Pioneer 2 / PeopleBot™

Operations Manual

for P2OS-based ActivMedia Robots

PIONEER 2-DXE PIONEER 2-AT PEOPLEBOT™

Important Safety Instructions

- ✓ Read the installation and operations instructions before using the equipment.
- ✓ Avoid using power extension cords.
- ✓ To prevent fire or shock hazard, do not expose the equipment to rain or moisture.
- ✓ Refrain from opening the unit or any of its accessories.
- ✓ Keep wheels away from long hair or fur.

Inappropriate Operation

Inappropriate operation voids your warranty! Inappropriate operation includes, but is not limited to:

- ✓ Dropping the robot, running it off a ledge, or otherwise operating it in an irresponsible manner
- ✓ Overloading the robot above its payload capacity
- ✓ Getting the robot wet
- ✓ Continuing to run the robot after hair, yarn, string, or any other items have become wound around the robot's axles or wheels
- ✓ All other forms of inappropriate operation or care

Table of Contents

Chapter 1 Introduction

Figure 1. The Pioneer 2-DX and -AT Mobile Robots first appeared commercially in 1995.

Congratulations on your purchase and welcome to the rapidly growing community of researchers, developers, and enthusiasts of *Activ*Media Robotics' intelligent mobile robots.

This *Pioneer 2 Operations Manual* provides both the general and technical details you need to operate your Pioneer 2-DX, -CE, -DXe, -AT, or PeopleBot Mobile Robot and to begin developing your own Robotics hardware and software. Please consult the *Pioneer 2 H8* or the *Performance PeopleBot* manuals if you own a newer, Hitachi H8S-based robot.

Robot Package

Our experienced manufacturing staff put your mobile robot and accessories through a "burn in" period and carefully tested them before shipping the products to you. In addition to the companion resources listed above, we warranty your *Activ*Media 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.

Even though we've made every effort to make your *Activ*Media Robotics package complete, please check the components carefully after you unpack them from the shipping crate.

Basic Components (all shipments)

- ✓ One fully assembled mobile robot with battery
- ✓ CD-ROM containing licensed copies of *Activ*Media software and documentation
- ✓ Hex wrenches and assorted replacement screws
- ✓ Replacement fuse
- ✓ Set of manuals
- ✓ Registration and Account Sheet

Optional Components and Attachments (partial list)

- ✓ Battery charger (some contain power receptacle and 220VAC adapters)
- ✓ Onboard PC computer with PC104+ bus, hard-drive and other accessories
- ✓ Radio Ethernet and/or serial modems; one mounted inside the robot
- ✓ Companion radio for LAN or basestation connection
- ✓ Supplementary and replacement batteries
- ✓ 3-Battery Charge Station (110/220 VAC)
- ✓ Added sonar arrays
- ✓ Gripper
- ✓ 5-DOF Arm with gripper
- ✓ *Activ*Media Color Tracking System (ACTS)
- ✓ Stereo Vision Systems
- ✓ Pan-Tilt-Zoom Surveillance Cameras
- ✓ Custom Vision System
- ✓ Range-finding laser
- ✓ Global Positioning System
- ✓ Compass
- ✓ Bumper rings

- ✓ Serial cables for external connections
- ✓ Many more…

User-Supplied Components / System Requirements

- ✓ Client computer: 586-class or later PC with Microsoft Windows© 9x/ME, or RedHat© Linux operating system
- ✓ One RS-232-compatible serial port
- ✓ Four megabytes of available hard-disk storage

Additional Resources

New *Activ*Media Robotics Pioneer 2 and PeopleBot customer get three additional and valuable resources:

- ✓ A private account on our Internet server for downloading software, updates, and manuals
- ✓ Access to private newsgroups
- ✓ Direct access to the *Activ*Media Robotics technical support team

Software

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

http://robots.activmedia.com

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

Newsgroups

We maintain several email-based newsgroups through which *Activ*Media robot owners share ideas, software, and questions about the robot. Visit the support `http://robots.activmedia.com` website for more details. To sign up for pioneer-users, for example, send an e-mail message to the `–requests` automated newsgroup server:

> To: **pioneer-users-requests@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 email comments, suggestions, and questions intended for the worldwide community of Pioneer users:[1]

> To: **pioneer-users@activmedia.com**
> From: <*your return e-mail address goes here*>
> Subject: **<something of interest to 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 or PeopleBot robots.

[1] Note: Leave out the –requests part of the email address when sending messages to the newsgroup.

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 our robots? Share your thoughts and questions directly with us:

`support@activmedia.com`

Please include your robot's ***serial number*** (look for it beside the Main Power switch)—we often need to understand your robot's configuration to best answer your question.

Your message goes directly to the *Activ*Media Robotics 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 your robot or software.

See Chapter 8, *Maintenance & Repair,* for more details.

Chapter 2 What Is Pioneer?

Figure 2. ActivMedia Robots

Pioneer is a family of mobile robots, both two-wheel and four-wheel drive, including the Pioneer 2-DX, -DXe, -CE, and -AT, and the PeopleBot (V1 and Performance) Mobile Robots. All are intelligent mobile robots, whose client-server architecture was originally developed by Kurt Konolige, Ph.D., of SRI International, Inc. and Stanford University.

*Activ*Media's robots are truly intelligent, off-the-shelf mobile platforms, 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, integrated sensors, and accessories. They are all managed by an onboard microcontroller and mobile-robot server software.

Your *Activ*Media robot also has a variety of expansion power and I/O ports for attachment and close integration of additional sensors and other accessories. Expansion includes an addressable I/O bus for up to 16 devices, two RS-232 serial ports, eight digital I/O ports, five A/D ports, PSU controllers and more—all accessible through a common application interface to the robot server software, P2OS.

With the onboard computer option, your *Activ*Media robot becomes an autonomous agent. With Ethernet-ready onboard autonomy, your robot even becomes an agent for multi-intelligence work.

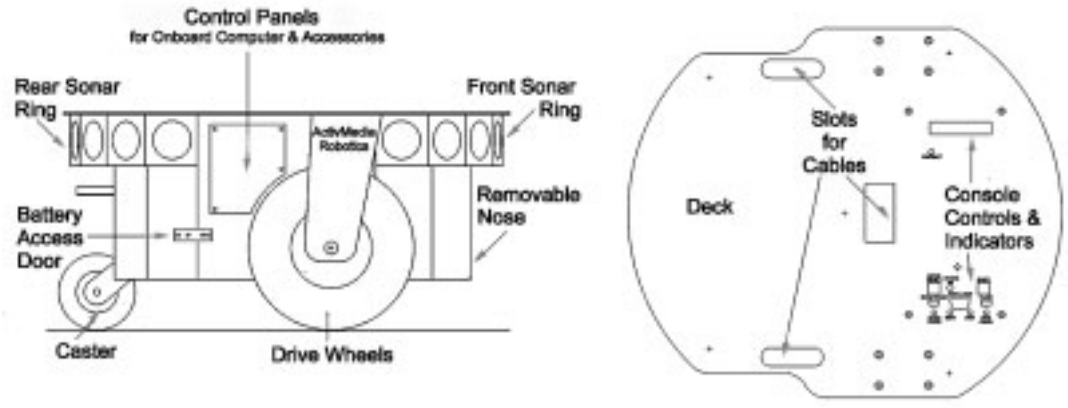

Figure 3. Components and some accessories of the Pioneer 2-DX, many of which are identical for the Pioneer 2-DXE and CE.

Client Software

Your *Activ*Media robot is the server in a client-server environment: It handles the low-level details of mobile robotics, including maintaining the platform's drive speed and heading over uneven terrain, acquiring sensor readings, such as the sonar, and

managing attached accessories like the Gripper. To complete the client-server architecture, *Activ*Media robots require a client connection: software running on a computer workstation connected with the robot's controller via a serial link that provides the high-level, intelligent robot controls, including obstacle avoidance, path planning, features recognition, localization, and so on.

An important benefit of *Activ*Media Robotics' client-server architecture is that different robot servers can be run using the same high-level client. For example, we provide a robot simulator that runs on the host machine that can look and act just like your real robot. With the Simulator, you may conveniently perfect your application software, then run it without modification on any *Activ*Media robot. Several clients also may share responsibility for controlling a single mobile server, which permits experimentation in distributed communication, planning, and control.

Currently available client software and development environments for the Microsoft Win32- or Red Hat© Linux-based computing platform of your choice include:[2]

- ✓ *Activ*Media Robotics Basic Suite with WorldLink Internet surveillance and control
- ✓ *Activ*Media Robotics Interface for Applications (ARIA)
- ✓ *Activ*Media robot simulator
- ✓ The Saphira client development suite with Colbert

Versions and updates for supported computing platforms are available to password-registered customers for download from our software website:

`http://robots.activmedia.com`

ActivMedia Robotics Basic Suite

To better support our customers, *Activ*Media Robotics software designers have blended and refined the best of advanced mobile robotics software found in the many development environments into a suite of state-of-the-art software tools and applications. We call this suite *Activ*Media Robotics Basic Suite, and it includes the following five modules plus a robot simulator:

- ✓ **NAVIGATOR** is the crown jewel—a sophisticated graphical-user control module with which you access your *Activ*Media robot's many intelligent capabilities, from guarded teleoperation to self-guided navigation along a planned path to a goal that you select onscreen with a click of the mouse. Navigator also lets you remotely share, connect and operate the robot from the Internet, or a local area network. You can see and hear from afar what your robot sees and hears through its camera and microphone. Navigator even lets you chat and exchange audio and video with others who may be connected simultaneously.

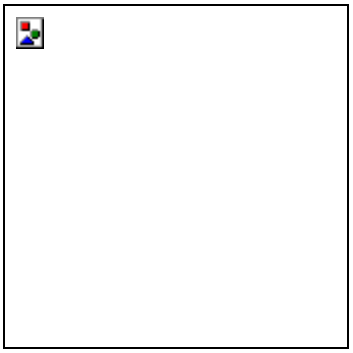

Figure 4. Navigator and WorldPass guide your robot to a goal with a click of the mouse.

- ✓ **WORLDPASS** is a free version of Navigator that lets you share you robot with colleagues, friends, and family. WorldPass provides all the networking and remote-control functionality of Navigator, including network video and audio, but only connects with a robot through a host running Navigator, not to one directly. You may distribute WorldPass to anyone you want

[2] Some software may come bundled with your robot. Other packages require purchase for licensing. Some software also are available for alternative operating systems, such as Macintosh, SunOS, Solaris, and BSD Unix.

to use your robot, but it will not appear on your own menu since it copies functionalities of Navigator, which you may not distribute.

- ✓ **MAPPER** provides the tools you need to construct a map of your robot's real operating space ("world"). Navigator and WorldPass use this map floor plan to plan a path from one point to another within a space.

- ✓ **TRAINER** is a programming editor and robot interface in which you create and perfect your own intelligent mobile robot-control programs with the simple, yet powerful Colbert programming language.

- ✓ **SIMULATOR** is not a separate module of Basic Suite. It is a connection option that provides a virtual replacement for your robot. By connecting to the simulator instead of a real robot, you can test Colbert programs, maps, and so on, when the real robot isn't practical or available.

- ✓ **AMIGOSOUNDS** is for AmigoBot only. With AmigoSounds, you assemble recorded sounds stored on your PC's disk into a playlist of sounds for your AmigoBot. With AmigoSounds, you may give AmigoBot different audio-based personalities.

ARIA

The *Activ*Media Robotics Interface for Applications (ARIA) is C++-based open-source development environment that provides a robust client-side interface to a variety of intelligent robotics systems, including your *Activ*Media robot's controller and accessory systems.

ARIA is the ideal platform for integration of your own robot-control software, since it neatly handles the lower-level details of client-server interactions, including serial communications, command and server-information packet processing, cycle timing, and multithreading, as well as a variety of accessory controls, such as for the PTZ robotic camera, the P2-Gripper, scanning laser-range finder, and motion gyros, among many others.

What's more, it comes with source code so that you may examine the software and modify it for your own sensors and applications.

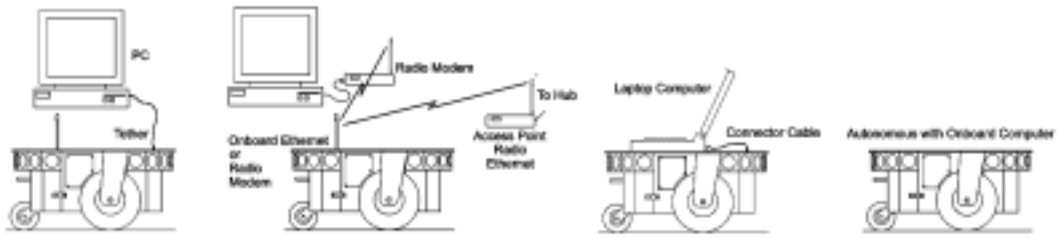

Figure 5. ActivMedia robots require a computer, typically a Windows© 9x/ME- or RedHat© Linux-based PC, to run client software for intelligent robotics command and control operations.

Saphira

Saphira, including the Colbert language, is a full-featured robotics control environment developed at SRI International's Artificial Intelligence Center. Saphira and its ARIA foundation form the robotics-control and applications-development foundation for

much of the *Activ*Media Robotics Basic Suite and many other ventures. The complete, licensed Saphira robotics development environment, including C/C++ libraries, GUI interface and Simulator, comes bundled with your *Activ*Media robot.

The Pioneer Legacy

Commercially introduced in August of 1995, Pioneer 1 was the original Pioneer platform. It introduced a single-board 68HC11-based robot microcontroller and the Pioneer Server Operating System (PSOS) software. It's low-cost and high-performance caused an explosion in the number of researchers and developers who now have access to a real, intelligent mobile robotic platform.

Pioneer 1

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 in the Summer of 1997 for operation in uneven indoor and outdoor environments, including loose, rough terrain.

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 2s.

Figure 6. All ActivMedia robots may be configured with a variety of integrated accessories, including (DX shown here) a very high-performance laser range-finder and robotic pan-tilt-zoom color camera with onboard PC and framegrabber for color-tracking and surveillance.

Pioneer 2 and PeopleBot

The next generation of Pioneer Mobile Robots—including the Pioneer 2-DX, -CE, and -AT, introduced in Fall 1998 through Summer 1999, improved upon the Pioneer 1 legacy while retaining its many important advantages.[3] Indeed in most respects, particularly with applications software, Pioneer 2 works identically with Pioneer 1 models.

[3] Price/performance ratio included! The much more capable and expandable Pioneer 2 was introduced four years later for just a few hundred dollars (US) more than the original Pioneer 1.

Sporting a more holonomic body, larger wheels and stronger motors for better indoor performance, the Pioneer 2-DX and CE models, like Pioneer 1, were two-wheel, differential-drive mobile robots.

Figure 7. Nearly identical to its research predecessor (right), the Performance PeopleBot (left) sports an attractive body design and bundled systems, including integrated Gripper, for commercial-consumer human-interaction applications.

The four-wheel drive Pioneer 2-AT has independent motor drivers, as well as a stall-detection system. And unlike its predecessor, the Pioneer 2-AT comes with inflatable pneumatic tires and metal wheels for much more robust operation in rough terrain, as well as the ability to carry nearly 30 kilograms (66 lbs) of payload and climb a 60-percent grade! The newest version of the 2-AT, introduced in mid-2001, includes an integrated joystick port for manual operation and a hinged top-plate for easy access to the internal systems.

The PeopleBot robots were introduced in 2000. They are architecturally Pioneer 2 robots, but with stronger motors and integrated human-interaction features, including a pedestal extension, integrated voice and sound synthesis and recognition.

The Performance PeopleBot is a fully redesigned, but very Pioneer 2-like system. Because of its attractive, non-threatening, and human-accessible body, Performance PeopleBot is ideal for human-interaction studies as well as for commercial and consumer mobile-robotics applications.

The latest Pioneer 2—the new Pioneer 2-DXe (Summer 2001)—replaces the DX. The DXe comes with pneumatic rubber tires for better mobility and a hinged top-plate for easy access to internal components.

All *Activ*Media Robotics Pioneer 2 and PeopleBot robots 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 your *Activ*Media robot, as well as your own custom attachments.

*Activ*Media robots also supports a full complement of 32 sonar in four arrays 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 and with Pioneer 2 platforms. This also means that the higher level clients, such as Saphira, ARIA, and others including your own software, will work with P2OS and any host Pioneer 2 or PeopleBot platform just as they had worked with Pioneer 1.[4] Of course, you will have to extend your client software, as we have done with Saphira, ARIA, and others, in order to take full advantage of P2OS.

[4] The two-time gold medal winners of the International RoboCup robot soccer competition used Pioneer 1s one year and quickly converted to Pioneer 2s in the next year.

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 with the tried-and-true Pioneer architecture.

