2OS version 1.0 image in the `p2os/` folder.

The `comm-port` argument is optional. It lets you specify the serial communication port that connects `p2osdl` with the Pioneer microcontroller. For Linux/UNIX systems, the default is `/dev/cua0`.
To communicate through /dev/cua3, for example, use:

```
% p2osdl p2os1_0.hex /dev/cua3
```

For 32-bit Windows systems, you must pass the required `p2os.hex` file to `p2osdl.exe`, so you cannot simply double-click on the program icon from the Windows desktop. Instead, use your mouse to successfully launch the program by “dragging and dropping” the `p2os.hex` icon (`p2os1_0.hex`, for example) onto the `psosdl.exe` icon. But this works only if you also plan to use the default `COM1` serial port. The universal solution is to execute `p2osdl.exe` and pass it the proper arguments from the MS-DOS Prompt program, which normally resides in the Programs section of the Start menu.

Note that p2osdl and p2oscf (below) both autoconfigure the serial ports to communicate at 19,200 bps, regardless of other settings.

Download Troubleshooting

The p2osdl program will tell you if the download was successful or not. If it was, simply reboot the robot and go on to connect with a client or change its operating parameters (see next section). Otherwise, try the download from Step 1 again, to be sure you performed each step correctly.

Configuring P2OS Operating Parameters

The program `p2oscf` (`p2ocf.exe` for 32-bit Windows) is the way you view and change your Pioneer 2’s identity and operating parameters.

Limited reconfigurations.
FLASH PROM in the Pioneer 2 is guaranteed for only 100 erase cycles.

Steps 1–3: Preparing Pioneer 2 for Configuration

Prepare your connection and Pioneer 2 for changes to its operating configuration identically to Steps 1, 2 and 3 for updating P2OS.

Step 4: Run p2oscf

As with `p2osdl`, you will find p2oscf (.exe) in the `p2os/` directory of your PSOS distribution.

```
% p2oscf <comm-port>
```

The program accepts a single, optional parameter – the serial communication port name (`comm-port`), whose specification is identical to that for `p2osdl`. The port `/dev/cua0` (or `/dev/ttyS0`) in Linux/UNIX systems and `COM1` are the default serial ports `p2oscf` and `p2osdl` use, if none is given from the command line. Accordingly, if you use `COM1` on your Win32 PC, you may launch `psoscf.exe` directly from the desktop, rather than executing the program from the MS-DOS Prompt window.

Step 5: Changing the Configuration Parameters

On startup (after power cycle or RESET), P2OS reads a set of operating parameters from its flash PROM and uses these values if and until you override them with explicit P2OS commands. For instance, a default maximum velocity is stored in flash PROM (TransVelMax) which value is used by P2OS when receiving a

Colbert move command or other “goto position” client command (robot accelerates to that TransVelMax and runs at that velocity until it nears and then reaches its destination).

When it starts properly, `p2oscf` retrieves from the microcontroller the current identifying and operating parameters that P2OS uses for your Pioneer 2. Some of the parameters, "Constants" in Table 7-1, currently cannot be changed. The others, "Variables" in Table 7-1, are the identifying and operating parameters that you may edit with p2oscf.

The list of P2OS parameters with descriptions and default values is in the Table 7-1. The list of p2oscf commands is in the Table 7-2. With p2oscf, you edit a temporary copy of the parameters list. Your changes are not written to the flash ROM until you choose to explicitly "save" them. Even then, p2oscf will write to flash ROM only if you have changed some variable parameter. Writes to the flash ROM are guaranteed for only 100 cycles, so we caution that you reconfigure/update Pioneer 2 only when necessary.

Editing P2OS Parameters

To view the list of current P2OS constants or variables, type '**c**' or '**v**', respectively, followed by a return (Enter). Similarly, type '**?**' or '**help**' to see a list of p2oscf commands.

