Pioneer 3™
&
Pioneer 2™
H8-Series
Operations Manual

Pioneer 3 & Pioneer 2 H8-Series Operations Manual, version 3, August 2003

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.
- ✓ Never access the interior of the robot with charger attached or batteries inserted.

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
- ✓ Opening the robot with charger attached and/or batteries inserted
- ✓ All other forms of inappropriate operation or care

Table of Contents

Chapter 1 Introduction

Figure 1. Pioneer Mobile Robots first appeared commercially in 1995.

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

This *Pioneer 3 & Pioneer 2 H8-Series Operations Manual* provides both the general and technical details you need to operate your new Pioneer 3-DX or –AT, or Pioneer 2-DX8/DX8 *Plus* and –AT8/AT8 *Plus* mobile robot, and to begin developing your own robotics hardware and software.

For operation of previous versions of Pioneer 2 which use the Siemens C166-based microcontroller, original motor-power boards, and support systems, please consult the *Pioneer 2 Operations Manual* available through sales@activmedia.com or at our support website: http://robots. activmedia.com.

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)
- ✓ Automated dock and recharge station
- ✓ Onboard PC computer and accessories
- ✓ Radio Ethernet
- ✓ Supplementary and replacement batteries
- ✓ 3-Battery Charge Station (110/220 VAC)
- ✓ Added sonar arrays
- ✓ 2-DOF Gripper
- ✓ 5-DOF P2 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
- ✓ Heading-correction gyro
- ✓ Compass
- ✓ Bumper rings
- ✓ Serial cables for external connections
- ✓ Many more…

User-Supplied Components / System Requirements

- ✓ Client PC: 586-class or later PC with Microsoft Windows© or RedHat© Linux OS
- ✓ One RS-232-compatible serial port or Ethernet
- ✓ Four megabytes of available hard-disk storage

ADDITIONAL RESOURCES

New *Activ*Media Robotics customers get three additional and valuable resources:

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

Support Website

We maintain a 24-hour, seven-day per week World Wide 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>**

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

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.

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 with us from the online form at the support website:

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

or by email:

`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.

Tell us your robot's SERIAL NUMBER.

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 1 and Pioneer AT, Pioneer 2™ -DX, -DXe, -DXf, -CE, -AT, the Pioneer 2™-DX8/Dx8 *Plus* and -AT8/AT8 *Plus*, and the newest Pioneer 3-DX and -AT mobile robots. These small, research and development platforms share a common architecture and foundation software with all other *Activ*Media robots including AmigoBot™, PeopleBot™ V1, Performance PeopleBot™, and PowerBot™ mobile robots. All employ a common client-server robotics control architecture.

PIONEER REFERENCE PLATFORM

*Activ*Media robots set the standards for intelligent mobile platforms by containing all of the basic components for sensing and navigation in a real-world environment. They have become reference platforms in a wide variety of research projects, including several US Defense Advanced Research Projects Agency (DARPA) funded studies.

Every *Activ*Media robot comes complete with a sturdy aluminum body, balanced drive system (two-wheel differential with caster or four-wheel skid-steer), reversible DC motors, motor-control and drive electronics, high-resolution motion encoders, and long-life, hot-swappable battery power, all managed by an onboard microcontroller and mobile-robot server software.

Besides the open-systems *Activ*Media Robotics Operating System (AROS) software onboard the robot controller, every *Activ*Media robot also comes with a host of advanced robot-control client software applications and applications-development environments. Software development includes our own foundation *Activ*Media Robotics Interface for Applications (ARIA), released under the GNU Public License, and complete with fully documented C++, Java, and Python libraries and source code. SRI International's Saphira robotics development system with simulator and GUI, as well as support for advanced localization and gradient-based navigation comes bundled, too. Several third-party robotics applications development environments also have emerged from the research community for *Activ*Media robots, including Ayllu from Brandeis University, Pyro from Bryn Mawr and Swarthmore Colleges, Player from the University of Southern California, and Carmen from Carnegie-Mellon University.

Every *Activ*Media robot also comes with a plethora of expansion options, including built-in hardware support for sonar and bump sensors and lift/gripper effectors, as well as serial-port and server software support for a number of sensors, effectors, and control accessories, like an onboard PC system, automated docking/recharging system, laser range-finder, 5-DOF arm, robotic pan-tilt cameras, and much, much more.

PIONEER FAMILY OF MICROCONTROLLERS AND OPERATING SYSTEM SOFTWARE

The original Pioneer 1 mobile robot had a microcontroller based on the Motorola 68HC11 microprocessor and powered by Pioneer Server Operating System (PSOS) software. The first generation of Pioneer 2 and PeopleBot robots use a Siemens C166-based microcontroller and Pioneer 2 Operating System (P2OS) software. Now, all new

*Activ*Media robots, including Pioneer 3, Performance PeopleBot, and PowerBot, use a multifunctional Hitachi H8S-based microcontroller and new *Activ*Media Robotics Operating System (AROS) software.[2] The newest Pioneer 3 and 2 *Plus* platforms also sport an advanced motor-power board for high-power motor drives and systems power.