Modes of Operation

You may operate your Pioneer 2 or PeopleBot robot in one of four modes:

- ✓ Joydrive
- ✓ Self-test
- ✓ Server
- ✓ Standalone

The Pioneer 2/PeopleBot microcontroller comes with 32K flash-programmable, read-only memory (FLASH-ROM) as part of its Siemens 88C166 microprocessor, and an additional 32K of dynamic RAM: 64K total memory space for your standalone 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 ARIA, or Saphira running on an onboard PC or other user-supplied computer, P2OS lets you take advantage of modern client-server and robot-control technologies to perform advanced robot tasks. (See Chapter 6, *Pioneer 2 Operating System*, for details.)

Most users run their *Activ*Media robot in server mode, because it gives them quick, easy access to its robotics functionality while working with high-level software on a familiar host computer.

For experiments in microcontroller-level operation of your robot's functions, you may reprogram the onboard flash-ROM and RAM for direct and standalone operation of your *Activ*Media robot. We supply the means to download, but not the microcontroller's programming software, for you to work in standalone mode.

In fact, the download utilities we provide for you to reprogram the 88C166-based controller's FLASH 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 the `pioneer-users` email newslist**.** That's where we notify our customers of the upgrades, as well as where we provide access to Pioneer and PeopleBot users worldwide.

Finally, we provide onboard software that lets you drive the robot from a tethered joystick. And we provide some self-test programs that exercise your robot's microcontroller hardware and software. We examine these modes in some detail in Chapter 5, *Joydrive and Self-Tests*.

Chapter 3 Specifications & Controls

*Activ*Media's robots may be smaller than most, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and much more expensive machines. For example, the Pioneer 2-DXe with onboard PC is a fully autonomous intelligent mobile robot, but unlike other commercially available robots, the DXe's modest size lends itself very well to navigation in tight quarters and cluttered spaces, such as classrooms, laboratories, and small offices.

At the same time, the powerful P2OS server with *Activ*Media Robotics client software, is fully capable of mapping its environment, finding its way home, and performing other sophisticated path planning tasks.

Physical Characteristics

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

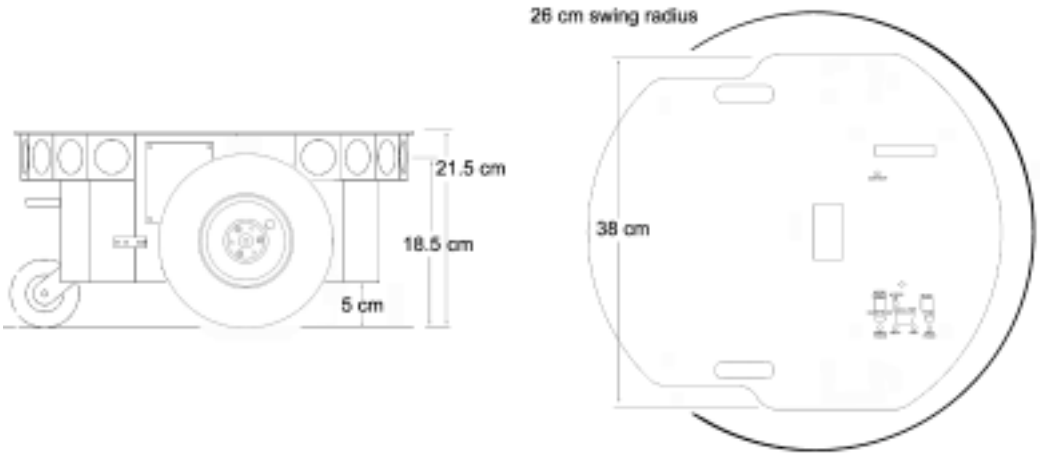

Figure 8. The Pioneer 2-DXE's physical dimensions and swing radius

These characteristics also permit it to carry extraordinary payloads: The DXe can carry up to 23 Kg (50 lbs.) additional weight; the Pioneer 2-AT can carry over 30 Kg (66 lbs.) more! Yet the Pioneer 2 is lightweight enough that it is also as easy to transport as a suitcase—a task made even easier by the DXe's built-in handle.

Main Components

*Activ*Media robots are composed of several main parts:

- ✓ Deck(s) and Console
- ✓ Body, Nose, and Accessory Panels
- ✓ Sonar Array(s)
- ✓ Motors and encoders
- ✓ Batteries and Power

Deck(s) and Console

The original Pioneer 2-DX, -CE, and -AT Decks are one piece—the top plate of the robot. The new DXe and AT models now have hinged top-plates which let you much more easily access the internal components of the robot.

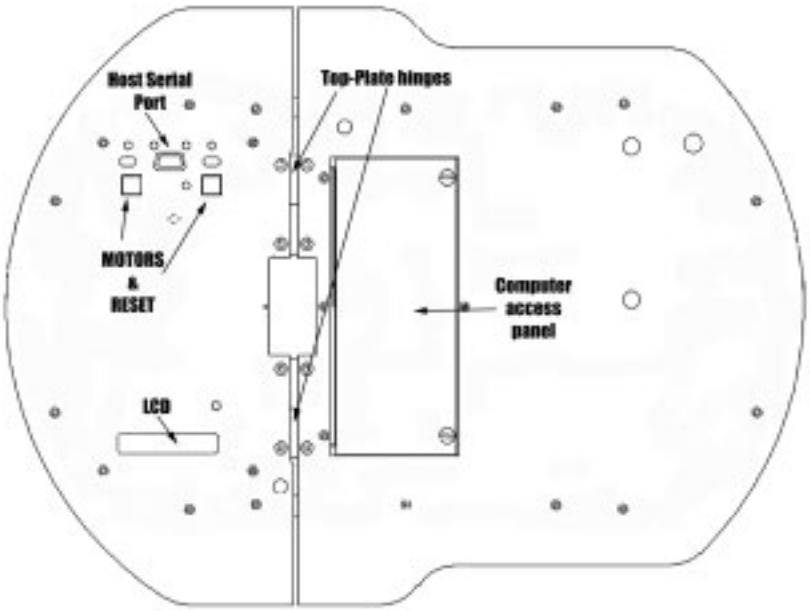

Figure 9. Pioneer 2-AT Console and hinged Deck

The robot's Deck is simply the flat surface for mounting projects and accessories, such as the PTZ Robotic Camera or a laser range finder. The PeopleBot's have lower and upper Decks.

Feed-through slots on each side of the DX, CE, and DXe Deck let you conveniently route cables to the accessory panels on each side of the robot. A removable plug in the middle of the Deck on all models[5] gives you convenient access to the interior of the robot.

In general, you should try to center the robot's payload over the drive wheels. If you must add a heavy accessory to the edge of the Deck, counterbalance the weight with a heavy object on the opposite end. A full complement of batteries helps balance the robot, too.

Figure 10. The Console is in the left column of the Performance PeopleBot.

The Pioneer 2's or PeopleBot's Console consists of a liquid-crystal display (LCD), MOTORS, and RESET control buttons and indicators, and an RS232-compatible serial port with a 9-pin DSUB connector. The Pioneer 2 and PeopleBot V1 Consoles sit at the front of the Deck. The Performance PeopleBot's Console is in the left column. Attached directly underneath the Console is the robot's microcontroller. Operations details are provided in the next chapter, *Quick Start*.

[5] Lower Deck of the PeopleBot's.

Body, Nose, and Accessory Panels

Your *Activ*Media robot's sturdy, but lightweight aluminum Body houses the batteries, drive motors, electronics, and other common components, including the front and rear sonar arrays. The Body also has sufficient room, with power and signal connectors, to support a variety of robotics accessories inside, including an A/V wireless surveillance system, radio modems or radio Ethernet, onboard computer, and more.

On all models except the DE, a hinged rear door gives you easy access to the batteries, which you may quickly hot-swap to refresh any of up to three batteries.

The PeopleBot V1 has a removable pedestal mounted to its base Deck. A removable back panel gives you access to internal wiring and components, including stereo speakers, A/V and Ethernet radios, and microphone preamplifier. On top of the pedestal is the upper Deck where you may add components like the PTZ Robotic Camera that normally mounts to the Deck of a Pioneer 2. PeopleBot's also have an additional front sonar array.

Both PeopleBot models include front and rear bump rings for stability as well as sensitive collision detection.

The Noses of Pioneer 2s and PeopleBots are empty, except when equipped with an onboard PC. The Nose is readily removable for access: Simply remove two screws from underneath the front sonar array. With the 2-AT, a third screw holds the Nose to the bottom of the Body; the DXe's Nose is hinged at the bottom.

Earlier Pioneer 2 models and the PeopleBots have a Nose that is secured by a single screw beneath the front sonar array and one on the bottom of the robot.

Figure 11. The PeopleBot V1 has a removable pedestal with top Deck and sonar array. A rear panel gives you access to the pedestal's internal components and connectors.

Once the mounting screws are removed, simply pull the Nose away from the Body.[6] This provides a quick and easy way to get to the accessory boards and disk drive of the onboard PC, as well as to the sonar gain adjustment for the front sonar array. The Nose also is an ideal place for you to attach your own custom accessories and sensors.

The Pioneer 2-DX, -DXe, -CE, and PeopleBot V1 robots come with removable panels on each side through which you may install accessory connectors and controls. A special side panel comes with the onboard PC option, for example, which gives users monitor, keyboard, mouse, and 10Base-T Ethernet access, as well as the means to reset and switch power for the onboard computer.

The AT comes with a single accessory panel in the Deck. Fastened down with finger-tight screws, the AT panel is accessible through a hinged door.

[6] With older Pioneer 2 models, you also need to remove the Gripper before removing the Nose. With the DXE and newer AT, the Nose and Gripper come off together, so you only need to remove the Nose's mounting screws. See Chapter 8, *Maintenance & Repair* for details.

All models come with an access port near the center of the Deck through which to run cables to the internal components.

Sonar Arrays with Gain Adjustment

Natively, *Activ*Media robots support 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. With sonar expansion electronics, you may add up to 16 more sonar in two additional arrays of eight sonar each. PeopleBots, for instance, have an additional array at the front of the upper Deck.

The sonar positions in all arrays are fixed: one on each side, and six facing outward at 20-degree intervals. Together, fore and aft sonar arrays provide 360 degrees of nearly seamless sensing for the platform.

Each sonar array comes with its own driver electronics for independent control**.** Each arrays' sonar are multiplexed; the sonar acquisition rate is 25 Hz (40 milliseconds per sonar per array. Sensitivity ranges from ten centimeters (six inches) to nearly five meters (16 feet). You may control the sonar's firing pattern through software; the default is left-to-right in sequence 0 to 7 for each array.

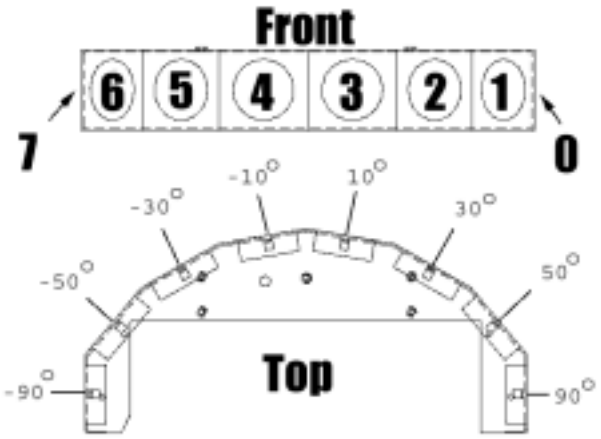

Figure 12. ActivMedia robot sonar array

The driver electronics for each array is calibrated at the factory. However, you may adjust the array's sensitivity and range to accommodate differing 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.

Sonar sensitivity adjustment controls are accessible directly, although you may need to remove the Gripper to access the front sonar, if you have that accessory attached.[7] For the front sonar, for instance, locate a hole near the front underside of the array through which you can see the cap of the sonar-gain adjustment potentiometer. Using a small 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 robot's 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 floor—a heavy shag carpet, for example. If the sonar are too sensitive, they will "see" the carpet immediately ahead of the robot as an obstacle.

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

[7] It's easier to remove the DXE's Nose with Gripper attached.

Motors and Position Encoders

Pioneer 2 and PeopleBot drive systems use high-speed, high-torque, reversible-DC motors, each equipped with a high-resolution optical quadrature shaft encoder for precise position and speed sensing and advanced dead-reckoning. Motor gearhead ratios and encoder ticks per revolution vary by robot model. See *Appendix D* for details.

Batteries and Power

Except the 2-CE, Pioneer 2 and PeopleBot robots may contain up to three, hot-swappable, seven ampere-hour, 12 volts direct-current (VDC) sealed lead/acid batteries (total of 252 watt-hours), accessible though a hinged and latched back door. We provide a suction cup tool to help grab and slide each battery out of its bay. Spring contacts on the robot's battery power board alleviate the need for manually attaching and detaching power cables or connectors.

Balance the batteries in your robot.

Battery life, of course, depends on the configuration of accessories and motor activity. Pioneer 2-AT and PeopleBot charge life typically ranges from two to three hours. The Pioneer 2-DX and -DXe run continuously for six hours or more; up to four hours with onboard computer. If you don't use the motors, your robot's microcontroller will run for several days on a single battery charge.

IMPORTANT: Batteries have a significant impact on the balance and operation of your robot. Under most conditions, we recommend operating with three batteries. Otherwise, a single battery should be mounted in the center, or two batteries inserted on each side of the battery container.

Typical recharge time using the recommended accessory (800 mA) charger varies according to the discharge state; it is roughly equal to three hours per volt per battery. The Power Cube accessory allows simultaneous recharge of three swappable batteries outside the robot.

With the optional high-speed (4A maximum current) charger, recharge time is greatly reduced. It also supplies sufficient current to continuously operate the robot and onboard accessories, such as the onboard PC and radios. But with the higher-current charger, care must be taken to charge at least two batteries at once. A single battery may overcharge and thereby damage both itself and the robot.

Both recommended chargers are specifically designed for optimal and safe lead-acid battery recharging. Indicators on the module's face show fast-charge mode (typically an orange LED) in which the discharged batteries are given the maximal current, and trickle mode (green LED indicator) which the batteries are given only enough current to remain at full charge.

Electronics

Pioneer 2 and PeopleBot standard electronics reside on two main boards: a microcontroller and a motor-power distribution board. Each sonar array also has a controller board mounted in its base. A special I/O expansion board found inside the left column strut of the Performance PeopleBot distributes User I/O for use by the robot's joystick port as well as it's tabletop and breakbeam IR sensors.

A Main Power switch at the back of the robot controls power for the entire system. Processor control switches and indicators fit through the Console.

Motor-Power Board

Inside the robot, mounted to the battery box, is the Motor-Power board. It supplies both the 12 and five volts direct-current (VDC) power requirements of your robot's systems. The standard Motor-Power board has a 12-pin User-Power connector that supports four sets of five- and 12-VDC power ports (total 1.5 ampere) for custom accessories.

An optional computer-power section to the board supplies power for the onboard PC. It includes a special low-power and power-down circuit that lets you gently shut down the onboard PC without direct connection through a keyboard or monitor. (See Appendix B in this manual and the *Computer Tech Note*s that may accompany your robot for details.)

Microcontroller

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

All of the I/O ports, except those used for the motors, encoders, and sonar, are available to the user for accessory hardware. The embedded operating software (P2OS) lets you support and manage each of these I/O ports. Connector pinouts and electronics details appear in the Appendices.

Sonar Boards

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 power/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 Pioneer 2 and PeopleBot robots 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) is the Main Power Fuse. It is an automotive-type (spade terminals) 15A (DX, DXe, CE, and PeopleBots) or 20A (AT) fuse designed for tool-less replacement. To the left of the fuse, on the same battery connection board, is the main 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. Use the recommended accessory power charger or equivalent.

You should maintain your robot's batteries in a charged state above 11 VDC, as indicated on the Console 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 level (see Chapter 7, *Updating & Reconfiguring P2OS*), and the optional computer power circuitry

will automatically shut down the onboard PC. Discharging the batteries below 10 VDC may permanently damage them.

You may continue to operate the robot while charging its batteries, although that will lengthen the recharge time. Because the onboard PC draws much current, even the high-speed 4A charger will not be able to fully recharge the batteries unless you power down the PC.[8]

If you have only one battery onboard, plug your robot into the charger before "hot-swapping" the exhausted battery for a fresh one. To hot-swap two or three batteries, replace each battery one-at-a-time, leaving at least one battery in place to supply power to the robot.

The Power Cube accessory is a convenient way to externally recharge one to three of the hot-swapped batteries.