To see a parameter's current value individually, type its keyword alone. To change a P2OS variable parameter's value, type its keyword followed by the replacement value. That value may be a string (no quotes) or a decimal or hexadecimal ("0xN") number. For example, to change the watchdog timeout to last for four seconds, type:

```
        > watchdog 4000
or:
        > watchdog 0xfa0
```

The critical operating parameters for Pioneer 2 are `revcount`, and `the` PID control parameters. If you get them wrong, your robot won’t run properly. Note, too, that your p2oscf-edited parameters *are not* used by P2OS unless and until you '**save**' them to the flash ROM. And, too, you may over-ride many of these parameters with respective P2OS commands from the client.

Table 7-1. Pioneer 2 DX configuration parameters `(P2OS 1.0)`

KEYWORD	**Default**	**Description**
CONSTANTS		*Cannot be changed using p2oscf*
Type	Pioneer	String identifies the robot as a Pioneer type and is included in the SYNC2 connection return packet along with Subtype and Name.
Subtype	P2DX	Identifies the Pioneer model; P2AT or P2CE for other models.
Serial	factory set	Serial number for the Pioneer 2.
FourMotors	0	Is '1' for the 4-motor Pioneer 2-AT
RotVelTop	360	Maximum allowable rotational velocity in degrees per second
TransVelTop	2200	Maximum allowable speed in millimeters per second
RotAccTop	720	Maximum allowable rotational (de)acceleration in degrees per second
TransAccTop	4000	Maximum allowable translational (de)acceleration; millimeters per second
PwmMax	500	Maximum motor pulse period (500=fully on).
VARIABLES		*Parameters that you may change with p2oscf*
Name	not_set	Unique name you may give your Pioneer 2. Besides its ownership value, this parameter gets passed to a connecting client as the first argument in the SYNC2 packet, therefore useful for differentiating among multiple Pioneers. Maximum of 20 characters; no intervening spaces.
SInfoCycle	0	Server information packet communication cycle time: 0=classic 100 milliseconds, as for PSOS; 1=new 50ms cycle time
HostBaud	0	Baud rate for host (client) serial port connection. 0=9600, 1=19200, 2=38400 bps.
AuxBaud	2	Baud rate for second (AUX) serial port
RearSonar	0	'1' if rear sonar array installed.
LowBattery	110	In 1/10 volts; microcontroller alarm activated when battery charge falls below this value.
RevCount	16250	The number of encoder ticks for a 360 degree revolution of the robot. Reset this parameter to a number that best reflects the characteristics of your robot in a particular environment.

`WatchDog`	2000	Milliseconds time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection with client.
`P2Mpacs`	0	'1' enables new, extended P2OS server information packet.
`StallVal`	200	Maximum PWM before stall; either or both motors. If > PwmMax, never stalls.
`StallCount`	100	Milliseconds after a stall for recovery. Motors not engaged during this time.
`CompX`	0	Compass calibration X-offset
`CompY`	0	Compass calibration Y-offset
`RotVelMax`	50	Maximum velocity for completion of a Colbert or similar rotation.
`TransVelMax`	300	Maximum velocity for completion of a Colbert or similar translation.
`RotAcc`	50	Rotational acceleration in degrees per second
`RotDecel`	50	Rotational deceleration in degrees per second
`RotKp`	30	Proportional PID system control parameter controls the basic responsiveness of the drive system. Lower values give a slower, less-responsive system; higher values make the robot "zippier", but can lead to overshoot and oscillation.
`RotKv`	60	Differential control parameter dampens oscillation and overshoot. Increasing values gives better control oscillation and overshoot, but they also make the robot's movements more sluggish.
`RotKi`	0	Integral control parameter adjusts residual error in turning and velocity. Higher values make the robot correct increasingly smaller errors between its desired and actual angular position and speed.
`TransAcc`	300	Translational acceleration in degrees per second
`TransDecel`	300	Translational deceleration in degrees per second
`TransKp`	30	Proportional PID system control parameter controls the basic responsiveness of the drive system. Lower values give a slower, less-responsive system; higher values make the robot "zippier", but can lead to overshoot and oscillation.
`TransKv`	60	Differential control parameter dampens oscillation and overshoot. Increasing values gives better control oscillation and overshoot, but they also make the robot's movements more sluggish.
`TransKi`	0	Integral control parameter adjusts residual error in turning and velocity. Higher values make the robot correct increasingly smaller errors between its desired and actual angular position and speed.