Although differing in some power and interfacing features, processing power, support for various sensors, and I/O, all *Activ*Media Robotics' server-operating system software—PSOS, P2OS, AmigOS, and now AROS—are upwardly compatible and virtually interchangeable. Accordingly, client software written to operate a six-year old Pioneer AT will work with a brand new Pioneer 3. We've taken great care to have all client commands for control of that original Pioneer 1 work identically in our latest robots. Client-server communications protocols over a serial communication link remain identical, too. See *Chapter 6, ActivMedia Robotics Operating System,* for details.

HITACHI H8S-BASED MICROCONTROLLER

Your H8S-based *Activ*Media robot also has a variety of expansion power and I/O ports for attachment and close integration of a client PC, sensors, and a variety of accessories—all accessible through a common application interface to the robot server software, AROS. Features include:

- ✓ 18 MHz Hitachi H8S/2357 with 32K RAM and 128K FLASH
- ✓ Optional 512K FLASH or SRAM expansion
- ✓ 3 RS-232 serial ports (4 connectors) configurable from 9.6 to 115.2 kbaud
- ✓ 4 Sonar arrays of 8 sonar each
- ✓ 2 8-bit bumpers/digital input connectors
- ✓ 1 P2 Gripper/User I/O connector with 8-bits digital I/O and 1 analog input
- ✓ 1 Expansion/bus connector containing
- ✓ 5 Analog input
- ✓ 2 Analog output
- ✓ 8-bit I/O bus with r/w and 4 chip-selects
- ✓ 2-axes, 2-button joystick port
- ✓ User Control Panel
- ✓ Controller HOST serial connector
- ✓ Main power and bi-color LED battery level indicators
- ✓ AUX and RADIO power switches with related LED indicators
- ✓ RESET and MOTORS pushbutton controls
- ✓ Piezo buzzer
- ✓ Motor/Power Board (drive system) interface with PWM and motor-direction control lines and 8-bits of digital input

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

PLUS MOTOR-POWER BOARD

The new Pioneer 3 and previous Pioneer 2-*Plus* robots come with an advanced motor-power board. It can be configured as a plug-and-play replacement for some older Pioneer 2s, as well.

Besides expanded user-power connectors and connections for ease and versatility of use, the new board supplies three to four times the motor power than the original Pioneer 2 board. Accordingly, the Pioneer 3 and 2-*Plus* platforms operate more robustly over rougher terrain (fewer stalls!) and carry significantly more payload when compared with their predecessors. And because of the power improvements, the Pioneer 3-AT and 2-

[2] AmigoBot has an H8S-based controller, too, but uses the AmigoBot Operating System tailored for its electronics.

AT8 *Plus* now come with a lower motor-gearhead reduction for faster speeds, even with much-improved turning power.

CLIENT SOFTWARE

All *Activ*Media robots operate as the server in a client-server environment: Their controllers handle 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 connected with the robot's controller via the HOST serial link and which provides the high-level, intelligent robot controls, including obstacle avoidance, path planning, features recognition, localization, gradient navigation, 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 and 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 Windows or Red Hat© Linux-based computing platform of your choice include:[3]

- ✓ *Activ*Media Robotics Interface for Applications (ARIA)
- ✓ SRIsim *Activ*Media robot simulator
- ✓ SRI's 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`

ARIA

The *Activ*Media Robotics Interface for Applications (ARIA) is a 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.

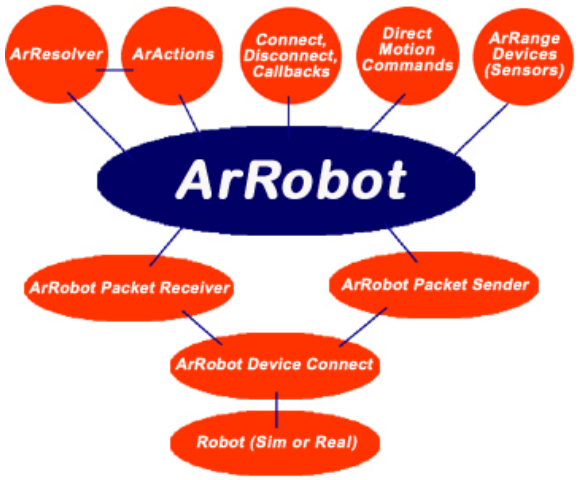

Figure 3. ARIA's architecture

ARIA is the ideal platform for integration of your own robot-control software, since it neatly handles the lowest-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, motion gyros, among many others.

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

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

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 most *Activ*Media robot owners and users. The complete, licensed Saphira robotics development environment, including C/C++ libraries, GUI interface and Simulator, comes bundled with your *Activ*Media robot.