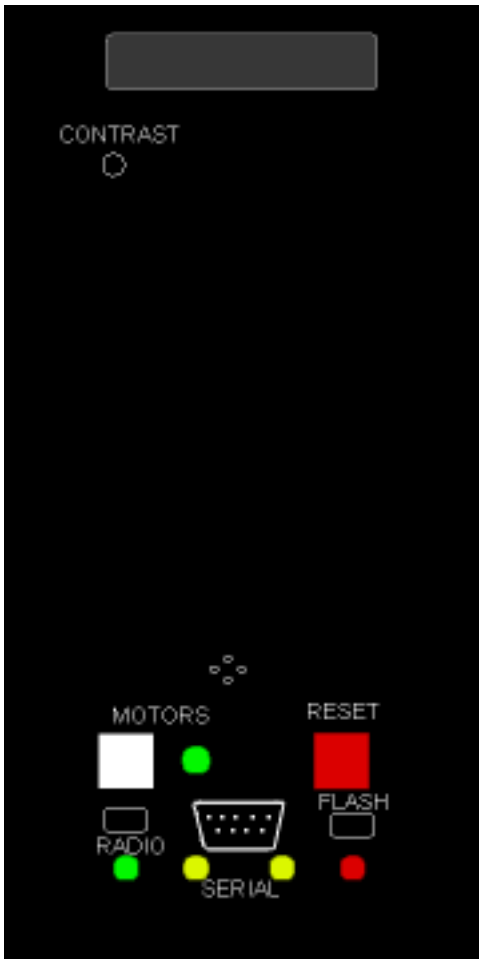

Figure 13. The Console

Liquid-Crystal Display & Contrast Adjustment

Information about your robot's state and connections appears on a 32-character (two lines) liquid-crystal display (LCD) on the Console. When under control of the P2OS servers, for example, the display shows the state of communication with the client computer, along with the battery voltage and a blinking "heartbeat" asterisk (*) in the second line of text.

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 Console affect the microcontroller's logic and motor-driver systems.

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 microcontroller's RAM, so any user programs downloaded in standalone mode get restarted.

> The MOTORS pushbutton is *not* a power switch—it does not directly control power to the motors.

[8] We deliberately limit the charger power to avoid serious damage that can occur from overcharging lead-acid batteries.

The MOTORS button and its associated green LED are under software control. Normally, your *Activ*Media robot's motors are disabled when not connected with a client, such as Navigator or Saphira. When first connected with a client,[9] the motors remain disabled (LED flashes) until you press and release the MOTORS button. Pressing and releasing the white MOTORS button then enables/disables the motors as long as the robot remains connected with a client.[10] The green LED should light continuously when the motors are enabled and blink ON and OFF when disengaged.

When not connected with a client, pressing and releasing the MOTORS button puts your robot into joystick-drive mode. A subsequent press and release of the MOTORS button puts the robot into Self-Test Mode that exercises the robot's drive, controller, and I/O systems. See Chapter 5, *Joydrive and Self-Test Modes,* for details.

Press and hold the MOTORS button in combination with the RESET button to put the microcontroller into a special system-download mode for reprogramming the onboard FLASH ROM. See *Pioneer 2 Operating System* and *Updating & Reconfiguring P2OS*, Chapter 6 and Chapter 7.

SERIAL

Your *Activ*Media robot's microcontroller has two serial ports and three connectors. One connector, labeled SERIAL, is a standard 9-pin D-SUB receptacle located on the Console and is for direct RS232-compatible serial data communication between the microcontroller and a client computer. This "Host" serial port shares its three-line transmit, receive, and ground connections with one of the two serial connectors that is inside the robot. See Appendix A for pinouts and cabling connections.

Amber LEDs on each side of the Host serial port light during data-exchange activity transmitted from or received by the microcontroller.

> Remove any tether or laptop connection from the Console SERIAL port when using the optional radio modems.

RADIO

The RADIO slide switch on your robot's Console controls power to the optional radio modem or Ethernet radio. It does not affect the SERIAL port functions directly, but you must switch the 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 power as well as signals through the internal, shared Host serial port and can interfere with Console SERIAL communications. In some cases, you may have to physically remove the radio serial connection from the microcontroller to eliminate that interference.

FLASH

A slide switch labeled FLASH is recessed into the Console. It write-protects the FLASH ROM-stored P2OS software and your robot's operating parameters (see *Updating and Reconfiguring P2OS*, Chapter 7). When switched forward, FLASH is enabled for writing. *Activ*Media's P2OS maintenance utilities warn you if FLASH is disabled.

---

[9] See the *Quick Start* chapter for client connection details.
[10] A P2OS command also lets you toggle the motors under client control.

PeopleBot Sensors and Emergency STOP

Performance PeopleBot's tabletop sensors are very reliable diffuse IR detectors mounted to the front of the robot and which detect obstacles, particularly tabletops or rope barriers, that otherwise aren't detected by the sonars. The tabletop IR detectors respond to any surface except glass or other mirrored surfaces, and can detect objects as thin as a human finger. They are oriented to trigger when an object is 28 cm (11.5 inches) or nearer to the front of the robot and 3.75 cm (1.5 inches) at the height of the lower deck.

Two "breakbeam" IR sensors, one on each side 3.75 cm (1.5 inches) forward of the left and right column struts and between the top and lower Decks of the Performance PeopleBot, sense objects which intrude into the robot's profile, but which may not be otherwise detected by the sonars or tabletop IR sensors.

Since the tabletop and breakbeam IR sensors are connected to User I/O digital ports, their states are communicated from the P2OS server to a connected client, such as Saphira, in the standard Server Information Packet. See the . *IOpac packet* contents

Pioneer 2 Arm-related SIPs and Commands

Please consult the Pioneer 2 Arm Manual for details.

Performance PeopleBot IRs sections in Chapter 6 and in Appendix A for details.

The Performance PeopleBot contains a large, red Emergency STOP button prominently positioned on the left column just beneath the upper Deck. When pressed, the button physically disconnects power from the motors and electrically shorts them to brake the motors. A separate digital sense line to the microcontroller can activate an emergency stop process in P2OS. See *E_STOP and E_STALL* in Chapter 6 for details. To release the Emergency STOP button, press it in and twist.

Safety Watchdogs and Configuration

Pioneer 2's and PeopleBot'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 parameter). The robot will automatically resume activity, including motion, as soon as communications are restored.

P2OS also 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). The server software also notifies the client which motor is stalled. When the stallwait time elapses, motor power automatically switches back ON and motion continues under server control.

All these "failsafe" mechanisms help ensure that your robot will not cause damage or be damaged during operation. You may reconfigure the communications, drive current, and stallwait values to suit your application. See Chapter 7, *Updating & Reconfiguring P2OS*, for details.

Chapter 4 Quick Start

Your *Activ*Media robot comes ready for action.[11] This chapter describes how to operate the mobile robot with the Saphira demonstration software. For more details about programming and operating your *Activ*Media mobile robot with Saphira, ARIA, or other client software, see their respective programming manuals.

Preparative Assembly

Out of the box, your *Activ*Media robot comes fully assembled, with its batteries fully charged—just slide them into the battery box through the back door. You also may need to attach an antenna or plug in an accessory that we intentionally left unattached so as to prevent damage during shipping. Consult the Tech Notes and accessory manuals that may accompany your robot for final assembly details.

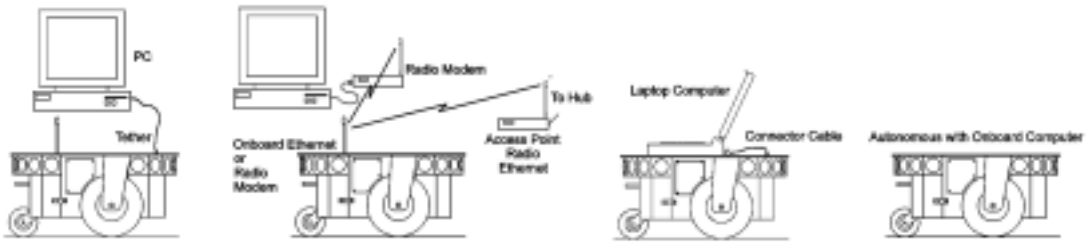

Figure 14. Your robot server needs a serial link with a client computer.

Your robot requires a serial communication link with a client for operation. The serial link may be:

- ✓ A tether from the robot's 9-pin serial connector on the Console to a computer
- ✓ A piggyback laptop cabled to the robot's 9-pin serial connector on the Console
- ✓ An optional radio modem pair—one inside the robot attached to the microcontroller's host serial port, and a companion radio modem connected to the serial port of the client computer
- ✓ An integrated onboard PC wired internally for direct onboard control

Saphira Client Installation

The Saphira client software-development environment, including the Saphira demonstration program and robot Simulator, comes on CD-ROM with your new robot. It is configured and compiled for Windows© 9x/ME as well as for RedHat© Linux-based PCs.

If you have an onboard PC installed with software from the factory, the Saphira demonstration program already is installed. *Activ*Media Robotics customers also may obtain Saphira and related software and updates for other platforms from our support website (see Chapter 1, *Introduction*, 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 of hard-disk space. To install the software, follow the instructions in the accompanying README file.

[11] You may need to attach some accessories and the PeopleBot V1's pedestal needs to be attached.

For instance, the Windows© 9x/ME version is a self-extracting WinZip© archive. Simply double-click its .EXE icon and follow the extraction program's instructions.

The distribution archive decompresses into a Saphira\verxx directory of files.[12]

Windows© 9x/ME users also need to set an environment variable in the C: (boot drive) autoexec.bat file. Load the file with a simple text processor like WordPad and add the line:

```
SET SAPHIRA=C:\Saphira\ver62
```

which is the default installation path to Saphira's top-level directory. Then reboot.

Linux users should be sure they have permission to read/write through their PC's serial port that connects with the robot. The default is /dev/ttyS0. Saphira also requires X-Window. If you have an onboard PC with Ethernet radio, export the X-Windows display if you are operating the Saphira GUI from a remote X-terminal over the network.

With bash, for example:

```
%export DISPLAY=remote_computer:0
```

where remote_computer is the known hostname (/etc/hosts or DNS entry) or IP address of the remote X-terminal.

Saphira Client Start-Up

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

For instance, with the mouse double-click the saphira.exe icon inside C:\Saphira\ver62\bin on your Windows© 9x/ME desktop. On your UNIX or Linux machine via an X-Windows terminal, navigate (cd) to the /usr/local/Saphira/ver62/bin directory and type saphira, or ./saphira to execute the Saphira client software there. We set the ${PATH} for the onboard, Linux-based PC systems that we install for you, so that you need only type saphira in an X-Window terminal to execute the demonstration program.

If the demonstration program is installed and executed properly, a Saphira main window should open and appear on your graphics screen. Otherwise, diagnostic error dialogs will inform you of the problem. See *Quickstart Troubleshooting* below for additional tips.

Robot Cold Start-Up

Position your *Activ*Media robot on the floor or ground in an open space. Slide the Main Power switch up to ON. The red power indicator LED beneath the FLASH switch on the Console should light. After a short P2OS initialization phase, the LCD on the control panel then displays the current status, the Host serial port baud rate, the current P2OS version number, and the battery charge, in volts. For example,

```
no conn  19.2 kB
P2OS v1.F  13.8*
```

The "heartbeat" asterisk character following the battery voltage should be flashing ON and OFF.

[12] Current verxx is ver62.

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

RADIO ON

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

Starting Client-Server Communications

After it starts up, resets, or completes its self-tests, your robot's onboard servers (P2OS) enters disconnect ("noconn") mode—a quiet state in which P2OS awaits connection with a client computer and software like Saphira or ARIA.

To establish a connection between your robot and the Saphira demonstration client, pull down the Connect menu and engage the appropriate serial port: It's the one that you connected to the robot via a direct cable or through modems. With the onboard PC, it is the first (default) serial port, COM1 or /dev/ttyS0. The port name varies by computer platform.[13]

The Saphira client initiates a connection with your robot server by exchanging three synchronization packets. You may monitor this process on your robot's LCD and in Saphira'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. Press RESET to return P2OS to its client-connection waiting state.

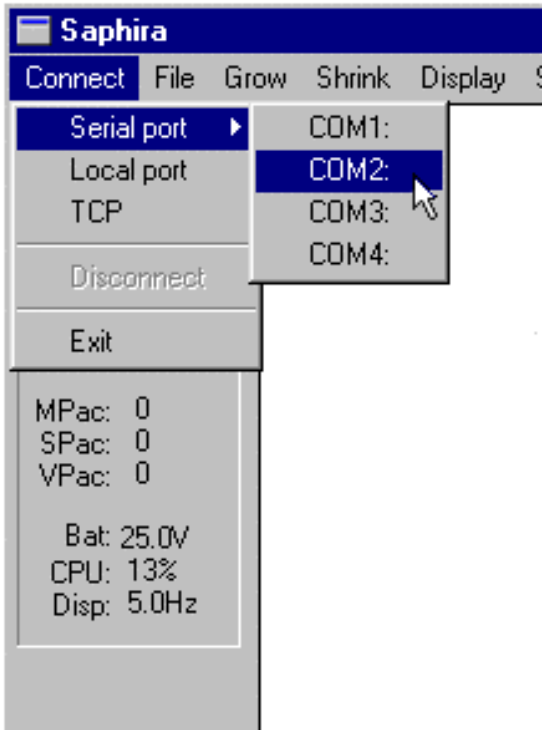

Figure 15. Connect Saphira with your ActivMedia robot through a serial port.

A Successful Connection

After Saphira negotiates a connection successfully, the client requests various P2OS servers to start their activities, including sonar polling, position integration, and so on. The microcontroller sounds an audible connection cue, and you should hear the robot's sonar ping with a distinctive and repetitive clicking.

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 to P2OS-server communications. The Console LCD also should display a message similar to the following:

[13] Choose Local Port to connect with the simulator.

```
Connected
P2OS v1.F 13.2*
```

Operating the Saphira Demonstration Client

When communications between the Saphira client and your robot's servers are established, the robot becomes responsive and intelligent. For example, although it may drive toward an obstacle, your Pioneer 2-DXe will not crash (unless its obstacle-avoidance behaviors have been disabled) because it can detect and actively avoid collisions.

Engage the robot's motors (white MOTORS button) after connecting or it won't move, no matter how excited you get.

Collision avoidance is just one of the many mobile robot behaviors available through Saphira and other client software available from *Activ*Media Robotics, including the ActivMedia Robotics Basic Suite, ARIA, and others. Remember, this section is meant to familiarize you with your new robot. Please read the respective software manuals for how to get the most out of your robot and software.

Table 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	Toggle constant velocity
c	Toggle obstacle avoidance

The main window of the Saphira client displays a sonar map built by Saphira as the robot moves around its environment. Landmarks may be defined as Saphira recognizes and classifies certain sensor data patterns—walls and openings, for example.

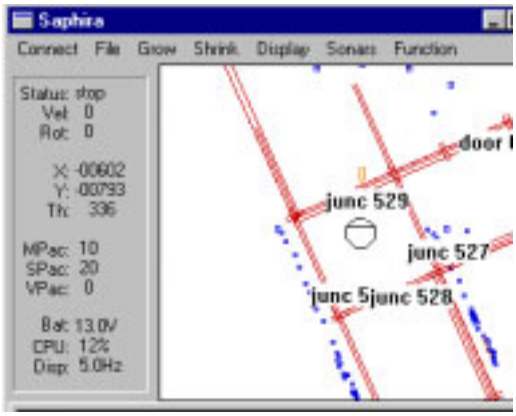

Figure 16. The main window of a robot-connected Saphira client

In Figure 17, for example, a Pioneer 2-DX (center octagon) has identified a corridor and several doors. Notice the small dots, which are recent reflections detected by the front sonar array? The long lines through the onscreen sonar artifacts are the calculated corridor's geometry. The rectangles directly ahead of the robot represent an obstacle "detected and of interest" to the robot. One of Saphira's behaviors, by the way, is to have your robot seek and traverse the center of a corridor.

You may enable and disable Saphira behaviors that intelligently guide your *Activ*Media robot 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.

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 press the left- or right-turn key rapidly several times in a row, because the turn increment is small.

Disconnecting Serial Communications (intentionally or unintentionally)

When you finish playing with your robot, pull down the Saphira client's Connect menu and choose the Disconnect option.

Your Pioneer 2 or PeopleBot will disengage its drive motors and stop moving automatically, and its sonar should stop firing. The LCD also should return to the waiting-state message. You may now slide the robot's Main Power switch to OFF.

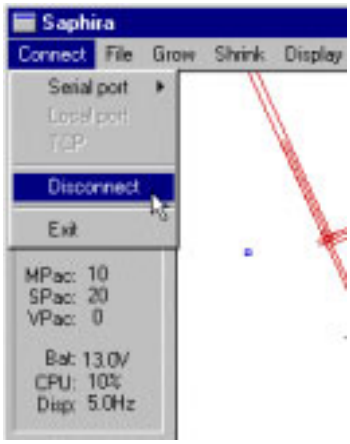

Figure 18. Please, gracefully disconnect the Saphira client from the robot server.

Quickstart Troubleshooting

You must have a valid Saphira license to connect with a robot. Unlicensed Saphira clients may only connect with the simulator. Licensed versions of Saphira come with the robot and can be downloaded by customers from the *Activ*Media Robotics support website:

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

Most problems occur when attempting to connect the Saphira client with a robot for the first time. Make sure you have Saphira properly installed and its related 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 the connection serial port. On the server side, make sure your radio modem is ON, if that is the connection route.