Table 7-2. p2oscf control commands

Command	Description
`keyword` `<value>`	Alone, keyword displays current, edited value. Add value argument to change current value.
`c` `or` **`constants`**	Display P2OS constant values. User cannot change.
`v` `or` **`variables`**	Display current, edited P2OS operational values; may be different than values currently stored in flash ROM.
`r` `or` **`restore`** `<pathname>`	Restores edited (p2oscf) variables to values currently stored in flash ROM or from file, if argument included.
`save` `<pathname>`	Saves current edited values to flash ROM and exits program *OR* saves current edited values to disk for later reference and continues in editor.
`q` `or` **`quit`**	Exits p2oscf without saving any changes to flash.
`?` `or` **`help`**	Displays commands and descriptions.

Stand-alone Mode

As mentioned in previous chapters, the Pioneer 2 microcontroller's flash-PROM is reprogrammable. And there is 256K RAM onboard waiting for something to do. Stand-alone mode also lets you perform experiments in low-level sensing and control, such as feedback control of the motors.

*Activ*MEDIA is developing Pioneer 2 stand-alone mode support kits and materials. Stay tuned to the pioneer-users newsgroups for availability.

Maintenance & Repair

Your Pioneer 2 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 won't hurt. And keep the axles clear of carpet or other strings that may wrap around and bind up Pioneer's drive.

Pioneer Batteries

Lead-acid batteries like those in Pioneer 2 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 below 11 VDC.

Changing Batteries (DX and AT)

The Pioneer 2 DX and AT models have a special battery harness and latched doors for easy access to the onboard batteries. Simply unlatch the rear door of the DX, for instance, swing it open and locate the one to three onboard batteries inside.

To remove a battery, simply grasp it and pull out. We also provide a suction-cup tool to help. Spring-loaded contacts eliminate the need to detach any connecting wires.

Similarly, insert batteries by simply sliding one into place. Load the batteries so that their weight gets distributed evenly across the platform: Center a single battery and place two batteries one on each side.

Hot-Swapping the Batteries (DX and AT)

You may change the batteries on a Pioneer 2 DX or AT without disrupting operation of the onboard systems (except the motors, of course): Either connect the charger, which powers the robot's systems while you change the battery or batteries. Or, if you have two or three batteries, swap each with a freshly charged one individually, so that at least one battery is in place and providing the necessary power.

Charging the Battery

Insert the battery charger that comes with your Pioneer 2 into a standard 110 VAC three-pronged wall power receptacle. (Some users may require a special power adapter.) Locate the round plug at the end of the cable that is attached to the charger and insert it into the charge socket that is just below Pioneer 2's Main Power switch. The LEDs on the charger indicate charge status, as marked on its case.

It takes fewer than 12 hours – often just a few hours, depending on the level of discharge – to fully charge a Pioneer battery using the accompanying charger (roughly, three hours per volt). Although you may operate the robot while recharging, it restricts the robot's mobility. 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 2 is the positive (+) side of the battery; the case is the negative (-) side. If you choose to use an alternative battery charger for Pioneer, 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 at least 0.75A at 13.75 to 14 VDC. It also should be voltage- and current-limited so that it cannot overcharge the battery.

Factory Repairs

If, after reading this manual, you're having *hardware* problems with your Pioneer, and you're satisfied that it needs repair, contact us:

```
    http://www.pioneer-support@activmedia.com
          (603) 924-2184 fax
        (603) 924-9100 (voice)
```

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

We will try to 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.