Laser Navigation and Localization

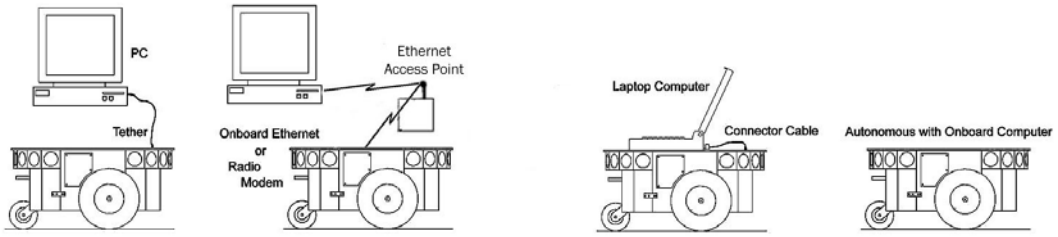

Figure 4. ActivMedia's robot servers require a computer, typically a Windows©- or RedHat© Linux-based PC, to run client software for intelligent robotics command and control operations.

A separate *Laser Navigation and Localization* package is available as a Saphira add-on. It is a comprehensive suite of software tools and applications by which, with your laser-scanning/range-finder enabled robot, you automatically create, edit, and use maps and floor plans for advanced robotics applications including localization and gradient navigation.

SUPPORTING SOFTWARE

Simulator

The SRIsim Simulator is a connection option that provides a virtual replacement for your *Activ*Media robot. By connecting to the simulator instead of a real robot, you can test your client programs, maps, and so on, when the real robot isn't practical or available.

Mapper

Mapper provides the tools you need to construct a map of your robot's real operating space ("world").

THE PIONEER LEGACY

Commercially introduced in Summer 1995, Pioneer 1 is the original platform. It came with a single-board 68HC11-based robot microcontroller and the Pioneer Server Operating System (PSOS) software. Its 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 and AT

Intended mostly for indoor use on hard, flat surfaces, the Pioneer 1 had 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.

Figure 5. The original Pioneer 1s

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 its 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 and 3s.

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.[4] Indeed, in most respects, particularly with applications software, Pioneer 2 works identically to Pioneer 1 models.

Figure 6. The Performance PeopleBot sports an attractive body design and bundled systems, including voice synthesis and recognition for human-interaction research and applications.

The *Activ*Media Robotics Pioneer 2 models -DX, -DE, -DXe, -DXf, and -AT, and the V1 and Performance PeopleBot robots used a high-

[4] 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.

performance 20 MHz Siemens 88C166-based microcontroller, with independent motor/ power and sonar-controller boards for a versatile operating environment. The controller had 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.

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

Figure 7. PowerBot carries over 100 kg of payload.

The four-wheel drive Pioneer 2-AT has independent motor drivers. Unlike its Pioneer AT predecessor, the Pioneer 2-AT comes with a stall-detection system and inflatable pneumatic tires with 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.

Other Pioneer 2-like robots include the Performance PeopleBot robots, which 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—ideal for human-interaction studies as well as for commercial and consumer mobile-robotics applications.

New Pioneer 3 and Recent Pioneer 2-DX8, -AT8, and *Plus* Mobile Robots

Two new models of Pioneer 2 appeared in the Summer of 2002, two more at the beginning of 2003, and the Pioneer 3 debuted in the Summer of 2003. They are the topics of this manual: the Pioneer 3-DX and –AT, and Pioneer 2-DX8/DX8 *Plus* and –AT8/AT8 *Plus* mobile robots. All sport a microcontroller based on the Hitachi H8S microprocessor, with new control systems and I/O expansion capabilities. The Pioneer 3 and 2-*Plus* robots also have new, more powerful motor/power systems for better navigational control and payload.[5]

Software-wise, Pioneers all are compatible with all other *Activ*Media robots, including Pioneer 1. The new *Activ*Media Robotics Operating System (AROS) software extends—but does not replace—the original PSOS and P2OS. This means that even programs that interface at the lowest communication levels will work with *all* Pioneer 1, 2, and 3 platforms. This also means that the higher level clients and applications, including Saphira, ARIA, and others including your own software, will work with AROS and any host *Activ*Media robot just as they had worked with PSOS or P2OS.[6] 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 AROS.

To the relief of those who have invested years in developing software for Pioneer 1 and 2, Pioneer 3 truly does combine the best of the new mobile robot technologies with *Activ*Media's tried-and-true robot architecture.

[5] The interim Pioneer 2-DXf had the same, more-powerful motors as the DX8s and AT8 *Plus*.

[6] 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.

MODES OF OPERATION

You may operate your Pioneer 2 and 3 robots in one of five modes:

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

Server Mode

The Pioneer H8S microcontroller comes with fully programmable 128K FLASH and 32K dynamic RAM included in its Hitachi 18 MHz H8S/2357 microprocessor. An additional 512K of dynamic RAM or FLASH-ROM is available as optional equipment. But we don't recommend that you start learning H8S programming. Rather, the robot comes to you installed with the latest AROS robotics server software.

In conjunction with client software, such as ARIA or Saphira, running on an onboard or other user-supplied computer, AROS lets you take advantage of modern client-server and robot-control technologies to perform advanced robot tasks.

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.

Maintenance and Standalone Modes

For experiments in microcontroller-level operation of your robot's functions, you may reprogram the onboard FLASH 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.