If you access the wrong serial port, the Saphira demonstration program will complain, "Error opening" that 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 get severed between the Saphira client and P2OS 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 also will 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 servers back to their wait state, ready to accept client connections again. If the Saphira client application is still active, simply pull down the Connect menu and choose the Disconnect option. Otherwise, restart the application and reconnect the Saphira client with the P2OS servers.

The Saphira-P2OS connection is SERIAL *only*.
You may not connect a Saphira or other client software running on an offboard computer with the P2OS controller over a TCP/IP link.

Rather, you need the special IPTHRU program running on an onboard computer and which translates IP packets into P2OS serial packets to support a TCP/IP-based remote client.

This is not the same as EXPORTING the onboard PC controls, in which Saphira or other client software connects via a serial link with the P2OS servers locally, but the computer is otherwise operated from a remote terminal.

Chapter 5 Joydrive and Self-Tests

Beginning with P2OS version 1.C, all Pioneer 2s and PeopleBots may be tethered and driven manually with a common PC (analog) joystick. In version 1.G and later, you may also use the tethered joystick to manually drive the robot even when it is connected with a client. And P2OS comes with a series of short test routines for your *Activ*Media robot's drive motors, sonar, User I/O functions, and accessories.

To run in either joydrive or self-test mode, 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 self-test and joydrive modes. To enable joydrive while connected with a client, you must send the P2OS command number 47 with the integer argument 1. See *Chapter 6, Pioneer 2 Operating System*, for details.

Joystick Connection

You may attach a common PC joystick to the User I/O's analog ports and manually drive your robot around without the need for a client connection and incumbent accessories. But it also means that you may drive your Pioneer 2 around without the niceties of a client's protective behaviors like obstacle avoidance—so be careful. Pioneer 2-ATs and the Performance PeopleBot come standard with a joystick connector and internal wiring. An accessory kit is available to adapt other models. See *Appendix C* for connection details.

Beware:
The Joystick and the older Gripper interfere at the User I/O port.

Because they share ports on the User I/O connector of the microcontroller, older versions of the Pioneer 2 Gripper interferes with joystick operation.[14] One solution is to switch the control lines with a 4PDT toggle switch. See *Appendix C* for details.

Alternatively, there is an accessory kit available that moves the Gripper controls to the General I/O bus, liberating the User I/O ports for other uses, including attaching a joystick. Visit the Pioneer webpages (http://www.activrobots.com) and contact sales@activmedia.com for details.

Joydrive Operation

To joydrive your robot when not connected with a client program like Saphira or ARIA, attach a PC joystick to the 15-pin DSUB socket of the robot's joystick port. Then, after switching the robot's Main Power ON, press the white MOTORS button once. Listen for a rhythmic, low-tone beeping indicating joydrive mode.

To joydrive while connected with a client, you must have the client send the P2OS command number 47 with an integer argument 1 to enable joydrive. Then, press and release the joystick fire button to enable self-calibration mode. Have your client send the P2OS Joydrive command number 47 with an integer argument of 0 to disable the joystick drive-override.

The joystick is self-calibrating: When you first enable joydrive mode, P2OS detects the joystick's center and extreme positions and saves these values to balance the driving action. Accordingly, rotate the joystick around its extreme limits and then let the joystick

[14] Identified as hasgripper = 1 in your robot's parameters; see Chapter 7, *Updating & Reconfiguring P2OS*, and the *Pioneer 2 Gripper Manual*, version 2 or later, for details.

handle find its default centered position before starting to drive. Try exiting (reset) and restarting joydrive mode if the joystick doesn't seem to function well.

The joystick's "fire" button (button 1) acts as the joydrive "deadman"—press it to start driving; release it to stop the robot's motors. The robot should drive forward and reverse, and turn left or right in response and at speeds relative to the joystick's position.

When not connected with a client control program, releasing the joystick fire button stops the robots. However, when connected with a client, the client program resumes automatic operation of your robot's drive system. So, for example, your robot may speed up or slow down and turn, depending on the actions of your client program.

You may adjust the maximum translational and rotational speeds, even disable joydrive mode, through special P2OS configuration parameters. See Chapter 7, *Updating & Reconfiguring P2OS*, for details.

Engaging Self-Tests

To enable self-test mode, press the white MOTORS button twice after startup or RESET.[15]

ATTENTION!
Place your robot on the floor or ground and have everyone step back ***before*** engaging self-tests.

A message should appear in the LCD:

```
Press again to
begin tests 13.2*
```

And there should be that rhythmic beeping sound coming from the Console speaker to alert you that you may have engaged joydrive and self-test modes inadvertently. If so, press the red RESET button. If you really intended to run the self-tests, move on...

Note that you may interrupt the current self-test and move on to the next one by pressing the white MOTORS button. Or you may discontinue all testing at any time and return to the P2OS servers' wait state by cycling power or, better, by pressing the red RESET button on your robot's Console.

Motors Test

The first self-test exercises your *Activ*Media robot'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. When ready, press the white MOTORS button.

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 counterclockwise, then clockwise.

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

```
Left forward
             13.2*
```

[15] As described above, the first MOTORS press and release puts the robot into joydrive mode.

Use the drive test to ensure that the drive motors are working and that the encoder cables are in their correct sockets on the motor-power 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. In that erroneous case, the robot will spin madly in place.

Sonar Test

When the motors test completes, self-tests automatically continue to the sonar tests.

You should hear the distinctive clicking sound as each sonar "pings", one from each of two arrays simultaneously, each in order from left to right in front and right to left in the rear. The LCD displays, in hexadecimal notation, the sonar number and the number 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:#B 7D00 12.4V*
```

Absent echoes return at the maximum time of 7D00.

Press the white MOTORS button to individually test each sonar in the arrays.[16] If you have added sonar such as with a PeopleBot, a second set of sonar tests automatically commence.

Bumpers

The next self-test displays the states of the inputs from the front and rear bumpers available for Pioneer 2 and PeopleBot robots. The state of each bump switch is mapped into a series of eight LCD display digits. The five switches in each bumper—front and rear—are related to the digits one through five, left to right, on the LCD. For instance in the example below, the front bumpers 1 and 5 are pressed; the rest are OFF, whereas in the rear bumper, the second and third bumps are ON.

```
Front 00100010
Rear  00001100
```

The first two and last display digits are not connected and may toggle between 0 and 1. Likewise, without an attached bumper, the readings may vary (usually all "0"s).

Press the white MOTORS button to proceed to the next test.

Gripper

Those of you fortunate enough to own a Gripper accessory for your Pioneer 2 or PeopleBot automatically get a self-test for the accessory. See the *Pioneer 2 Gripper Manual* for details.

Compass

If you own a TCM2 or V-2XG compass module, this self-test displays its compass heading in the LCD. See their respective manuals for details.

[16] Sonar tests automatically proceeded in P2OS versions before 1.F.

DIGIN and DIGOUT Test

A subsequent self-test lets you examine the values of the eight digital input (ID0–7) and output (OD0-7) ports associated with User I/O on your *Activ*Media robot's microcontroller. The state of each port is mapped into a series of eight digits displayed on the LCD. Each digit, 0 or 1, representing the ON or OFF state of a port, numbered right-to-left from 0 to 7. For instance, in the example below, digital input (DIGIN) ports 1 and 5 are ON; the rest are OFF:

```
DIGIN:  00100010
DIGOUT: 01010101
```

The DIGOUT ports automatically and rhythmically toggle ON and OFF.

The basic microcontroller has no attachments to the I/O ports. Because their normal state is floating, the DIGIN readings will vary. The DIGIN self-test becomes useful, of course, when you have equipment attached to one or more of the digital input ports.

See *Appendix A* for the location of the digital input and output ports.

ANALOG Tests

P2OS versions 1.2 and later let you alternatively use four of the digital input ports as analog-to-digital (A/D) input ports. And there is one dedicated A/D port on the User I/O connector. The ANALOG self-test measures the voltage (0-5 VDC) applied to each of the A/D ports and display its digitized value as a single byte in hexadecimal on the LCD:

```
ANALOG #1: A9
           13.2*
```

Press the white MOTORS button to select the next analog port, 2-5. Port #5 is the dedicated and default A/D port. Like DIGIN, the measured value is unreliable unless you connect something to the port.

See *Appendix A* for the location of the A/D ports on the microcontroller.

User PWMs

P2OS versions 1.2 and later let you alternatively use four of the digital output ports as pulse-width-modulating sources to drive servo motors, for example. The User PWM self-test puts a one millisecond pulse onto each of these four PWM ports. Monitor them with an oscilloscope.

See *Appendix A* for the location of the PWM ports on the microcontroller.

Chapter 6 Pioneer 2 Operating System

All *Activ*Media robots use a client-server mobile robot-control architecture developed by Dr. Kurt Konolige and others at SRI International. In the model, the robot's controller servers—the Pioneer 2 Operating System (P2OS)—work to manage all the low-level details of the mobile robot's systems. These include operating the motors, firing the sonar, collecting sonar and wheel encoder data, and so on—all on command from and reporting to a separate client application, such as ARIA or Saphira.

With this client/server architecture, robotics applications developers do not need to know many details about a particular robot server, because the client 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 and control your *Activ*Media robot via the P2OS client-server interface. The same P2OS functions and commands are supported in the various client-programming environments that accompany your robot or are available for separate license.

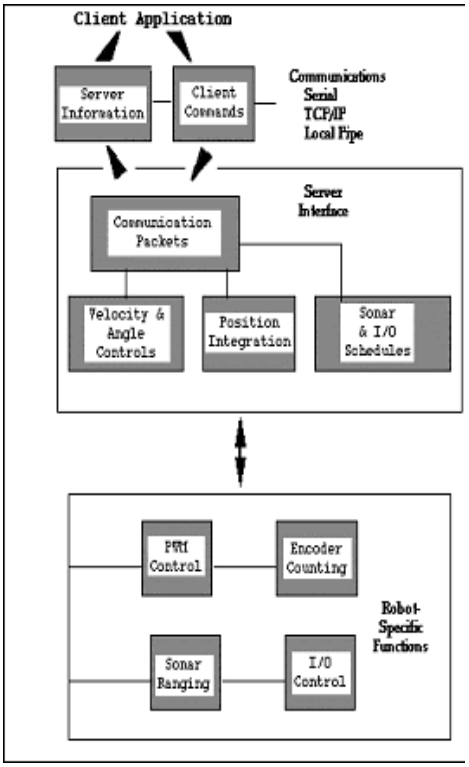

Figure 19. ActivMedia Robotics client-server control architecture

Experienced Pioneer users can be assured that P2OS is 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 original Pioneers.

Communication Packet Protocol

Table 2. Main elements of PSOS/P2OS 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 the Checksum word, but not the Byte Count. Maximum 200 bytes.
Data	N	command or SIP	Client command or server information packet (SIP)
Checksum	2	computed	Packet integrity checksum

P2OS communicates with a client application using a special packet protocol for both command packets from client to server and server information packets (SIPs) from the

server to client. Both are bit streams consisting of four main elements: a two-byte header, a one-byte count of the number of subsequent packet bytes, the client command and its arguments or the server information data bytes, and, finally, a two-byte checksum.

Packet Data Types

Client commands and server information packets contain several data types, as defined in Table 3.

Table 3. 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 PSOS/P2OS client-server 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.

```
int calc_chksum(unsigned char *ptr)      // ptr is array of bytes
{                                        // first is data count
  int n;
  int c = 0;