Appendix A

C166 Controller Ports & Connections

This Appendix contains pinout and electrical specifications for the external and internal ports and connectors on the Pioneer 2 microcontroller board. These include an external serial port for P2OS/client connections, two internal serial ports for host and accessory communications, each with switched five and 12 VDC power, an expansion bus, and a discrete user-I/O connector.

Note that all the internal connectors (IDC sockets) are numbered odd pins on top and even pins on the bottom, from right to left, relative to the key (Figure A-1):

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

Figure A-1. Common IDC connector pinouts

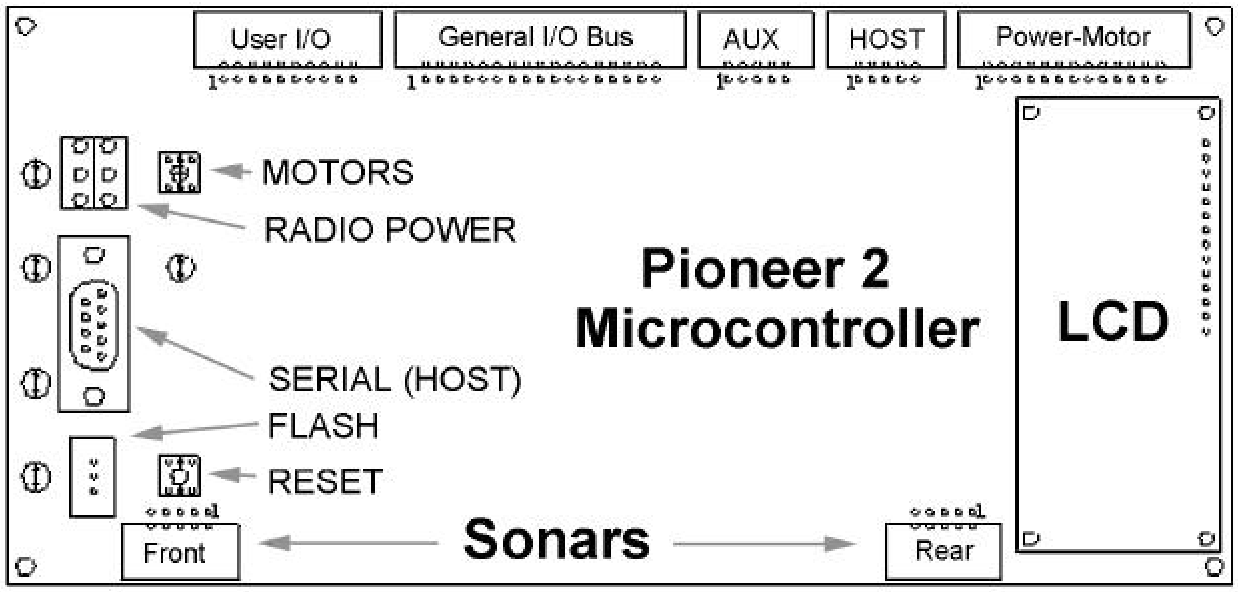


Figure A-2. Pioneer 2 microcontroller ports and connectors

Console Serial Port

Use the Console DSUB-9 serial port connector for client connections to the Pioneer 2 servers and for P2OS downloads. Only three pins are active: RS232-compatible pins 2 (TxD1), 3 (RxD1), and 5 (signal ground).

Table A-1 lists the cable connections for various host computers to this external serial port. Most direct serial connections use a simple, straight-through cable, such as the one for a common PC. The communication lines are shared with those on the internal P2OS serial port (see below).