The utilities we provide for you to reprogram the H8S-based controller's FLASH also may be used to update and upgrade your robot's AROS. In a special Maintenance Mode, you also adjust your robot's operating parameters that AROS uses as default values on startup or reset. See Chapter 7, *Updating & Reconfiguring AROS,* for much more detail.

We typically provide the maintenance utilities and AROS 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 *Activ*Media robot users worldwide.

Joydrive and Self Test Modes

Finally, we provide onboard software and controller hardware that lets you drive the robot from a tethered joystick when not otherwise connected with a controlling client. And we provide some self-test programs that exercise your robot's hardware and software. We examine these modes in some detail in Chapter 5, *Joydrive and Self-Tests*.

Chapter 3 Specifications & Controls

*Activ*Media's Pioneer 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 3-DX with onboard PC is a fully autonomous intelligent mobile robot. Unlike other commercially available robots, Pioneer'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 AROS 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.

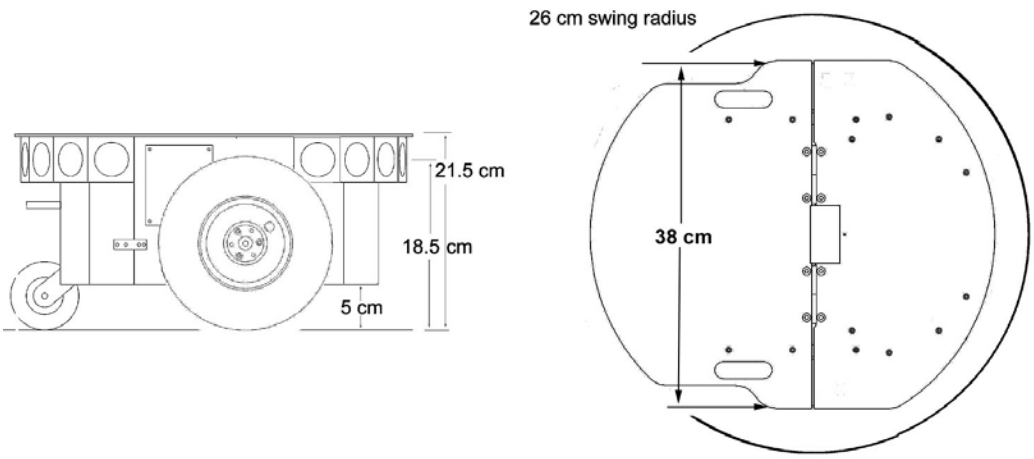

Figure 8. Pioneer 3-DX's physical dimensions and swing radius

PHYSICAL CHARACTERISTICS

Weighing only 9 kg (20 pounds with one battery), the basic Pioneer 3- and 2-DX8/DX8 *Plus* mobile robots are lightweight, but their strong aluminum body and solid construction make them virtually indestructible.

These characteristics also permit them to carry extraordinary payloads: The new Pioneer 3-DX can carry up to 23 Kg (50 lbs.) additional weight; the 3-AT can carry over 35 Kg (70 lbs.) more! Yet, Pioneer 2s and 3s are lightweight enough that it is also as easy to transport as a suitcase—a task made even easier by the DX's built-in handle.

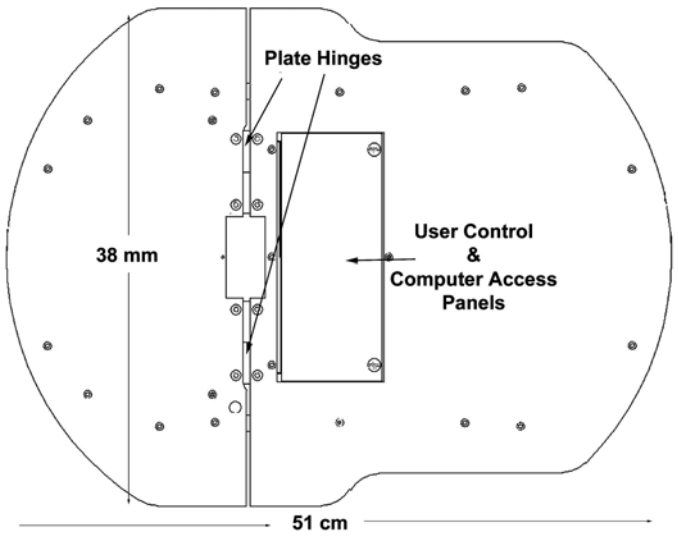

Figure 9. Pioneer 3-AT's console and hinged deck

MAIN COMPONENTS

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

- ✓ Deck
- ✓ Motor Stop Button
- ✓ User Control Panel
- ✓ Body, Nose, and Accessory Panels
- ✓ Sonar Array(s)
- ✓ Motors, Wheels, and Encoders
- ✓ Batteries and Power
- ✓ Deck