  n = (ptr++);          /* Step over byte count        */
  n -= 2;               /* don't include checksum 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);
}
```

NOTE: In P2OS, 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.

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.

Table 4. Standard P2OS Server Information Packet (SIP)

Name	Data Type	Description
Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 (checksum); must be less than 201 (0xC9)
Status/Packet Type	byte = 0x3S; where S =	Motors status
	2	Motors stopped
	3	Robot moving
Xpos	unsigned integer (15 ls-bits)	Wheel-encoder integrated coordinates; platform-dependent units; multiply by *DistConvFactor*‡ 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 (respectively Left and Right) in platform-dependent units; multiply by *VelConvFactor*‡—currently 1.0 for all—to convert into millimeters per second.
R vel	signed integer	
Battery	byte	Battery charge in tenths of volts
Stall and Bumpers	integer	Motor stall and bumper accessory indicators. Bit 0 of the lsbyte is the left wheel stall indicator = 1 if stalled. Bits 1-5 of that same byte correspond to the bump switch states (1=on) for the rear bumpers accessory. Bit 0 of the msbyte is the right wheel stall; the bits 1-5 of that same msbyte correspond to the front bumpers switch states.
Control	signed integer	Setpoint of the server's angular position servo—multiply by *AngleConvFactor*‡ for degrees
FLAGS (was PTU)	unsigned integer	b0 – motors flag (1=motors enabled) b1 – sonar flag: enabled if 1.
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 range; multiply by RangeConvFactor‡—currently 0.268 for all—for millimeters
…rest of the sonar readings…		
Timer	unsigned int	Selected analog port number 1-5
Analog	byte	User Analog input (0-255=0-5 VDC) reading on selected port
Digin	byte	User I/O digital input
Digout	byte	User I/O digital output
Checksum	integer	Checksum (see previous section)

‡ *Factors* see Table 17, Appendix D, and Saphira's parameter definition file for your robot.

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 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 server interface.

In the meantime, the *Activ*Media Robotics 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 re-issue them until achieving the correct state.

Server Information Packets

Like its Pioneer 1 PSOS predecessor, P2OS automatically sends a packet of information over the Host serial communication port to a connected client every 100 milliseconds.[17] The standard P2OS Server Information Packet (SIP) informs the client about a number of operating states and readings, using the order and data types described in Table 4.

P2OS also supports several additional server information packet types, extending the capabilities of the Pioneer 2 and PeopleBot robots. These include an "alternative" SIP that currently is not supported by Saphira or ARIA.[18] See following sections in this chapter for a description of the extended server information packet types.

Client Commands

P2OS has a structured command format for receiving and responding to directions from a client for control and operation of your *Activ*Media robot or its simulator.

Table 5. 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 following command bytes plus Checksum's two bytes, but not including Byte Count. Maximum of 200.
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), or string (sfARGSTR)
Argument (command dependent)	n	data	Command argument; integer or string
Checksum	2	computed	Packet integrity checksum

The number of client commands you may send per second depends on the Host serial baud rate, average number of data bytes per command, synchronicity of the communication link, and so on. Commands are processed at a maximum rate of one per millisecond, although the P2OS servers may not be up to the task of managing a deluge of commands.

[17] Or 50 milliseconds. Read on.
[18] Indeed, if enabled, the alternative SIP apparently will "break" the client software. Read carefully.

Table 6. 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	string	Set sonar polling sequence	3.9	1.0
ENABLE	4	int	Enable=1; disable=0 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
MOVE	8	signed int	Translation distance to move in mm		
SETRV	10	int	Resets maximum rotational velocity in degrees per second	4.8	1.0
VEL	11	signed int	Move forward (+) or reverse (-) at millimeters per second	3.x	1.0
HEAD	12	signed int	Turn to absolute heading (+) = counterclockwise; ± degrees	4.2	1.0
DHEAD	13	signed int	Turn relative to current heading (+) = counterclockwise; ± degrees	3.x	1.0
SAY	15	string	As many as 20 pairs of duration (20 ms increments) /tone (half-cycle) pairs; *int* is string length	4.2	1.0
CONFIG	18	int	1=Request configuration SIP	–	1.4
ENCODER	19	int	Request 1 or continuous stream (>1), or tell to stop sending (0) Encoder SIPs	–	1.4
RVEL	21	signed int	Rotate at ± degrees per second	4.2	1.0
DCHEAD	22	signed int	Heading setpoint relative to last setpoint; ± degrees; (+) = counterclockwise		
SETRA	23	signed int	Sets rotational (+)acceleration or (-)deceleration, in mm/sec/sec	–	1.0
SONAR	28	int	1=Enable; 0=disable all the sonar	–	1.0
STOP	29	none	Stops robot (motors remain enabled)	–	1.0
DIGOUT	30	int	Msbits is a byte mask that selects output port(s) for changes; lsbits set (1) or reset (0) the selected port.	4.2	1.2
VEL2	32	signed int	Independent wheel velocities; lsb=right wheel; msb=left wheel; PSOS is in ±4mm/sec; P2OS in ±2cm/sec increments	4.1	1.0
GRIPPER	33	int	Pioneer 1 and Pioneer 2 Gripper server command. See the Pioneer Gripper manuals for details.	4.0	1.3
ADSEL	35	int	Select the A/D port number for analog value in SIP. Selected port reported in SIP Timer value.	–	1.2
GRIPPERVAL	36	int	Pioneer 2 gripper server value. See P2 Gripper Manual for details.	–	
GRIPREQUEST	37	none	Request 1 or continuous stream (>1), or tell to stop sending (0) Gripper SIPs	–	1.E

IOREQUEST	40	none	Request 1 or a continuous stream (>1) or tell to stop sending (0) IO SIPs	–	1.E
PTUPOS	41	int	msb is the port number (1-4) and lsb is the pulse width in 100µsec units PSOS, 10µsec units P2OS. Version 1.J uses RC-servo 40ms duty cycle.	4.5	1.2 - 1.J
TTY2	42	string	Send string argument to serial device connected to AUX port on microcontroller	4.2	1.0
GETAUX	43	int	Request to retrieve 1-200 bytes from the aux serial channel; 0 flushes the aux serial input buffer.	–	1.4
BUMP_STALL	44	int	Stop and register a stall if front (1), rear (2) or either (3) bump-ring contacted. Off (default) is 0.	–	1.5
TCM2	45	int	TCM2 Module commands; see *TCM2 Manual* for details.	–	1.6
DOCK	46	int	Default is 0=OFF; 1=enable docking signals; 2=enable docking signals and stop the robot when docking power sensed.	–	1.C
JOYDRIVE	47	int	Default is O=OFF; 1=allow joystick drive from hardware port	–	1.G
E_STOP	55	none	Emergency stop, overrides deceleration	–	1.8
E_STALL	56	int	Emergency stop button causes stall on PeopleBot *only*	–	1.E
STEP	64	none	Single-step mode (simulator only)	3.x	1.0
ARM	70 – 80	–	Please consult the Pioneer 2 Arm Manual for details.	–	1.H
ROTKP	82	int	Change rotation's proportional drive factor	–	1.J
ROTKV	83	int	Change rotation's differential drive factor	–	1.J
ROTKI	84	int	Change rotation's integral drive factor	–	1.J
TRANSKP	85	int	Change translation proportional drive factor	–	
TRANSKV	86	int	Change translation differential drive factor	–	
TRANSKI	87	int	Change translation integral drive factor	–	
PLAYLIST	90	int	Must be 0; request AmigoBot sound playlist	–	1.E

The client must send a command packet at least once every watchdog (default is two) seconds (see next chapter). Otherwise, the robot's onboard drives automatically stop, but will resume again on receipt of a client packet. To maintain the watchdog, Saphira, for instance, sends the action-less PULSE command in the absence of other client commands.

A 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 four types of P2OS client command arguments: none, unsigned integers two bytes long, signed integers two bytes long, and NULL-terminated strings consisting of as many as 196 characters.

The byte order, where applicable, 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.

Saphira Client Command Support

Saphira, as well as ARIA, all support 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. Not all P2OS command names are supported in Saphira. See the *Saphira Software Manual*.

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

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

Or to have your Pioneer 2 play a tune (albeit rather tinny), type:

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

Programming P2OS

You may create your own P2OS interface, or use the convenience functions available through the various applications-development software that come with your *Activ*Media robot.

Establishing a Client-Server Connection—SYNC

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

When first started, P2OS is in a special wait ("noconn") state, listening for communication packets to establish a client-server connection. To establish a connection, the client application must send a series of three synchronization packets through the Host communication port, containing SYNC0, SYNC1, and SYNC2 command bytes, in succession, and retrieve the server responses.

When in wait mode, P2OS echoes packets back to the client. The client should listen for the returned packets and only issue the next synchronization packet after it has received the appropriate echo.

Autoconfiguration

P2OS automatically sends robot-configuration information back to the client in the last synchronization packet (SYNC2) echoed when establishing a connection. After the SYNC2 byte, there are three NULL-terminated strings that comprise the robot's name, class, and subclass. You may uniquely name your Pioneer 2 with the p2oscf configuration tool we provide with the robot. The class and subclass also are parameters stored in the robot's FLASH, but are constants normally set at the factory and not changed thereafter. (See next chapter 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. ARIA, for instance, loads the robot's related parameters file found in a special robot params directory.

Opening the Servers—OPEN

Once you've established a communication link, the client should send the OPEN command to the server, which causes the *Activ*Media robot controller to perform a few

housekeeping functions, start its sonar and motor controllers (among other things), listen for client commands, and begin transmitting server information to the client.

Note that once connected, Pioneer 2's and PeopleBot's motors are disabled, regardless of their state when last connected. After starting a connection, you must either enable the motors manually (white MOTORS button) or send the P2OS motors ENABLE command with the argument 1; `sfRobotComInt(4,1)`, for example.[19]

Keeping the Beat—PULSE

A P2OS safety watchdog expects that the controller receives at least one communication packet from the client every watchdog seconds (default is two). Otherwise, it assumes the client-server connection is broken and stops the robot.

It's good practice to have the client send a PULSE command just after opening the P2OS servers. And if your client application will be otherwise distracted for some time, periodically issue the PULSE command to let P2OS 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 and heading setpoints.

Closing the Connection—CLOSE

To close the client-server connection, disabling the motors and sonar, and resetting P2OS to its wait state, simply issue the client CLOSE command.

Movement Commands

The P2OS motor-control servers accept several different motion commands of two mutually exclusive types: either direct wheel-velocities or translational/rotational motor controls. The robot servers automatically abandon any P2OS translational or rotational setpoints and switch to direct wheel-velocity control mode when they receive a VEL2 command. Any other motion command makes P2OS abandon direct wheel-velocity control.

For example, if P2OS is in direct-wheel velocity (VEL2) mode and is given a HEAD command, it disables that direct-wheel velocity mode and starts controlling the heading

Table 7. P2OS movement commands

Rotation	
HEAD	Absolute heading
DHEAD, DCHEAD	Differential heading from control point
ROTATE	Rotational speed
SETRA	Rotational (de)acceleration to achieve setpoint
SETRV	Sets maximum rotational velocity and is velocity used for Colbert turn and turnto command speeds.
Translation	
VEL	Forward/back velocity
MOVE	Forward/back distance
SETA	Translation (de)acceleration to achieve setpoint
SETV	Sets maximum translational velocity and is used for Colbert move command speed.

[19] Alternatively, disable the motors with the ENABLE command argument 0.

and velocity of the robot.

When in translation/ rotation (TR) motion control (recommended), separate translation and rotation servers work independently to achieve the specified forward/ reverse speed and heading of the robot. P2OS will try to make the robot achieve the desired translational velocity and rotate to the desired heading as soon as the commands are received. Acceleration and deceleration managers, which default values you may prepare and also change on-the-fly (SETRA and SETA), operate independently for translation and rotation, too.

Pioneer in Motion

When P2OS receives a translation, rotation, or direct-wheel velocity command, it accelerates the robot at the SETA (both TR and VEL2 modes)and SETRA (TR mode only) rates you program, or at the rates preset in the P2OS configuration parameters. Rotational headings and translational setpoints are achieved by a trapezoidal velocity function. This function is recomputed each time a new motion command is received, making on-the-fly changes possible.

Rotational and translational velocities are limited to the maximum values set in the P2OS parameters or which you may reset with the SETVEL and SETRV commands. Maximums take affect on subsequent commands, not previously established velocity or heading setpoints.

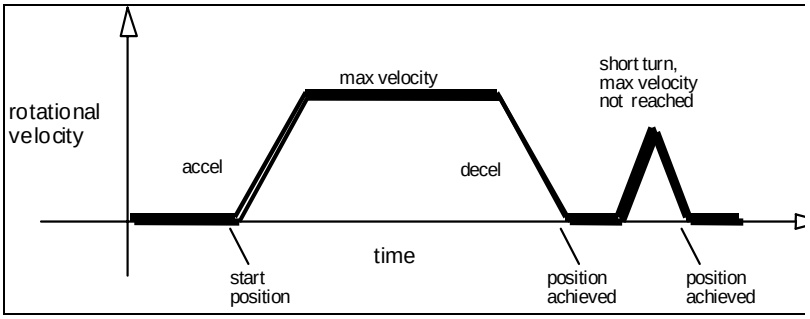

Figure 20. Pioneer's trapezoidal turning velocity profile

Note that with P2OS version 1.8 and later, you may override deceleration with the emergency stop (E_STOP) command number 55 or with the Emergency STOP button enabled on the Performance PeopleBot (E_STALL command #56). Accordingly, the robot brakes to zero translational and rotational velocities with very high deceleration and remains stopped until it receives a subsequent translational or rotational velocity command from the client. (See *E_STOP and E_STALL* later in this chapter.)

PID Controls

The Pioneer drive servers use a common Proportional-Integral-Derivative (PID) control system to adjust the PWM pulse width at the drivers and subsequent power to the motors. The motor-duty cycle is 200 µsec; pulse-width is proportional 0-500.

The P2OS drive servers recalculate and adjust your robot's trajectory and speed every 10ms (P2OS v1.I and earlier) or every 5ms (P2OS1.J) based on feedback from the wheel encoders. Note that the latter time cycle leads to enhanced performance of the robot.

The default PID values for both translation and rotation are stored in FLASH. Beginning with P2OS1.J, you may change those values on the fly with the client commands 84-87.

Position Integration

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

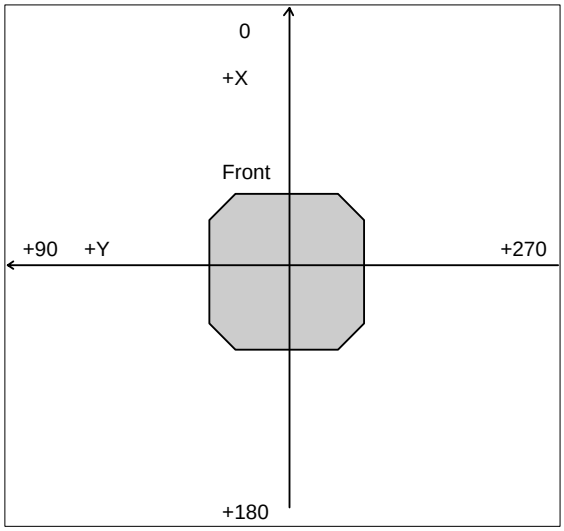

Figure 21. Internal coordinate system for P2OS

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 or two revolutions, 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 in a global map.

The orientation commands HEAD and DHEAD turn the robot with respect to its internal dead-reckoned angle. 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 359. 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).

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

Sonar

When OPENed (see Opening the Servers—OPEN above), P2OS automatically begins firing your robot's sonar arrays in the predefined default sequence: 1–8, 9–16, 17–24, and 25–32, one sonar per array simultaneously.[20] P2OS also begins sending the sonar-ranging results to the client via the server-information packet. Use the SONAR command to enable (argument is "1") or disable (argument is "0") the sonar pinging. This is an all or nothing command; you cannot disable a single array this way.

For example:

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

Use the sonar POLLING command to change the polling sequence of an array. Its argument is a null-terminated string of up to 15 sonar numbers. Front sonar are numbered \001 through \010 (octal), and the rear sonar are \011 through \100, for

[20] See the next chapter for details on how to enable sonar arrays with p2oscf.

example. You may repeat a sonar number and have it ping more than once per sequence.

For example:

```
sfRobotComStrn(sfCOMPOLLING,"\001\002\001\002",4); /* ping front left only */
```

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

BUMP_STALL

Because the robots are programmed to climb, it can take up to a second or more to detect object collisions and subsequent motor stalls. With P2OS version 1.5, we introduced a responsive way to stop your robot within a few milliseconds upon collision with an obstacle: BUMP_STALL. With the robot equipped with forward and/or rear bumpers, you can now program P2OS to immediately stop the robot and notify the client of a stall if any one of the forward or rear bump sensors get triggered.

NOTICE

BUMPSTALL requires supporting bump ring accessories.

Send the BUMP_STALL command (#44) with a integer argument of 0 to disable the behavior (default on startup). Use the argument value 1 to enable BUMP_STALL only when a forward bump sensor gets triggered; 2 for rear-only BUMP_STALL; or 3 for both rear and forward bump ring-activated stalls.

For example, to enable BUMP_STALL when any one of the five contact sensors on a forward bump ring gets triggered:

```
sfRobotComInt(44,1);
```

E_STOP and E_STALL

In an emergency, your client may want the robot to stop quickly, not subject to normal deceleration. In that case, send the E_STOP command (#55).

Like BUMP_STALL, use P2OS' built-in E_STALL feature (version 1_E or later) to simulate a stall when someone presses the Performance PeopleBot's Emergency STOP button. An integrated switch in the button causes the User I/O analog port #4 on the microcontroller to go high (+5VDC) and thereby notify P2OS of the emergency condition. Accordingly, when enabled, E_STALL will disable the motors in P2OS and notify your Saphira or other software client that the robot has stalled (stall bytes in the standard SIP), thereby avoiding the motor rush that may occur when you reactivate the emergency STOP button.

Since it is normally disabled, you must enable E_STALL through software at least once after a reset or power cycle of the Performance PeopleBot. To enable E_STALL, send the E_STALL P2OS command (#56) with argument = 1; 0 disables E_STALL.

For example, with Saphira:

```
sfRobotComInt(56,1);  /* Enable E_STALL */
sfRobotComInt(56,0);  /* Disable E_STALL */
```

You may also monitor the state of the Emergency STOP button with or without E_STALL as an analog value reported in the standard SIP. You must first request that the value be included in the standard packet since it is not the default analog port. (The current analog port# is encoded in the timer entry.)

For example,

```
sfRobotComInt (35,4);  /* select AN4 to b in the standard SIP */
…
later…
if (sfRobot.timer & 0xff = = 4 && sfRobot.analog > 200)
  /* E-Stop button pressed and engaged */
else
  /* E-stop not engaged. */
```

Extended Packets

Several different, additional types of server information packets come with P2OS to better support the *Activ*Media Robotics community. On request from the client by a related P2OS command, the P2OS server packages and sends one or a continuous stream (one per cycle) of information packets back to the client over the Host serial communication line. Extended packets get intermingled with the standard SIP that P2OS sends to your client every 100 or 50 milliseconds, depending on the SinfoCycle parameter.[21]

Packet Processing

Like the standard SIP, all P2OS server information packets get encapsulated with a header (0xFA, 0xFB), byte count, packet type byte, and trailing checksum. It is up to the client to parse the packets, sorted by type for content. Please consult the respective client application programming manuals for details.

Saphira, for example, has an internal parser for the PSOS/P2OS packet type 0x3S (S=0-2; aka "standard" SIP), but not for the P2OS extended packets we describe in this section. However, you are able to program and add packet handlers to Saphira: either as a special plugin program as found at the http://robots.activmedia.com support website (look for details in the contributed directory) or directly in your own Saphira client. A sample packet processor is included with the Saphira distribution—packet.c in the apps/ directory, and see the AUX serial SIP processing code in the following section.

CONFIGpac and CONFIG Command

[21] You may have to adjust the Host port serial baud rate to accommodate the additional communications traffic.

Send the CONFIG command #18 with or without an argument to have P2OS send back a special server information packet containing the robot's operational parameters. The CONFIGpac SIP packet type is 32 (0x20). The next chapter gives details about the configuration packet data.