Table A-1. Common serial cable connections to Pioneer 2

Platform & Connector	Pin 3 (Tx)	Pin 5 (Rx)	Pin 9 (Gnd)
Sun Sparcstation DB-25	3	2	7
SGI Irix mini-DIN 8	5	3	4
PC COM*n*: DB-9	2	3	5
PC COM*n*: DB-25	3	2	7
Macintosh mini-DIN 8	3	5	4

The Pioneer 2 operates at the following configuration: 9,600, 19,200 or 38,400 bits per second data rate; eight data bits; one stop bit; no parity or hardware handshaking (DTR).

Internal Serial Connectors

Two 10-pin serial port connectors on the Pioneer 2 microcontroller support independent RS-232 serial communications. The ports also contain switched (RADIO on the Console) five and 12 volts DC power (maximum available 0.75A or 2.25A, depending on Pioneer 2 configuration) for accessory connections, such as a radio modem, radio Ethernet, and the Pioneer electronic compass (Table A-2).

The “host” serial port (connector JP8) is the serial port used by the Pioneer 2 microcontroller for client communications. It shares its transmit (TxD1) and receive (RxD1) lines with the external serial port on the Console. The P2OS connector also has three data lines (P3_12, _14, and _15) that are used by the Pioneer compass.

Table A-2. P2OS Internal Serial Port Connections ("host" JP8)

Pin #	Connection		Pin #	Connection
1	Gnd		2	P3_12
3	TxD1		4	12 VDC (switched)
5	RxD1		6	Gnd
7	P3_15		8	P3_14
9	Gnd		10	5 VDC (switched)

The auxiliary RS232-compatible serial port (connector JP7) on the Pioneer 2 microcontroller operates independently from the host serial port and is for accessory attachments, such as the PTZ Robotic Camera. Note the additional power connections.

Table A-3. Auxiliary Internal Serial Port Connections ("aux" JP7)

Pin #	Connection		Pin #	Connection
1	Gnd		2	12 VDC (switched)
3	TxD2		4	12 VDC (switched)
5	RxD2		6	Gnd
7	No connection		8	5 VDC (switched)
9	Gnd		10	5 VDC (switched)

User I/O Expansion Port

A 20-pin IDC socket on the Pioneer 2 microcontroller provides these digital, analog, and power ports for user connections (Table A-4):

- 8 general-purpose digital I/O (P3_0-7)
- 5 analog-to-digital input (A/D) (P5_4-7,9)
- 4 capture/compare (P2_12-15)
- 1 signal ground (Gnd)
- 1 Vcc (+5 VDC)
- 1 Vpp (+12 VDC)

Note that the general-purpose I/O and analog-to-digital ports are shared with the onboard bus (below) and four-motors (Appendix B) circuitry, respectively, and may not be available for use on all robots. In particular, lines P2_3-6 are for controlling the additional two motors in the Pioneer AT. The analog lines P5_4-7, although not used by any Pioneer 2 configuration, are connected to and may be used for additional encoders in four-wheel drive Pioneer 2s. The digital I/O lines P3_0-7 are included in the I/O bus.

Table A-4. User I/O Expansion Port

Pin #	Label		Pin #	Label
1	P2_12		2	P3_0
3	P2_13		4	P3_1
5	P2_14		6	P3_2
7	P2_15		8	P3_3
9	P5_4		10	P3_4
11	P5_5		12	P3_5
13	P5_6		14	P3_6
15	P5_7		16	P3_7
17	P5_9		18	Vcc
19	Vpp		20	Gnd

The General I/O Bus

The 34-pin IDC socket on the Pioneer 2 provides a general-purpose I/O bus (Table A-5), containing:

- 8 read/write data lines (D0-7)
- 4 chip select lines (CS_2-5)
- 2 address lines (A0, A1)
- Read (RD#) and write (WR) lines
- 8 general-purpose digital I/O (P3_0-7)
- 5 analog-to-digital input (A/D) (P5_4-7,9)
- 2 signal ground (Gnd)
- 2 Vcc (+5 VDC)
- 1 Vpp (+12 VDC)