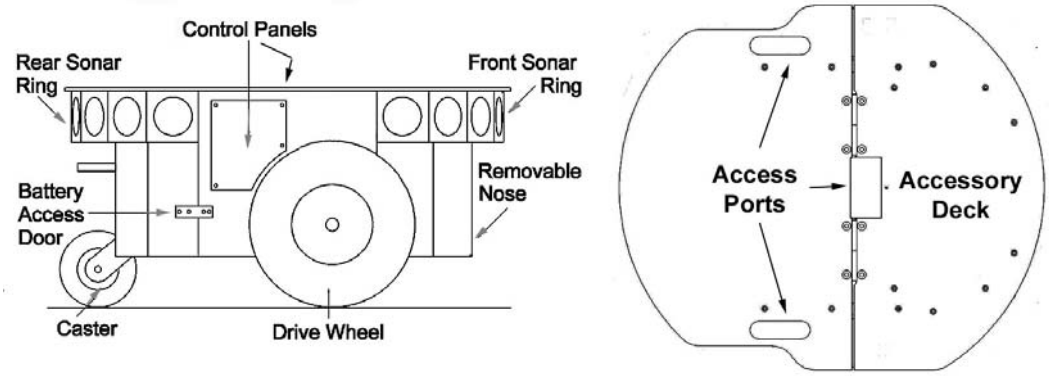

Figure 10. Components of the Pioneer 3

The original Pioneer 2-DX, CE, and AT decks are one piece—the top plate of the robot. The newer DXe and AT, and now the DX8/DX8 *Plus*, AT8/AT8 *Plus*, and Pioneer 3 models have hinged top-plates which give you much easier access to the internal components of the robot. See Chapter 8, *Maintenance & Repair*, for access details.

The robot's deck is simply the flat surface for mounting projects and accessories, such as the PTZ Robotic Camera and the laser range finder. Feed-through slots on each side of the DX deck let you conveniently route cables to the accessory panels on the side panels of the robot. A removable plug in the middle of the deck on all models gives you convenient access to the interior of the robot.

When mounting accessories, 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.

Motor Stop Button

All new Pioneer 3-AT and, upon request, some new Pioneer 3-DX robots have a `STOP` button at the rear of the Deck. Press and release it to immediately disengage the robot's motor power. It will also cause a stall and result in incessant beeping from the onboard piezo speaker (see *User Controls* below).

Press the `STOP` button in to re-engage motor power and stop that incessant beeping noise. Note that you may also have to re-engage the motor controls when connected with a client, either by manually pressing the `MOTORS` button on the User Control Panel, or through a special client command. Read on…

User Control Panel

The User Control Panel is where you have access to the AROS-based onboard microcontroller. Found inside the AT's hinged access panel on the deck or on the left-side panel of the DX, it consists of control buttons and indicators, and an RS232-compatible serial port with a 9-pin DSUB connector.

The red PWR LED is lit whenever main power is applied to the robot. The green STAT LED state depends on the operating mode and other conditions. It flashes slowly when the controller is awaiting a connection with a client and flashes quickly when in joydrive mode or when connected with a client and the motors are engaged. It also flashes moderately fast when the controller is in maintenance mode.

The BATTERY LED's apparent color depends on your robot's battery voltage: green when fully charged (>12.5 volts) through orange, and finally red when the voltage is below 11.5. When in maintenance mode, however, the BATTERY LED glows bright red only, regardless of battery charge.

A built-in piezo buzzer (audible through the holes just above the STAT and PWR LEDs) provides audible clues to the robot's state, such as upon successful startup of the controller and a client connection. An AROS client command lets you program the buzzer, too, to play your own sounds.

The SERIAL connector, with incoming and outgoing data indicator LEDs (RX and TX, respectively), is through where you may interact with the H8S microcontroller from an offboard computer for tethered client-server control and for AROS system maintenance. The port is shared internally by the HOST serial port, to which we connect the onboard computer or radio modem/Ethernet. Digital switching circuitry disables the internal HOST serial port if the computer radio modem is OFF. However, serial port interference will be a problem if the HOST and User Control SERIAL ports are both occupied and engaged. Accordingly, remove the cable from the SERIAL port if you plan to connect with the controller through the onboard radio modem or PC.

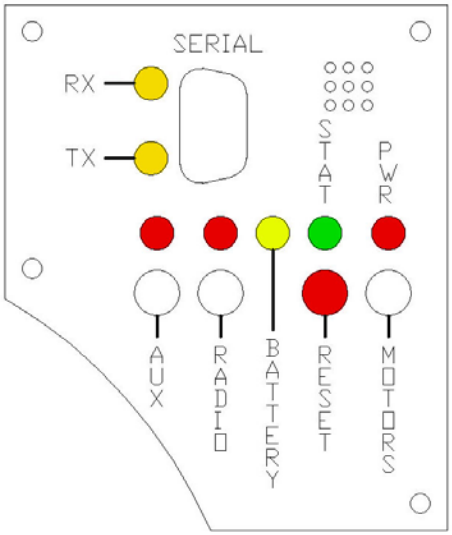

Figure 11. P3-DX User Control Panel

or

RADIO and AUX are pushbutton switches which engage or disengage power to the respective devices on the Motor/Power Interface board. See *Appendix B* for power connections. Respective red LEDs indicate when power is ON.