```
/*  Send commands and retrieve responses via the P2OS AUX serial port */

#define SERAUXpac 0xB0
int serbytes, inbufptr;
char inbuf[200];
...
/* Replace Saphira's default packet processor with
  our own that can detect and parse SERAUXpac packets */
void myStartupFn(void)
{
  sfInitProcess(myPackets, "GetAuxPkt");
  sfRemoveTask("packets");      /* default packet processing */
  ...
}

/* SERAUXpac packet processor */
void myPackets(void)
{
  char packet_type;
  if (sfIsConnected)
  {
    while(sfHaveClientPacket())
    {
      packet_type = sfReadClientByte();
      if (packet_type == SERAUXpac)
        {
      /* get aux serbytes into inbuf */
          while (inbufptr++ < serbytes)
              inbuf[inbufptr] = sfReadClientByte();
        }
    /* If not SERAUXpac, send along to default */
      else
        sfProcessClientPacket(packet_type);
    }
}

 void main(int argc, char **argv)
 {
/* Initialize client and connect with robot */
...
  sfRobotComInt(43,0);    /* flush the aux buffer */
  sfRobotComStrn(42,"fictitious command",18); /* send command to AUX device */
  inbufptr = 0;
  serbytes = 10;   /* number of response bytes to retrieve */
  sfRobotComInt(43,serbytes); /* Expect a 10 byte response */
...
}
```

SERAUXpac and GETAUX

P2OS 1.4 implemented two-way communications through the AUX serial port on the microcontroller. And with those earlier versions of P2OS, you are able to send a string of data (a command or otherwise) from the Host port-connected client to an AUX port-connected serial device (TTY2 command number 42), but you could not get data back from that device.

Now P2OS maintains a circular buffer for incoming serial data from the AUX port and sends successive portions of the buffer to your client via the Host port depending on the

number of bytes you request via the GETAUX command—up to 200 bytes at a time. P2OS waits to collect the requested number of incoming AUX-port serial bytes before sending the GETAUX packet to the client. Use the GETAUX command with a zero argument to flush the P2OS circular buffer and reset its pointers.

In the code example above, a fictitious command gets sent to a serial device attached to the microcontroller's AUX port and retrieves a 10-byte response from that device (note that ellipses mark the many portions of a complete client that have been deleted for brevity).

ENCODERpac and ENCODER Command

Issue the ENCODER command #19 with an argument of 1 for a single or with an argument of 2 or more for a continuous stream of ENCODERpac (type 144; 0x90) SIPs. Discontinue the packets by sending the ENCODER command #19 with the argument 0.

Table 8. P2OS Encoder Server Information Packet Contents

Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 (checksum)
Left Encoder	integer	Least significant, most significant portion of the
	integer	current 4-byte raw encoder count from the left drive wheel
Right Encoder	integer	Least significant, most significant portion of the
	integer	current 4-byte raw encoder count from the left drive wheel
Checksum	integer	Checksum for packet integrity

GRIPPERpac and GRIPREQUEST

To ameliorate a few shortcoming when we transferred connection and control of the P2 Gripper accessory from User I/O to the General I/O bus, P2OS 1.E introduced and we improved the new GRIPPERpac (type=224; 0xE0) packet type and related GRIPREQUEST P2OS command #37 in P2OS 1_F.

Table 9. Gripper packet contents (P2OS 1_F and later)

Header	integer	Exactly 0xFA, 0xFB
Byte Count	byte	Number of data bytes + 2 (checksum)
Type	byte	Packet type = 0xE0
HasGripper	byte	Gripper type: 0=none; 1=User; 2=General I/O
Grip_state	byte	See Table 12 and *P2 Gripper Manual*
Grasp_Time	byte	See *P2 Gripper Manual*
Checksum	integer	Computed checksum

Normally disabled, your client program must request one (command argument 1) or a continuous stream (command argument > 1) of Gripper packets. Send GRIPREQUEST with the argument value 0 to stop continuous packets.

PLAYLISTpac and PLAYLIST Command

Unlike its Pioneer and PeopleBot cousins, the AmigoBot mobile robot has onboard sound reproduction hardware and software that includes a playlist of contents. To support the *Activ*Media Robotics Basic Suite that includes all *Activ*Media's robots, we've added the PLAYLISTpac (type = 208; 0xD0) and PLAYLIST request command #91 to P2OS 1.E. We document the command and packet here for completeness, but they have no effect on the operation or performance of your Pioneer 2 or PeopleBot Mobile Robot.

Table 10. GRIPPERpac state byte

bit	Function	State
0	Grip limit	paddles fully open when 0; otherwise between or closed
1	Lift limit	Lift fully up or down when 0; otherwise in between
2	Outer breakbeam	obstructed when 0; nothing in between when 1
3	Inner breakbeam	obstructed when 0; nothing in between when 1
4	Left paddle	grasping when 0
5	Right paddle	grasping when 0
6	Lift	moving when 1
7	Gripper	moving when 1

The AmigoBot Sounds playlist consists of a series of one to 255 24-byte long sound references, followed by individual sound data. Sound references may be NULL or redundant.

Sound references consist of a 16-byte sound name followed by two long integers, which specify the sound data position and length in the playlist. The PLAYLIST P2OS command #91 with any or no argument responds with a packet of 25 NULL bytes, telling the client that your P2OS-based robot does not have any onboard sounds.

TCM2pac and TCM2 Command

The TCM2 accessory is an integrated inclinometer, magnetometer, thermometer, and compass that attaches to the AUX serial port of the P2OS microcontroller. When attached and enabled, special TCM2 compass servers read and report the heading as the compass byte in the standard SIP. Use the TCM2 command to request additional information from the device in the form of the TCM2pac. See the *TCM2 Manual* and supporting software that accompanies the device for details.

Input / Output (I/O)

Your Pioneer 2 or PeopleBot comes with a number of I/O ports that you may use for sensor and other accessories and custom attachments. See Appendix A for port locations on the microcontroller. The various I/O states and readings appear in the standard SIP and may be manipulated with P2OS client commands. P2OS 1.E also introduced a new SIP for convenient access to your Pioneer 2's I/O (see a subsequent section for details).

DIGIN, TIMER, and ADSEL

The states of the eight digital input ports, native to the controller, get mapped as a single byte, whose value is continuously updated and reported in each standard SIP (Table 4). For example, examine Saphira's `sfRobot` structure value `digin`:

```
sfSMessage("Digin is %i",sfRobot.digin);
```

When not connected, the digital input port values may vary and change without warning.

The digital input ports 4-7 (ID4-7) also may be used as A/D input ports (see below). Use the ADSEL client command to select the A/D port that is to appear in the P2OS SIP analog value. The default port is #5, the dedicated A/D port on the system. The P2OS SIP reports the currently selected analog input port number.

In the following example, the first Saphira statement queries for the current analog port and its A/D value. The second Saphira command changes the selected port which value gets reported through the last Saphira command in the example:

```
sfSMessage("Port# %  reads %i",sfRobot.timer, sfRobot.analog);
Port# 5 reads 33
sfRobotComInt(35,2);
sfSMessage("Port# %  reads %i",sfRobot.timer, sfRobot.analog);
Port# 2 reads 224
```

DIGOUT and PSUPOS

The eight digital output ports on the P2OS controller's User I/O connector are both reported in the standard SIP (digout) and controllable with the P2OS commands DIGOUT and PTUPOS. Electrically, the ports are digital high (1) at ~5 VDC (V_{cc}) and low (0) at ~0 VDC (GND).

For example, to read the state of the digital output ports with Saphira:

```
sfSMessage("Digout is %i",sfRobot.digout);
Digout is 240
```

Use the P2OS DIGOUT command to select and change the state of one or more of the output ports at a time. DIGOUT takes a 2 byte (unsigned integer) argument. The first byte is a mask whose bit pattern selects (1) or ignores (0) the state of the corresponding bit in the second byte to set (1) or unset (0) the digital output port.

For example, you might use Saphira to set digital output ports 1 and 3, reset port 4, and leave all the rest alone (retain original state):

```
sfRobotComInt2Bytes(30, 0x19, 0x09);
```

The digital output ports OD0-4 may also be used for pulse-width-modulated (PWM) control of accessories, such as DC motor speed or RC-servo motor position control. Use the PTUPOS command to select a port (msbyte) and specify its pulse-width (lsbyte) in 10 µsecond units.

For example, to have a repeating one millisecond pulse appear on PWM port #1 (pin 1 on the User I/O connector):

```
sfRobotCom2Bytes(41,1,100);
```

You've got to disable a running PWM port to use it as a digital output port:

```
sfRobotCom2Bytes(41,1,0);
```

In P2OS versions 1.J and later, the duty cycle is 20ms, with a 0-2.55ms pulse corresponding to common RC-servo applications.

IOpac and IOREQUEST

P2OS 1.E introduced a new server information packet in which your P2OS-based robot controller reports all of its I/O-connected sensor input and output values in a single cycle.

Your client software must explicitly request IO packets; they normally are disabled. Use the P2OS IOREQUEST command #40 with an argument value of 0, 1, or 2. The argument 1 requests a single packet to be sent within the next cycle. The request argument value 2 tells P2OS to send IO packets continuously, at approximately one per cycle, depending on serial port speed and other pending SIPs. Use the IOREQUEST argument value 0 to stop continuous packets.

The common IOpac contains three digital input bytes, User I/O digin and two bumper state bytes, front followed by rear; one digital output byte (digout); and five analog-to-digital values corresponding to the ports AN1-5. Unlike the standard SIP, which contains

a byte value for the selected analog port, analog values in the IOpac are integers, with resolution to 12 bits. If either bumper is not installed, its reported value can vary.

Table 11. IOpac packet contents

Header	integer	Exactly 0xFA, 0xFB
Count	byte	Number of data bytes + 2
Type	byte	Packet type = 0xF0
N_i	byte	Number of digtal input bytes
Digin	N_i bytes	Digital input bytes
N_o	byte	Number of digital output bytes
Digout	N_o bytes	Digital output bytes
N_a	byte	Number of A/D values
A/D	N_a ints	A/D values 1-Na; 0-2047 (12-bit resolution) = 0-5 VDC
Checksum	integer	Computed checksum

Pioneer 2 Arm-related SIPs and Commands

Please consult the *Pioneer 2 Arm Manual* for details.

Performance PeopleBot IRs

Two breakbeam IR sensors sense objects which intrude into the Performance PeopleBot's profile, but which may not be otherwise detected by its sonar or tabletop IR sensors. The left and right IR breakbeams are connected to the User I/O digital input ports ID2 and ID3, respectively. Normally high (1), the respective input port goes low (0) when an object breaks the IR transmitter's beam to its companion receiver.

The left and right tabletop IR detectors on the Performance PeopleBot are connected to the ID0 and ID1 digital input ports of the User I/O, respectively. Normally high (1), the digital input port goes low (0) when its respective tabletop IR sensor detects an object within its range of operation.

The tabletop and breakbeam IR sensor states appear in the standard P2OS Server Information Packet. Accordingly, to react to the state of an breakbeam or tabletop IR sensor in a Saphira client, for example, test the state of its respective bit:

```
#define LFT_TTIR 0x01
#define RGT_TTIR 0x02
#define LFT_BREAKBEAM 0x04
#define RGT_BREAKBEAM 0x08

if (sfRobot.digin &  LFT_BREAKBEAM = = 0)
   /* Something has triggered the left column breakbeam … */
```

Chapter 7 Updating & Reconfiguring P2OS

The P2OS server software and a set of operating parameters get stored in your robot's microcontroller's FLASH ROM. With special download and configuration utilities, you may change and update the FLASH memory image without physically replacing any hardware.

Where to Get P2OS Software

Your *Activ*Media robot 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

Software utilities for downloading a new P2OS (`p2osdl`) and for changing your robot's configuration parameters (`p2oscf`) come with your robot on CD-ROM and in the onboard computer. Fresh copies and updates also may be downloaded from the support website bundled with the P2OS image. Use the version that matches your host computer's environment. For example, use p2osV_r.tgz for RedHat© Linux or p2osV_r.EXE with Microsoft Windows© 9x/ME systems ("V" is the version number and "r" is the revision number; p2os1_J, for example).

The p2osV_r.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:

```
% tar –zxvf p2os1_J.tgz
```

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

Updating P2OS

Use the `p2osdl(.exe)` program that comes with each distribution to download a fresh copy of PSOS to your robot's microcontroller's FLASH ROM.[22]

> An onboard radio modem typically interferes
> with P2OS download and reconfiguration.
> You may need to remove its connector from the Host serial port
> inside the robot on the microcontroller.

Step 1. Serial Connection from Computer to Robot

Connect your Pioneer 2 or PeopleBot to your host computer through their respective serial ports. If you have an onboard PC, use it to perform the download or switch its power OFF. Otherwise, use a direct ("straight-through") serial tether from your PC to the 9-pin SERIAL connector on your robot's Console. If you have an onboard radio modem, you may have to remove its serial connector from the Host serial port on the microcontroller by reaching in with your fingers through the rectangular access port on the Deck. See Appendix A for port location. Sorry about the inconvenience—the modem interferes with the direct connection.

[22] The p2osdl program may update your robot's configuration parameter setting, so be sure to use the program that matches your current P2OS version.

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 your robot to enable FLASH writes.

Step 3: Put Microcontroller into Download Mode

Start up or reset your *Activ*Media robot. After it has finished initializing, place it in download mode:

1. Press and hold the white MOTORS button.
2. Press and release the red RESET button.
3. Continue holding the MOTORS button for three or more seconds.
4. Release the MOTORS button.

The robot should not reset. If it does, you probably didn't hold the MOTORS button down long enough. Try again. When successfully in BOOT mode, 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 with the following arguments:

```
% p2osdl p2os1_J.hex <comm-port>
```

The pathname for the P2OS HEX image file is required and may be a direct or relative path. The file p2os1_J.hex, for example, is the current P2OS version 1.H hex-encoded image.

The comm-port argument is optional. It lets you specify the serial communication port that connects p2osdl with the P2OS microcontroller. For Linux/UNIX systems, the default is /dev/ttyS0. To communicate through /dev/ttyS3, for example, use:

```
% p2osdl p2os1_J.hex /dev/ttyS3
```

For Win32 systems, you must pass the required p2osV_r.hex file to p2osdl.exe, so you cannot simply double-click the program icon. Instead, use your mouse to successfully launch the program by dragging and dropping the .hex icon (p2os1_H.hex, for example) onto the psosdl.exe icon. But this works only if you use the default COM1 serial port and the current directory is set to the P2OS one.

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. You might also Run... the program from the Start menu.

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. Since FLASH gets erased at the beginning of a download, an unsuccessful session may leave the controller empty and apparently broken, although it is not. Repeat downloads until successfully completed.

Configuring P2OS Operating Parameters

The program p2oscf(.exe) is the way you view and change your Pioneer 2's identity and operating parameters.

Limited reconfigurations.
FLASH ROM in the controller is guaranteed for only 100 erase cycles.

Steps 1–3: Preparing for Configuration

Prepare for changing your robot's configuration parameters identically to Steps 1, 2 and 3 for updating P2OS described above.

Step 4: Run p2oscf

As with p2osdl, you will find p2oscf(.exe) in the p2os/ directory of your PSOS distribution. The program accepts optional arguments (Table 12); the argument-less default startup is to connect with your robot's P2OS microcontroller through your PC's serial port /dev/ttyS0 in Linux/UNIX systems or COM1 with Win32. With the default, you may double-click to launch psoscf.exe directly from your desktop, rather than executing the program from the MS-DOS Prompt window.

Table 13. p2oscf startup options

Argument	Description
-h	List p2oscf argument options and quit
-b	Batch mode; series configuration command must follow. Configuration changes made to FLASH parameters
-n	Operate p2oscf without connecting with the microcontroller. Useful for editing parameter files saved to disk.
-p	Set serial port; serial port name must follow immediately after argument; /dev/ttyS1 or COM3, for example.
-s	Automatically save changes to disk file named in path immediately following argument.

Step 5: Changing Configuration Parameters

On start up (after power cycle or RESET), P2OS reads a set of operating parameters from its FLASH memory and uses these values if and until you override them with explicit P2OS commands. For instance, a default maximum velocity is stored in the TransVelMax parameter. After establishing a client-server connection, you may send a SETVEL command which changes that default maximum velocity.

Normally when it starts, p2oscf automatically retrieves the current identifying and operating parameters from your Pioneer 2 or PeopleBot robot. Some of the parameters, "Constants", are not to be changed. The others, "Variables", are the identifying and operating parameters that you may edit with p2oscf.

With p2oscf, you edit a temporary copy of the parameters list. Your changes are not changed in your robot's FLASH memory until you choose to explicitly "save" them. Even then, p2oscf will write to FLASH only if you have changed some parameter value. Writes to the C166 FLASH are guaranteed for only 100 cycles, so we caution that you reconfigure/update your P2OS microcontroller only when necessary.

Step 6: Save Your Work

Use the save command to save your configuration changes to FLASH or to a disk file. We strongly recommend that you save each of your robots' parameter values to disk for later retrieval should your microcontroller get damaged or it's FLASH inadvertently erased. Default parameter files come with each P2OS distribution, but it is tedious to reconstruct an individual robot's unique configuration.

Editing P2OS Parameters

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

Table 14. 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 on the robot.
a or arm	Display current, edited P2OS-based Arm-related values. May be different than those stored in FLASH on the robot.
r or **restore** <pathname>	Restores edited (p2oscf) variables to values currently stored in FLASH or from file, if argument included.
save <pathname>	Saves current edited values to FLASH 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.

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

```
> watchdog 4000
```

or

```
> watchdog 0xfa0
```

The critical operating parameters for your robot are revcount, encoder, 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 FLASH. And, too, you may over-ride many of these parameters with respective P2OS commands from the client.

Saving and Restoring

The p2oscf program lets you save and restore whole configuration sets from disk-stored files. This lets you easily configure your robot for various different environments, as well as maintain a record of your original and test parameters.

To save your current configuration to a disk file, get connected through p2oscf as described earlier. This loads the current operating parameters into the configuration editor. Then simply save your current configuration to disk. For example:

```
save C:\p2os\myP2DX
```

The command does not change the working configuration in any way.

Use the p2oscf restore command to not only retrieve the current operating parameters from the robot, but to load a saved parameters files from disk. The P2OS distributions come with common configuration files for the all *Activ*Media robot models, including p2de for the Pioneer 2-DXe, p2at for the Pioneer 2-AT, and so on.

You may edit the file-restored parameters just as you edit those retrieved from the robot. And you may save those edited parameters over the same file or a different one, using the p2oscf save command.

Note that file-restored configuration parameters are not necessarily the same as those stored in the robot's FLASH. You must save them there separately. The following sequence of p2oscf commands illustrates the restore/save process. Notice that the name stored on the robot doesn't change until the change is saved to FLASH:

```
command> name
Your robot's name is Son_of_Flakey
command>restore p2dx
command> name
Your robot's name is ActivMedia_P2DX
command> restore
command> name
Your robot's name is Son_of_Flakey
command> restore p2dx
command> save
command> restore
command> name
Your robot's name is ActivMedia_P2DX
```

Also notice that the last restore command is redundant. Immediately after a save, p2oscf 's and the robot's parameters are identical.

Arm Parameters

P2OS version 1.H introduced a comprehensive set of servers for managing the ActivMedia Robotics Pioneer 2 Arm accessory. Please consult the *Pioneer 2 Arm Manual* for details about its many operating parameters that are store in FLASH and modified through p2oscf.

Table 15. Configuration parameters (currently P2OS 1_J) with values for Pioneer 2-DX

KEYWORD	Type	Default	Description
CONSTANTS			*Should not be changed using p2oscf*
Type	str	Pioneer	Identifies the robot type.
Subtype	str	P2DX	Identifies the *Activ*Media robot model.
Serial	str	factory	Serial number for the robot.
FourMotors	byte	0	Is '1' for the four-motor Pioneer 2-AT
RotVelTop	int	360	Maximum rotational velocity; deg/sec
TransVelTop	int	2200	Maximum translation speed; mm/sec
RotAccTop	int	600	Maximum rotation (de)acceleration; deg/sec^2
TransAccTop	int	4000	Maximum translational (de)acceleration; mm/sec^2
PwmMax	int	500	Maximum motor PWM period (500=fully on).
Encoder	int	76	Encoder ticks/mm: $\frac{\text{(ticks/rev x gear-ratio)}}{\text{(wheel\_diameter x PI)}}$
VARIABLES			*Parameters that you may change with p2oscf*
Name	str	not_set	Unique name for your robot. Maximum of 20 characters, no spaces.
SInfoCycle	byte	0	Server information packet communication cycle time: 0=classic 100 ms; 1=new 50 ms cycle time
HostBaud	byte	0	Baud rate for host (client) serial port connection: 0=9600, 1=19200, 2=38400 bps.
AuxBaud	byte	2	Baud rate for AUX serial port; see HostBaud
HasGripper	int	0	1 if P2 Gripper installed into User I/O; 2 if installed in General I/O; otherwise 0
FrontSonar	int	1	1 if front sonar array installed; otherwise 0
RearSonar	byte	0	1 if rear sonar array installed; otherwise 0
LowBattery	int	110	In 1/10 volts; microcontroller alarm activated when battery charge falls below this value.
RevCount	int	20000	The number of differential encoder ticks for a 360 degree revolution of the robot.
WatchDog	int	2000	Ms time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection.
P2Mpacs	byte	0	1 enables alternative SIP. Otherwise 0.
StallVal	int	200	Maximum PWM before stall. If > PwmMax, never.
StallCount	int	100	Ms time after a stall for recovery. Motors not engaged during this time.
Compass	Int	0	Compass type: 0=none; 1=V2XG; 2=TCM2
CompX	int	0	V2XG Compass calibration X-offset
CompY	int	0	V2XG Compass calibration Y-offset
RotVelMax	int	200	Max rotational speed; deg/sec.
TransVelMax	int	300	Max translational speed; mm/sec.
RotAcc	int	50	Rotational acceleration; deg/sec^2
RotDecel	int	50	Rotational deceleration; deg/sec^2
RotKp	int	30	Proportional PID for rotation
RotKv	int	200	Differential PID for rotation
RotKi	int	0	Integral PID for rotation
TransAcc	int	300	Translational acceleration; mm/sec^2
TransDecel	int	300	Translational deceleration; mm/sec^2
TransKp	int	15	Proportional PID for translation
TransKv	int	800	Differential PID for translation
TransKi	int	4	Integral PID for translation
JoyVelMax	int	600	Joydrive maximum translation velocity
JoyRVelMax	int	125	Joydrive maximum rotational velocity

PID Parameters

The P2OS configuration parameters include settings for the PID motors controls for translation and rotation of the robot. The translation PID values also apply to independent wheel velocities. The default values shown in the Table are for a moderately loaded robot. Experiment with different values to improve the performance of your robot in its current environment.

Note that with P2OS version 1.J and later, the PID values have changed from earlier versions due to changes in the drive servers. Overall drive performance has improved with the latter version and is much less sensitive individual PID values. Consequently, we recommend the listed values for all Pioneer 2 and PeopleBot robots, whereas earlier P2OS versions required much more tuning of the parameters depending on robot type and configuration/payload.

The P term value Kp increases the overall gain of the system by amplifying the position error. Large gains will have a tendency to overshoot the velocity goal; small gains will limit the overshoot but cause the system to become sluggish. We've found that a fully loaded robot works best with a Kp setting of around 15 to 20, whereas lightly loaded robot may work best with Kp in the range of 20 to 30.

The D term Kv provides a PID gain factor that is proportional to the output velocity. It has the greatest effect on system damping and minimizing oscillations within the drive system. The term usually is the first to be adjusted if you encounter unsatisfactory drive response. Typically, we find Kv to work best in the range of 600 to 800 for lightly to heavily loaded robots, respectively.

The I Term Ki moderates any steady state errors thereby limiting velocity fluctuations during the course of a move. At rest, your robot will seek to "zero out" any command position error. Too large of a Ki factor will cause an excessive windup of the motor when the load changes, such as when climbing over a bump or accelerating to a new speed. Consequently, we typically use a minimum value for Ki in the range of 0 to 10 for lightly to heavily loaded robots respectively.

Encoder and Revcount

P2OS uses the encoder and revcount parameters to convert your platform-independent speed and rotation commands—typically expressed in millimeters or degrees per second, respectively—into platform-dependent units. In previous versions, the encoder value was contained in the P2OS code. With improvements to the motion control servers in P2OS and to support the increasing variety of *Activ*Media robot configurations, the encoder value is now included as a configuration parameter, although unlike revcount, its value is a constant.

The encoder value is the number of encoder pulses ("ticks") per millimeter of wheel rotation. The encoder value is, of course, dependent upon the wheel encoder's resolution, the motor-to-wheel gear ratio, and the wheel's diameter.

The revcount value is the number of encoder ticks for one full revolution of the robot. It depends on a number of factors, principally the length of the wheel base, which may change due to payload, tire wear, operating surface, and so on. Accordingly, we provide a calibration tool that lets you determine revcount for your individual robot configuration and operating conditions.

Table 16. Some platform-dependent robot parameter values

	Model

Parameter	DX	DXe	AT	CE	PB V1	Performance PB
Encoder ticks per rev	500	500	500	100	500	500
Gear ratio	19.7	19.7	85.5	19.7	38.3	38.3
Wheel diam (mm)	165	191	220	165	165	191
Encoder ticks per mm wheel rotation	76	66	49	76	148	128
DistConvFactor	0.840	0.969	1.32	0.826	0.413	0.424
DiffConvFactor	0.0056	0.0057	0.0034	0.0056	0.0056	0.0060

Calibration Tools - revcountcal *and* compasscal

P2OS 1.4 introduced two new P2OS calibration tools: revcountcal and compasscal. Both tools are for helping your robot and its V-2XG compass accessory adapt to your unique operating environment.

> Compasscal is for the V 2XG compass only; not for the TCM2 Module.

The revcount calibration tool helps you redefine the differential encoder counts that describes a full 360-degree turn of your robot. The default revcount value works fine for smooth, hard surfaces, but it can change dramatically for other operating environments, particularly for the Pioneer 2-AT on loose outdoor versus hard indoor surfaces.

The compass tool lets owners calibrate their integrated electronic compass for various environments and positions on the robot.

> **Saphira required**
> CompassCal and RevcountCal only work with licensed and properly installed Saphira.

Note that unlike the p2oscf and p2osdl tools, the compass and revcount calibration tools use the Saphira operating environment. Hence, the programs need to be located in the bin/ directory of a licensed Saphira installation.

A Saphira license comes with your *Activ*Media Mobile Robot, and you may retrieve new licensed versions from the http://robots.activmedia.com support website. See Chapter 3, "*Quick Start*", for additional detail.

Briefly, the programs operate in two stages. First, they connect with your Pioneer 2 or PeopleBot robot via a Saphira-based client and retrieve the robot's current operating parameters via the ENCODERpac or CONFIGpac extended SIPs (see previous chapter). Following program prompts, you change the orientation of the robot or compass—rotate the compass and/or the robot—and the program calculates a new revcount value or X and Y offsets for the electronic compass. The process may be repeated indefinitely, each time calculating fresh values; they are not recorded or changed in your robot's FLASH.

In a second stage, the calibration utilities let you save the last-acquired parameter to your P2OS microcontroller's FLASH ROM, so that the new value gets used by the robot

after each startup. You may choose not to change the FLASH ROM values. Rather, you record the new configuration values and set/reset them using the p2oscf tool described earlier in this chapter.

Install either compasscal(.exe) or revcountcal(.exe) into the bin directory of your Saphira distribution. Connect your robot's HOST serial port to the computer, by tether if you plan to save the calibration parameter to FLASH ROM. Turn on the robot's Main Power.

From an X-terminal (Linux/UNIX) or DOS command line, execute the calibration tool. The default connection port is the first serial port on your computer: COM1 for Win32 systems or /dev/ttys0 for other Linux/UNIX systems. Specify an alternative port by providing a startup argument.

For example:

```
% compasscal /dev/ttyS3
```

connects the compasscal calibration tool with the robot via the Linux computer's /dev/ttyS3 serial port.

If the tool fails to connect the Saphira client with the robot server, it will tell you "Connection refused" and hang. Press control-c to stop the program and try again after fixing the serial connection. Typically, reboot the robot with its red RESET Console button after an error.

Chapter 8 Maintenance & Repair

Your *Activ*Media robot 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 your robot's drive.

Batteries

Lead-acid batteries like those in your *Activ*Media robot 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

CAREFUL!
The Batteries slide in
TERMINALS LAST!

Except the Pioneer 2 CE and AmigoBot, *Activ*Media robots have a special battery harness and latched doors for easy access to the onboard batteries. Simply unlatch the rear door, 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 each one into a battery box compartment. 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

You may change the batteries on your *Activ*Media robot 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

If you have the standard charger accessory, insert it 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 your robot'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 2 or PeopleBot battery using the accompanying charger (roughly, three hours per volt per battery). Although you may operate the robot while recharging, it restricts the robot's mobility.

Alternative Battery Chargers

The center post of the charger socket is the positive (+) side of the battery; the case is the negative (-) side. A diode protects against the wrong charger polarity. Nonetheless, if you choose to use an alternative battery charge, be sure to connect positive to positive and negative to negative from charger to robot.

An alternative AC to DC converter/battery charger should sustain at least 0.75A at 13.75 to 14 VDC per battery, and not more than 2-2.5 amperes per battery. The High-Speed Charger accessory, for example, is a four ampere charger and should be used with at least two of the standard batteries.

An alternative charger also should be voltage- and current-limited so that it cannot overcharge the batteries.

Getting Inside

We normally discourage you from opening up your robot. However, on occasion, you may need to get inside, for instance to access the User Power connections on the Motor-Power board and attach your custom electronics. Or you may need to get to your onboard computer and its accessories.

Open the robot AT YOUR OWN RISK,
unless explicitly authorized by the factory.

We describe here how to remove your robot's Nose to get at the onboard computer. And we describe how to access the contents of the Body of your Pioneer 2-DX/DXe or -AT robot.

Removing the Nose

Pioneer 2's and PeopleBot's onboard computer sits just behind the robot's Nose. And you may have to remove the Nose to access the front sonar array's gain adjustment pot.

For early DX, CE, and AT models, and for the PeopleBot robots, the Nose is attached to the robot with two screws: one directly underneath the front sonar bay and one on the bottom of the robot just behind the seam between the robot's Body and the Nose. The new AT and the DXe model have two screws holding the Nose to the front sonar (or blank) array, and the DXe doesn't have the bottom screw since the Nose is hinged at the bottom.

Remove all Nose retaining screws with the hex wrench supplied with your robot. Except with the new AT or DXe, if you have either the Gripper or a front Bump Ring accessories, remove it first.

BE CAREFUL: Delicate wiring inside.

Once loosened, the DXd Nose pivots down on a hinge. For the other models, four pins along the Nose's back edges guide it onto the front Body of the robot. Simply pry the Nose out and away from the Body. Careful: The computer's hard-drive, fan, and speaker have attached wire harnesses that you need to relieve before completely detaching the Nose from the Body. We recommend unplugging the speaker wire and simply rotating the Nose out of the way to access the onboard computer.

Opening the Deck

The new Pioneer 2-AT and -DXe models have a center hinge in the Deck which let you easily open and access internal components without completely removing the top plate. With earlier Pioneer 2 models and with the PeopleBots, you need to remove the respective Deck and other covers to access the robot's components.

In all cases, you need to remove the selected screws completely, then carefully lift the Deck or section of the Deck up off the Body. Review the following figures for screw locations.

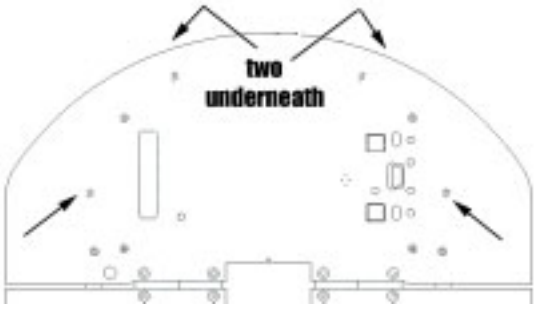

Figure 24. Remove indicated screws to flip open the front of the Pioneer 2-DXE or AT Deck. (Gripper or bump ring may remain attached.)

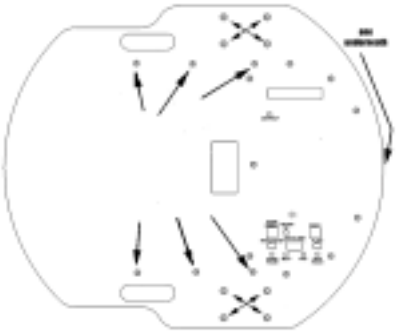

Figure 23. Remove indicated screws to remove original Pioneer 2-DX, CE, or PeopleBot Deck.

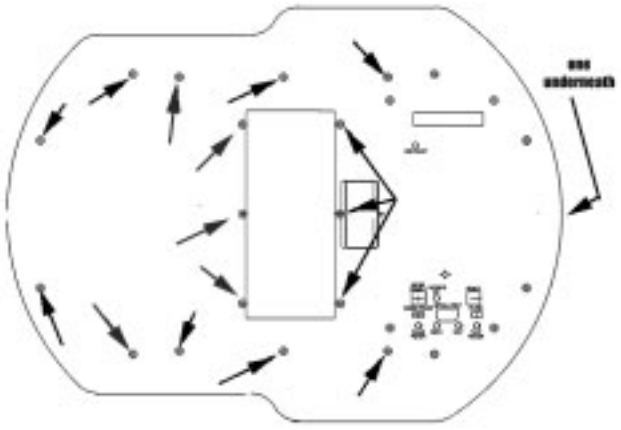

Figure 22. Remove indicated screws to remove original Pioneer 2-AT Deck.

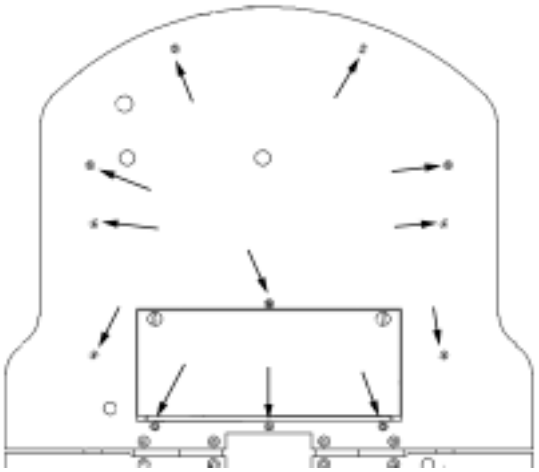

Figure 25. Remove indicated screws to flip open the hinged Pioneer 2-AT rear Deck.

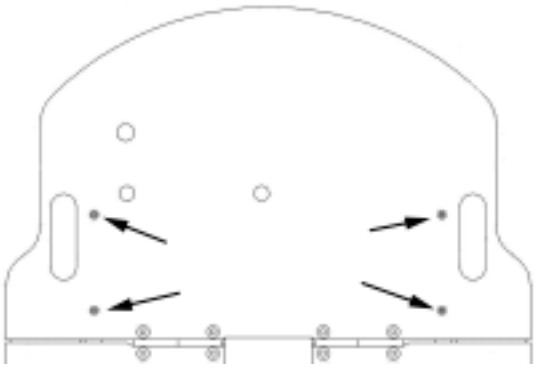

Figure 26. Remove indicated screws to flip open the Pioneer 2-DXE rear Deck.

Factory Repairs

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

support@activmedia.com
(603) 924-2184 (fax)
(603) 924-9100 (voice)

In the body of your email or fax message, describe the problem in as much detail as possible. Also, include your name, email and mail addresses, as well as phone and fax numbers. Tell us when and how we can best contact you (we will assume email 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/PeopleBot microcontroller board. These include an external (Host) serial port for P2OS-to-client connections; two internal serial ports for Host and auxiliary (AUX) communications, each with switched five and 12 VDC power; an expansion bus (General I/O); and a discrete connector for custom User I/O.

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 27):

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

Figure 27. Serial port IDC connector pins

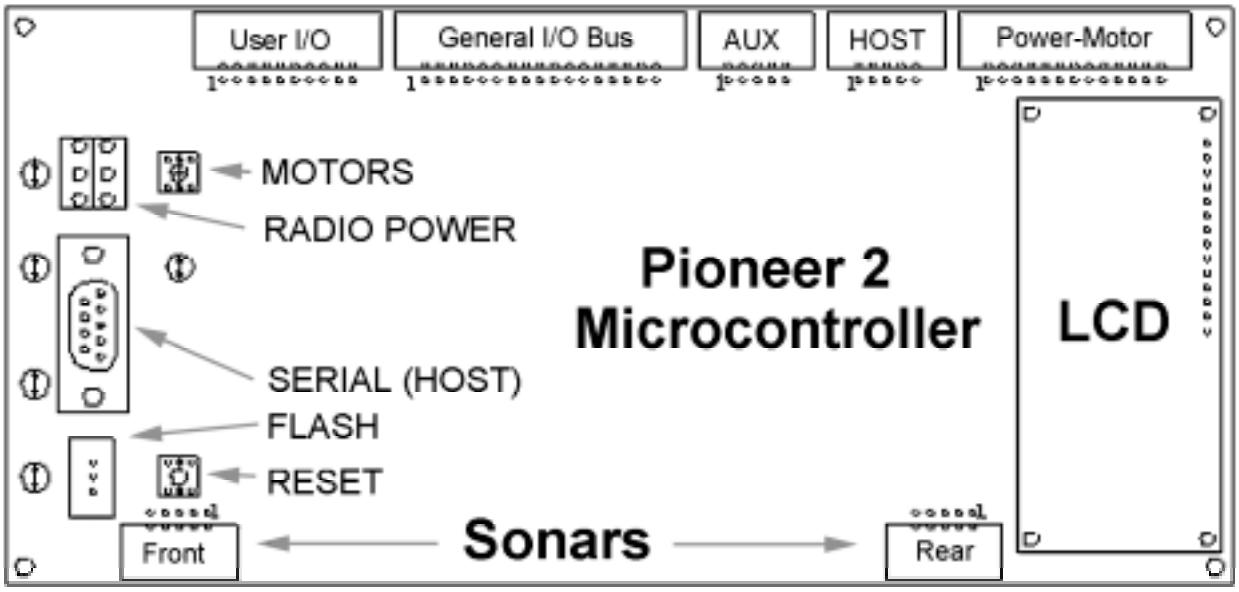

Figure 28. Pioneer 2/PeopleBot microcontroller ports and controls locations

Console Serial Port

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

Table 17 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 17. 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

P2OS 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 P2OS microcontroller support independent RS-232 serial communications. The ports also contain switched (RADIO on the Console) 5 and 12 VDC power @ 1.5A for accessory connections, such as a radio modem, radio Ethernet, PTZ robotic camera, or the TCM2 electronic compass.

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

Table 18. 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 AUX RS232-compatible serial port (connector JP7) on the Pioneer 2/PeopleBot 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 19. 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 P2OS microcontroller provides the following digital, analog, and power ports for user connections:

- ✓ 8 general-purpose digital input and output ports (ID/OD0-7)
- ✓ 5 analog-to-digital input ports (AD1-5)

- ✓ 4 Pulse-width-modulation ports (PWM1-4)
- ✓ 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 General I/O connector (below) and joystick circuitry and may not be available for use on all robots. For example, the analog lines P5_4-7, are used by the Joystick port, which now comes standard with the Pioneer 2-AT and Performance PeopleBot. In addition, four of the A/D ports and the PWM ports are shared with digital input and output lines, respectively.

Table 20. User I/O Expansion Port

Pin #	Label	Use		Pin #	Label	Use
1	P2_12	OD0 or PWM #1		**2**	P3_0	ID0
3	P2_13	OD1 or PWM #2		**4**	P3_1	ID1
5	P2_14	OD2 or PWM #3		**6**	P3_2	ID2
7	P2_15	OD3 or PWM #4		**8**	P3_3	ID3
9	P5_4	ID4 or A/D #1		**10**	P3_4	OD4
11	P5_5	ID5 or A/D #2		**12**	P3_5	OD5
13	P5_6	ID6 or A/D #3		**14**	P3_6	OD6
15	P5_7	ID7 or A/D #4		**16**	P3_7	OD7
17	P5_9	A/D #0/5		**18**	Vcc	<100ma
19	Vpp	<100ma		**20**	Gnd	

Performance PeopleBot I/O

The left and right Tabletop IR detectors on the Performance PeopleBot are connected to the ID0 and ID1 digital input ports of the User I/O, respectively, which appear as bits 0 and 1 in the digin byte of the standard P2OS Server Information Packet. Normally high (1), the digital input port goes low (0) when its respective Tabletop sensor detects an object within its range of operation.

The left and right IR breakbeams are connected the User I/O digital input ports ID2 and ID3, respectively. Normally high (1), the respective input port goes low (0) when an object breaks the IR transmitter's beam to its companion receiver.

Table 21. Performance PeopleBot I/O

User I/O Pin #	Label	Use		User I/O Pin #	Label	
1	OD0	—		**2**	ID0	Lft Tabletop
3	OD1	—		**4**	ID1	Rgt Tabletop
5	OD2	—		**6**	ID2	Lft Breakbeam
7	OD3	—		**8**	ID3	Rgt Breakbeam
9	ID4	Joystick		**10**	OD4	—
11	ID5	Joystick		**12**	OD5	—
13	ID6	Joystick		**14**	OD6	—
15	ID7	E-Stop		**16**	OD7	-—
17	AD1	—		**18**	Vcc	+5VDC
19	Vpp	+12VDC		**20**	Gnd	Gnd

— = Not used; available for other User applications

The General I/O Bus

The 34-pin IDC socket on the P2OS microcontroller provides a general-purpose I/O bus 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)
- ✓ 1 analog-to-digital input (A/D) (P5_9)
- ✓ 4 PWM/digital output (P2_12-15)
- ✓ 2 signal ground (Gnd)
- ✓ 2 Vcc (+5 VDC)
- ✓ 1 Vpp (+12 VDC)

Table 22. 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 (Bumpers)
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

The Pioneer 2 and PeopleBot robots have a separate Motor-Power board which, in conjunction with the microcontroller, provides power for motors as well as conditioned power for the standard and accessory onboard electronics. It also contains buffered

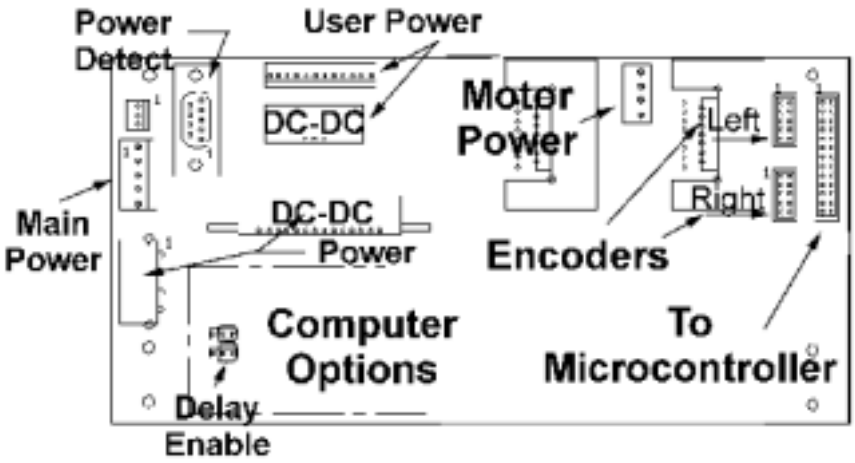

Figure 29. Pioneer 2 Motor-Power Board

pass-through connectors for the motor encoders. An optional power package ("computer ready") is available for the onboard computer system.

User Power Connections

On the Motor-Power board, four sets of unswitched battery (Vpp) and DC:DC-conditioned 5 VDC (Vcc) power, with power ground (Vpp), are available for accessory and custom electronics on a 12-pin Molex-type latch-lock (100mm) connector. Vcc gets shared with the microcontroller and accessories connected to that board (Appendix A). The maximum Vcc power available with standard robot models is 1.5A.

Table 23. User Power Connector 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, the 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, in conjunction with software, the circuit provides for automatic, unattended, crash-less shutdown of the onboard computer, either when manually powered 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 crash-less 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 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 a battery charger or by hot-swapping the batteries.

With older Motor-Power systems, 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.

Power-State Logic

A 9-pin DSUB receptacle (P1) on the Motor-Power board provides a logic-level shutdown indicator (DCD 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 24. Power State 9-pin DSUB receptacle (P1)

1	2	3	4	5	6	7	8	9
DCD	**nc**	**nc**	**pin 6**	**nc**	**pin 4**	**nc**	**nc**	**Gnd**
			10-20K adj. across 4 & 6					

Computer Power

Table 25. Computer power connector

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.

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

Appendix C

Joystick Connector

Use a 20-wire IDC-terminated ribbon cable to attach the 15-pin joystick connector to the User I/O port on the P2OS microcontroller. Both require P2OS version 1.c or later.

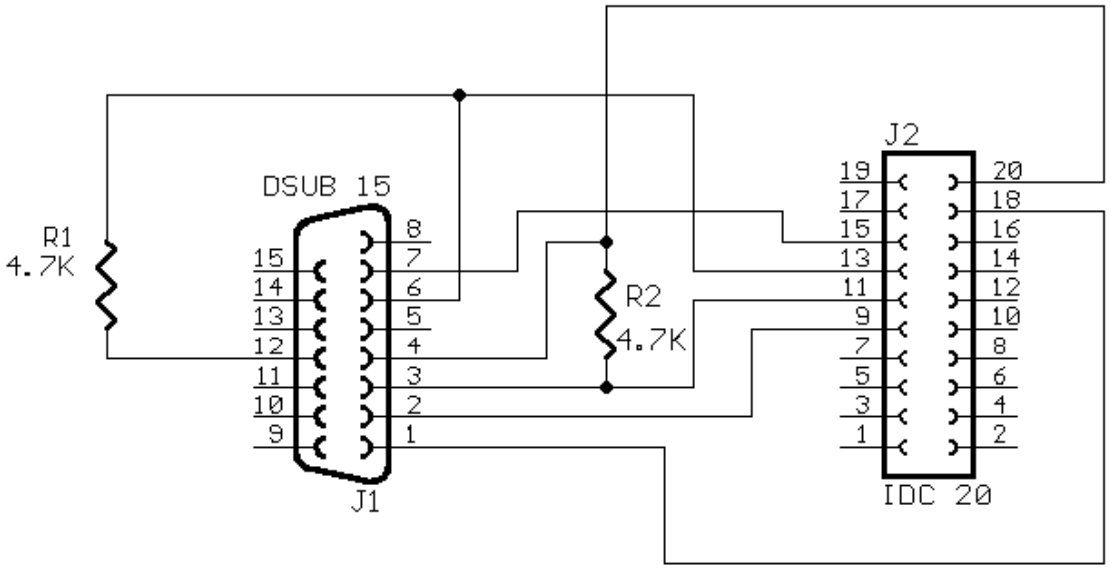

Figure 30. Standard joyport for manual drive of a Pioneer 2 or PeopleBot Mobile Robot

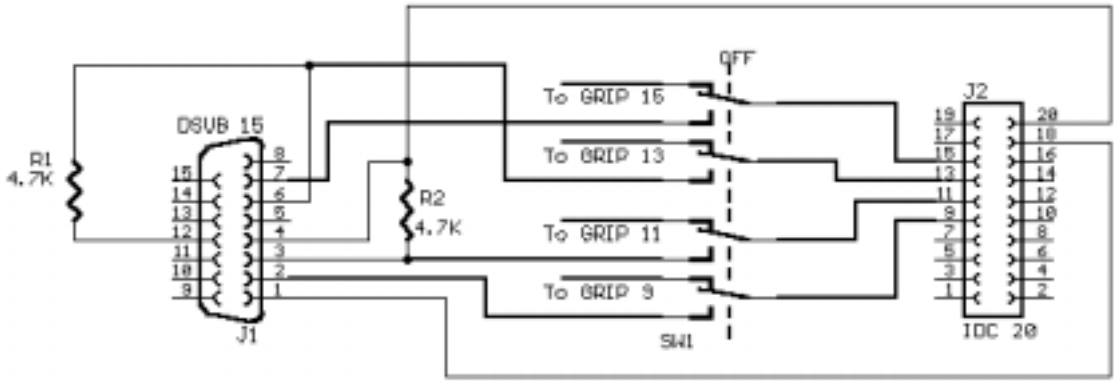

Figure 31. Joyport with signal switch for sharing User I/O ports with other accessories, such as the older style Gripper.

Appendix D

Specifications

	DXe	DX	AT	Perf PB	PB V1	CE
Physical Characteristics						
Length (cm)	44.5	44	50	47	47	44
Width (cm)	40	33	49	38	38	33
Height (cm)	24.5	22	24	124	104	22
Clearance (cm)	6.5	5.1	5.5	3.5	3.5	5.1
Weight (kg)	9	9	14	21	19	9
Payload (kg)	23	20	40	11	13	20
Power						
Batteries 12VDC lead-acid	3	3	3	3	3	1
Charge (watt-hrs)	252	252	252	252	252	84
Run time (hrs)	8–10	8–10	4-6	na	na	8-10
with PC (hrs)	3-4	3-4	2-3	3-4	3-4	na
Recharge time hr/battery std charger	6	6	6	6	6	6
High-Speed (3 batteries)	2.4	2.4	2.4	2.4	2.4	na
Mobility						
Wheels	2 pneumatic	2 solid rubber	4 pneumatic	2 pneumatic	2 solid rubber	2 solid rubber
diam (mm)	191	165	220	191	165	165
width (mm)	50	37	75	50	37	37
Caster (mm)	75	75	na	75	75	75
Steering	Differential	Differential	Skid	Differential	Differential	Differential
Gear ratio	19.7:1	19.7:1	85.2:1	38.3:1	38.3:1	19.7:1
Swing (cm)	32	32	40			32
Turn (cm)	0	0cm	0cm			0
Translate speed max (mm/sec)	1,800	1,600	700	900	800	1,600
Rotate speed max (deg/sec)	360	300	140	150	130	300
Traversable step max (mm)	20	20	89	15	15	20
Traversable gap max (mm)	89	89	127	50	50	89
Traversable slope max (grade)	25%	25%	40%	11%	11%	25%
Traversable terrains	Wheel-chair accessible	Wheel-chair accessible	Unconsolidated No carpets!	Wheel-chair accessible	Wheel-chair accessible	Wheel-chair accessible

Sensors	**DXe**	**DX**	**AT**	**Perf PB**	**PB V1**	**CE**
Sonar Front Array (one each side, six forward @ 20° intervals)	8	8	8	8	8	8
Rear Sonar Array (one each side, six rear @ 20° intervals)	8	8	8	8	8	na
Top Deck Sonar (one each side, six rear @ 20° intervals)	na	na	na	8	8	na
Encoders (2 ea) counts/rev	39,400	39,400	34,000	76,600	76,600	39,400
counts/mm	66	76	49	128	148	76
counts/rotation	18,400	20,000	22,500	33,500	39,000	18,400
Controls and Ports						
Main Power	✓	✓	✓	✓	✓	✓
Charge Port	✓	✓	✓	✓	✓	✓
Joydrive	Optional	Optional	Standard	Standard	Optional	Optional

Microcontroller and Console Controls & Ports	
Processor	Siemens 8C166 (20 MHz)
LCD	32 characters on 2 lines
Encoder inputs	4
Audio	Piezo buzzer
PWM outputs	8 (4 user-available)
Sonar inputs	2x8 (multiplexed)
Digital I/O	16 logic ports; 8 in, 8 out
A/D	5 @ 0-5 VDC; 1024- or 256-bit resolution
Digital timers	8 @ 1µsec resolution;
FLASH PROM	32 KB; P2OS and robot-specific parameters
RAM	32 KB
Power switches	1 main; 1 RADIO
Comm ports	2 RS-232 serial internal; 1 RS-232 external
Power (internal comm. ports)	12 VDC @ 1A switched; 5 VDC @ 3A switched
Logic pushbuttons	RESET and MOTORS
Indicator LEDs	Main power; RADIO power; Host SERIAL RxD and TxD

Index

Warranty & Liabilities

Your *Activ*Media robot is fully warrantied against defective parts or assembly for one year after it is shippedto you from the factory. Accessories are warrantied for 90 days. 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 *Activ*Media Robotics products 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 *Activ*Media Robotics products 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 *Activ*Media Robotics product take responsibility for support of any special or custom modification to *Activ*Media robots or their software.

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