Table A-5. General I/O Bus Connections

Pin #	Label		Pin #	Label
1	P3_0		2	D0
3	P3_1		4	D1
5	P3_2		6	D2
7	P3_3		8	D3
9	P3_4		10	D4
11	P3_5		12	D5
13	P3_6		14	D6
15	P3_7		16	D7
17	WR		18	CS2
19	Vpp		20	CS3
21	P2_12		22	CS4
23	P2_13		24	CS5
25	P2_14		26	A0
27	P2_15		28	A1
29	P5_9		30	Vcc
31	Gnd		32	RD#

33	Gnd		34	Vcc

Appendix B
Motor-Power Board Connectors

The Pioneer 2 has a separate Motor-Power Board which, in conjunction with the microcontroller, provides power for the two and four motors of the Pioneer 2 DX, CE and the Pioneer 2 AT models, respectively, as well as conditioned power for the standard and accessory onboard electronics (Figure B-1). It also contains buffered pass-through connectors for the motor encoders. An optional power package ("computer ready") is available for an onboard computer system and Ethernet radio.

Find the Motor-Power Board attached to the back of the battery box.

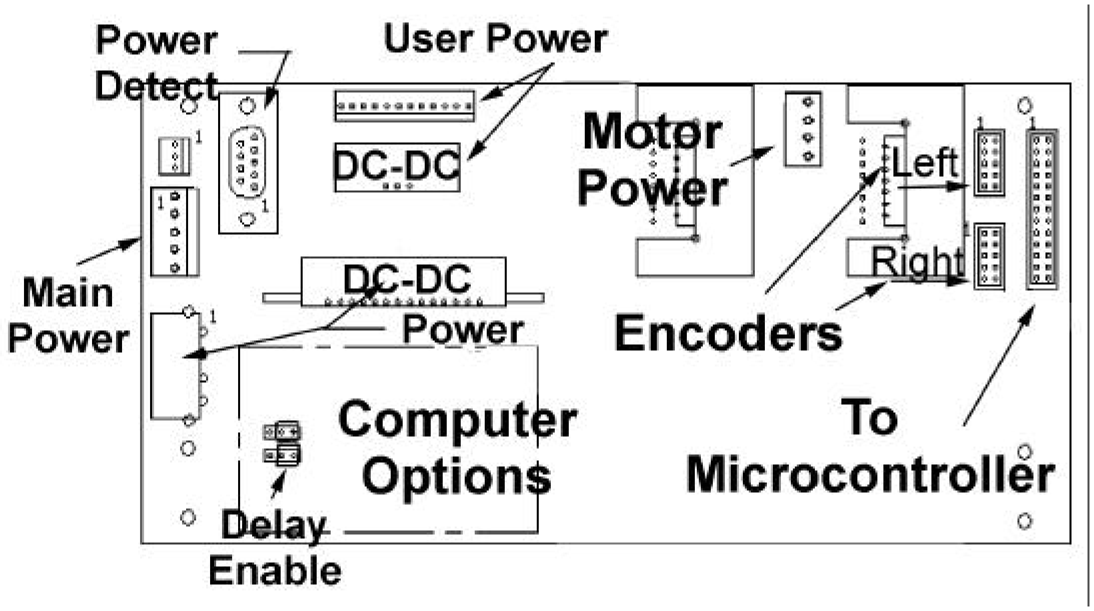


Figure B-1. Pioneer 2 Motor-Power Board

User Power Connections

On the Pioneer 2 Motor-Power board, four sets of un-switched battery (Vpp) and DC-DC-conditioned five volts (Vcc) power, with power ground (Gnd), are available for accessory and custom electronics on a 12-pin Molex-type latch-lock (100mm) connector (Table B-1). The Vcc power gets shared with the micro-controller and accessories connected to that board (Appendix A). The maximum Vcc power available with standard DX, CE and AT models is 1.5A; 3A with computer-ready option.