The red RESET pushbutton acts to unconditionally reset the H8S controller, disabling any active connections or controller-attached devices, including the motors.

The white MOTORS pushbutton's actions depend on the state of the controller. When connected with a client, push it to manually enable and disable the motors, as its label implies. When not connected, press the pushbutton once to enable joydrive mode, and again to enable the motors self-test.

To engage AROS maintenance mode, press and hold the white MOTORS button, press and release the red RESET button, then release MOTORS. In the future, the white MOTORS button may engage other modes, such as when in AROS standalone mode.

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, laser range finder, and more.

On all models except the Pioneer 2-CE, 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 nose is where we put the onboard PC. The nose is readily removable for access: Simply remove two screws from underneath the front sonar array. A third screw holds the nose to the bottom of the AT's body. The DX nose is hinged at the bottom.

Once the mounting screws are removed, simply pull the nose away from the body.[7] 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.

All DX's come with a removable right-side panel 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.

AT's come with a single access panel in the deck. Fastened down with finger-tight screws, the User Control Panel and onboard computer controls are accessible beneath the hinged door.

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, H8S/AROS-based *Activ*Media robots support up to four sonar arrays, each with eight transducers that provide object detection and range information for collision avoidance, features recognition, localization, and navigation. The sonar positions in all Pioneer 2 and 3 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.

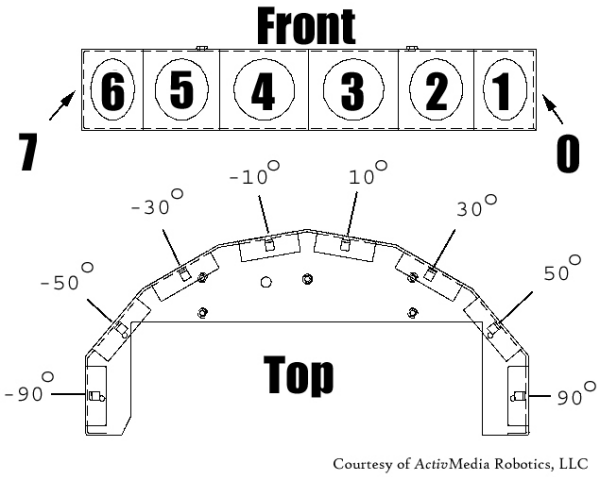

Figure 12. Pioneer 3 sonar array

[7] With older Pioneer 2 models, you also needed to remove the Gripper before removing the Nose. With the DXE, and newer DXs and ATs, the Nose and Gripper come off together, so you only need to remove the Nose's mounting screws.

Each sonar array comes with its own driver electronics for independent control. Each array's sonar are multiplexed; the sonar acquisition rate is adjustable, normally set to 25 Hz (40 milliseconds per sonar per array). Sensitivity ranges from ten centimeters (six inches) to over four meters, depending on the ranging rate. You may control the sonar's firing pattern through software, too; the default is left-to-right in sequence 0 to 7 for each array. See the AROS chapters 6 and 7 for details.

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.[8] 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.

Motors, Wheels, and Position Encoders

Pioneer 2's and 3's 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. However, AROS converts most client commands and server information from platform independent distance units into platform-dependent encoder ticks, as expressed in the `Ticksmm` FLASH parameter, calculated as the encoder counts (4 * 500, typically) divided by the product of wheel circumference times gear ratio.

Inflate the tires ***evenly***
or your robot won't drive properly.

All Pioneer 3 robots now come with pneumatic tires so that you may configure your robot for differing terrains. In any configuration, however, be careful to inflate the tires evenly and adjust the respective `Ticksmm` and rotational `Revcount` FLASH parameters for proper operation. We ship with the tires inflated to 23 psi each.

BATTERIES AND POWER

Except when outfitted with the automated docking/charging system (see below), Pioneer 2 and 3 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 through a hinged and latched rear door. We provide a suction cup tool to help grab

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

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. AT charge life typically ranges from two to three hours. The DX runs 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.

Battery Indicators and Low Voltage Conditions

The User Control Panel has a bi-color LED labeled `BATTERY` that visually indicates current battery voltage. From approximately 12.5 volts and above, the LED glows bright green. The LED turns progressively orange and then red as the voltage drops to approximately 11.5 volts.

Aurally, the User Control Panel's buzzer, if active (see the AROS `SoundTog` client command and FLASH parameter), will sound a repetitive alarm if the battery voltage drops consistently below the FLASH `LowBattery` level. If the battery voltage drops below 11 volts, the microcontroller's watchdog server automatically shuts down a client connection and notifies the computer, via the `HOST RI` (ring indicator) pin, to shut down and thereby prevent data loss or systems corruption due to low batteries.

Recharging

Typical battery 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.

The new automated docking/recharging system is the best option. Because its integrated charge-management system has sufficient power and actively adjusts to system loads, it can run your robot's onboard systems while properly and optimally recharging its batteries. And because the charging mechanism may be operated independently of your robot's systems power, you may start up and shut down your robot and its onboard systems without disturbing the battery charging cycle.

All our recommended chargers are specifically designed for 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.

DOCKING/CHARGING SYSTEM

The Pioneer 3/PeopleBot docking/charging accessory is both a manual and an automated mechanism. Onboard controls, triggered either by the DEPLOY CHARGER button near the manual CHARGE port, or by H8S controller-mediated client commands, deploy actuated contacts on the bottom of the robot, which in turn seat onto the charging platform. Then, when activated by an IR-based, unique frequency-modulated signal from the robot, the charger platform delivers up to 17 VDC @ 11.5 A to its plates.

While connected, onboard circuitry conditions the power to optimally charge the three 21-Ahr, 12 VDC lead-acid batteries (6 A charging current max) and provides sufficient power (up to 5.5 A) for operation of all onboard systems.

The charging mechanism and onboard power conditioning circuitry can be retrofitted to all Pioneer 3 and some Pioneer 2 and PeopleBot robots; all require return to the factory.

Manual Operation (Robot Power OFF)

With MAIN POWER off, place the robot over the charge platform so that its charging contacts are perpendicular to and, when deployed, contact the charger plates. Note that no charging power is applied to the plates on the platform; only low signal (5VDC @ <300mA) power for the IR detectors.

Press and hold the DEPLOY CHARGER button to manually deploy the charge mechanism on the bottom of the robot. Hold for a few seconds, but not more than 10 seconds. Charging is activated by positive contact with the charging platform. In that case, the charge lamp on the charger unit will light and the robot's contacts will remain deployed when you release the DEPLOY CHARGER button. Otherwise, the mechanism will retract. In that case, re-position the robot and try again.

The robot's charging mechanism automatically retracts if you press the DEPLOY CHARGER button while charging, if you move the robot on the docking platform and lose positive charging contact, or if you remove power from the charger unit. In all cases, charging power is removed immediately from the docking platform when not actively engaged by the robot.

Manual Operation (Robot Power and Systems ON)

Because the automated docking/charging system's charger and integrated circuitry actively adjusts to system loads, it can run your robot's onboard systems while properly and optimally recharging its batteries. And because the charging mechanism may be operated independently of your robot's systems power, you may start up and shut down your robot and its onboard systems without disturbing the battery charging cycle, if engaged.

For example, with MAIN POWER on, use joystick mode to position the robot onto the charging platform. Then reset the robot controller and manually deploy the charging mechanism as described in the section above. Thereafter, switch MAIN POWER off, or conversely, start up and shut down other onboard systems, including the PC, camera, laser, and other accessories, to proceed with development work without disturbing battery recharging.

The same conditions apply to remove charging power and retract the robot's charging mechanism with the robot's MAIN POWER on as well as off. In addition, engaging the motors, such as when you press the white MOTORS button on the robot controller to engage joystick/self-tests mode, also disengages recharging and retracts the charging

mechanism. And the charging mechanism will not activate until you disengage the motors, either manually or programmatically.

RADIO CONTROLS AND ACCESSORIES

All *Activ*Media robots are servers in a client-server architecture. You supply the client computer to run your intelligent mobile-robot applications. The client can be either an onboard piggy-back laptop or embedded PC, or an offboard PC connected through radio modems or wireless serial Ethernet. In all cases, that client PC must connect to the `HOST` serial port of the robot's microcontroller in order for the robot and your software to work.

For the piggyback laptop or embedded PC, that serial connection is a cable. Radio modems simply replace that serial cable with a wireless tether. Accordingly, if you have radio modems, one is inside your robot and connected to the controller's HOST serial port, and the other modem plugs into a serial port on some offboard computer where you run your client software. Hence, in these configurations, there is one dedicated client computer. (See Appendix C for radio modem settings.)

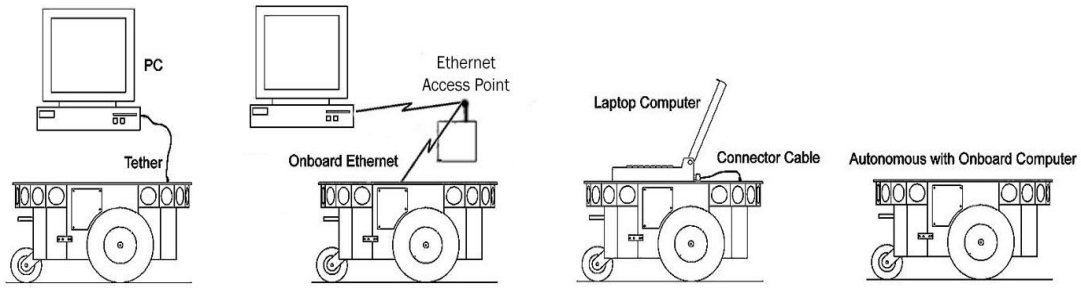

Figure 13. Client-server connection options.