Table B-1. User Power (J1)

Pin #	Connection		Pin #	Connection
1	Vcc		7	Vcc
2	Gnd		8	Gnd
3	Vpp		9	Vpp
4	Vcc		10	Vcc
5	Gnd		11	Gnd
6	Vpp		12	Vpp

Onboard Computer Option

With the "computer-ready" option (Figure B-2) available with the DX and AT models, the Pioneer 2 Motor-Power Board is specially populated with connectors and circuits to provide separate DC-DC conditioned, 7A @ 5 VDC power to an onboard computer and its accessories, and enhanced (3A instead of the standard 1.5A) DC-DC conditioned 5VDC power to the other onboard systems and accessories, including radio Ethernet. And, in conjunction with software, the circuit provides for automatic, unattended, crash-less shutdown of the onboard computer, either when manually power-down or when the battery voltage drops below 10 VDC.

Power Switch (J7) and Delayed Shutdown Logic

A simple SPDT switch controls logical power to the computer-power section of the Motor-Power Board. When on, the switch provides Vpp to the input of a voltage comparator which output controls the computer's DC-DC voltage converter. Designed to provide for "crashless" shutdown of the computer in the event of a power brownout or for unattended shutdowns, if Vpp drops below 10 VDC, the comparator automatically initiates a two-minute delay (LMC55 timer-mediated) before computer-power (not system-wide) shutdown.

Similarly, when the computer power is manually switched off, the comparator is grounded (below 10 VDC, of course) and the same two-minute delayed shutdown occurs.

In all cases, power shutdown, but not necessarily computer software shutdown, is canceled if power is restored to above 10 VDC, either by connecting the Battery Charger or by hot-swap refreshing the batteries.

Also, you may disable the two-minute delay circuitry (switched or low-voltage shutdown immediate) by moving the jumpers on both J3 and J6 on the Motor-Power Board to the pins opposite their ON position.

Table B-2. Computer Option Power Switch (J7)

Pin #	Connection
1	Vcc
2	Gnd
3	Vpp

Power-State Logic

A 9-pin DSUB receptacle (P1) on the Pioneer 2 Motor-Power Board provides a logic-level shutdown indicator (DCD line pin 1) which state goes low when computer power shutdown is first initiated. The onboard computer may use this line state to initiate software-mediated shutdown, such as provided by *genpowerd* for RedHat Linux PCs.

Table B-3. Power State 9-pin DSUB receptacle (P1)

1	2	3	4	5	6	7	8	9
DCD 100K pull-up pins 4,6	**nc**	**nc**	**pin 6**	**nc**	**pin 4**	**nc**	**nc**	**Gnd**

Computer Power

A common 4-conductor, bevel-keyed connector (J5) provides 5 VDC @ 7A power for the onboard computer, and battery power for the optional 12VDC fan.

Table B-4. Computer power and fan connector (J5)

Pin #	Connection
1	Vpp
2	Gnd
3	Gnd
4	5VDC @ 7A

Appendix C
Saphira Parameter Files

Saphira consults a special parameters file to determine various operating parameters for the robot server to which it connects. All the latest Pioneer parameter files can be found in `$(SAPHIRA)/params`.

P2DX.p

```
;; Saphira parameters describe the Pioneer 2 DX
;;
Class                  Pioneer
Subclass               p2dx
;;
AngleConvFactor        0.001534   ; radians per encoder count diff (2PI/1024)
DistConvFactor         0.0785     ; 5in*PI / 5000 (mm/count)
VelConvFactor          1.0         ; mm/sec per encoder count per 1/50 sec
RobotRadius             250.0      ; radius in mm
RobotDiagonal            120.0     ; half-height to diagonal of octagon
Holonomic              1           ; turns in own radius

;; These are for sixteen sonars: six each front and rear, two each side
;;
;; Sonar parameters
;;              SonarNum N is number of sonar
;;              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        0.268     ; sonar range mm per 800 nsec tick
;;
SonarNum 16
;;        #   x   y   th
;;-------------------------
SonarUnit  0  115  130  90
SonarUnit  1  155  115  50
SonarUnit  2  190  80  30
SonarUnit  3  210  25  10
SonarUnit  4  210  -25  -10
SonarUnit  5  190  -80  -30
SonarUnit  6  155  -115  -50
SonarUnit  7  115  -130  -90

SonarUnit  8   -115  130   -90
SonarUnit  9   -155  115  -140
SonarUnit  10  -190  80   -120
SonarUnit  11  -210  25   -100
SonarUnit  12  -210  -25   100
SonarUnit  13  -190  -80   120
SonarUnit  14  -155  -115  140
SonarUnit  15  -115  -130   90

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

Appendix D
Specifications

Physical Characteristics	**Pioneer 2**	**Pioneer 2-AT**
Length	44cm	50cm
Width	33cm	49cm
Height (body)	22cm	24cm
Body clearance	5.1cm	5.5cm
Weight	9kg	14kg
Payload	20kg	20kg
Construction		
Body	1.6mm CNC fabricated aluminum	Same
Console & main deck	2.4mm CNC fabricated aluminum	Same
Assembly	Allen hex screws (metric)	Same
Power		
Battery	3x12V lead-acid	Same
Charge	252 watt-hr; hot-swappable	Same
Run time	8–10 hrs	4-6 hrs
Recharge time	8 hrs	Same
Mobility		
Drive wheels	2 solid rubber, with	4 pneumatic
	rear balancing caster	4-ply all-weather tread
Wheel diameter	16.5cm	22cm
Wheel width	3.7cm	7.5cm
Steering	Differential	Skid
Gear ratio	19.7:1	38.3:1
Swing radius	26cm	34cm
Turn radius	0cm	Same
Translate speed max	1.6 m/sec	1.2 m/sec
Rotational speed max	300 degrees/sec	360 degrees/sec
Traversable step max	2cm	8.9cm
Traversable gap max	8.9cm	12.7cm
Traversable slope max	30% grade	80% grade

Traversable terrains	All wheelchair accessible	All surfaces
Sensors		
Front Ultrasonic	8	Same
	1 each side	Same
	6 forward @ 20° intervals	Same
Rear Ultrasonic	8	Same
	1 each side	Same
	6 rearward @ 20° intervals	Same
Position encoders	9,850 ticks per revolution	Same
Electronics (basic onboard microcontroller)		
Processor	Siemens 8C166 (20 MHz)	Same
Position inputs	4	Same
Sonar inputs	2x8 (multiplexed)	Same
Digital I/O	16 logic ports	Same
A/D	5 @ 0-5 VDC	Same
Digital timer output	8 @ 1µsec resolution	Same
Digital timer inputs	8 @ 1µsec resolution;	Same
Comm port	2 RS-232 serial	Same
FLASH PROM	32 KB; P2OS-encoded software	Same
RAM	256 KB; user programmable	Same
Power switches	1 main; 2 auxiliary	Same
Controls, Ports and Indicators		
Main Power	Robot power On/Off	Same
	red LED indicator Console	
Charge	System power/battery recharge	Same
LCD display	Systems status and messages	Same
RESET pushbutton	Warm reboot;	Same
MOTORS pushbutton	Motors/boot/self-tests	Same
	green LED indicator	
RADIO	Radio modem or Ethernet	Same
Speaker	Piezo	Same
Serial comm port	9-pin RS-232 with RCV and	Same
	XMT LED indicators	

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 responsibilities 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 support of any special or custom modification to Pioneer or its software.

ActivMEDIA
INCORPORATED

44 Concord Street
Peterborough, NH 03458
(603) 924-9100
(603) 924-2184 fax
http://www.activmedia.com/robots