Radio Ethernet is a little more complicated because it lets you use many different computers on the network to become the robot's client. A special onboard Serial-Ethernet accessory that we provide is a standard wireless Ethernet radio which connects to your local TCP/IP network through an Access Point. But it's different from most standard wireless Ethernet devices in that it also connects to the `HOST` serial port on the robot's microcontroller. It works by automatically translating network-based Ethernet packet communications into streaming serial for the robot controller and back again. Accordingly, you may run the robot's client on any network PC just as if that client PC were connected directly to the robot's controller. (See Appendix D for Serial Ethernet settings.)

A major disadvantage of the wireless Ethernet-to-serial device, however, as well as for radio modems, is that they require a constant wireless connection with the robot. Disruption of the radio signal—a common occurrence in even the most modern installations—leads to poor robot performance and very short ranges of operation.

This is why we recommend onboard client PCs for wider, much more robust areas of autonomous operation, particularly when equipped with their own wireless Ethernet. In this configuration, you run the client software and its interactions with the robot controller locally and simply rely on the wireless connection to export and operate the client controls, such as through X-Windows or VNCserver. Moreover, the onboard PC is often needed for local processing, such as to support a laser range finder or to capture and process live video for vision work.

ONBOARD PC

Unlike the original Pioneer 1, Pioneer 2 and 3 robots are designed to support an onboard, internally integrated PC for fully autonomous operation. Mounted just behind the nose of the robot, the PC is a common EBX form-factor that comes with up to four serial ports, 10/100Base-T Ethernet, monitor, keyboard, and mouse ports, two USB ports, and support for floppy, as well as IDE hard-disk drives. For additional functionality, such as for sound, video framegrabbing, firewire or PCMCIA bus, and wireless Ethernet, the onboard PC accepts PC104 and PC104-plus (PCI bus-enabled) interface cards that stack on the motherboard.

Necessary 5 VDC power comes from a dedicated DC:DC converter, mounted nearby. A hard-disk drive is specially shock-mounted to the robot's nose, in between a cooling fan and computer speaker.

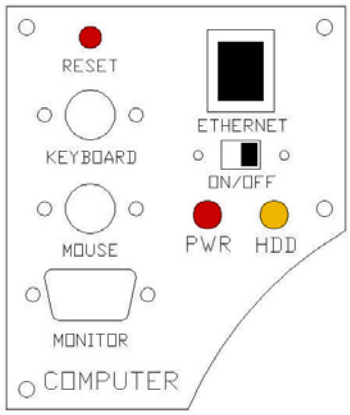

Figure 14. DX computer control side panel

The onboard PC communicates with the H8S microcontroller through its HOST serial port and the dedicated serial port `COM1` under Windows or `/dev/ttyS0` on Linux systems. Automatic systems on the microcontroller switch in that HOST-to-PC connection when PC-based client software opens the serial port. Otherwise, the PC doesn't interfere with externally connected clients through the shared `SERIAL` port on the User Control Panel.

Note also that some signals on the H8S microcontroller's HOST serial port as connected with the onboard PC or other accessory can be used for automated PC shutdown or other utilities: Pin 4 (DSR) normally is RS232 high when the controller operates normally; otherwise it is low when reset or in maintenance mode. Similarly, pin 9 (RI) normally is low and goes RS232-level high when the robot's batteries drop below a set (nominally 11 VDC) voltage level.

Computer Control Panel

User-accessible communication and control port connectors, switches, and indicators for the onboard PC are on the Computer Control Panel, found on the right side panel of the DX or in the hinged control well next to the User Controls of the AT.

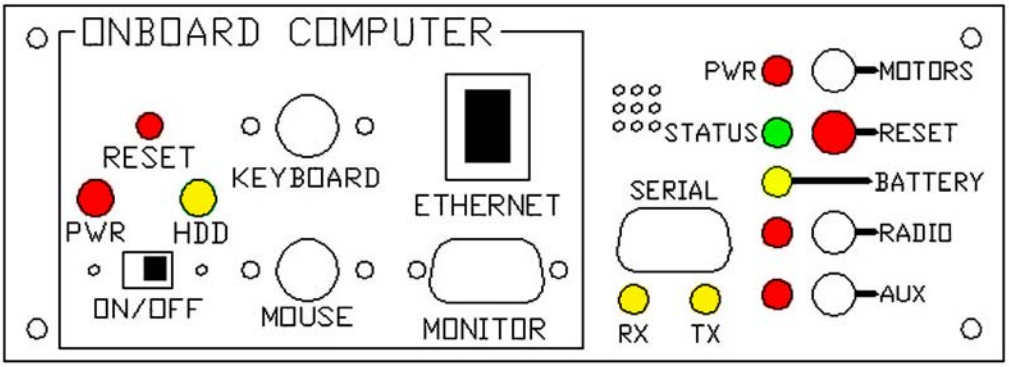

Figure 15. AT computer and user controls

The controls and ports use common connectors: standard monitor DSUB and PS/2 connectors on the mouse and keyboard. The Ethernet is a 10/100Base-T standard RJ-45 socket.

The `ON/OFF` slide switch directly controls power to the onboard PC—through `Main Power`, unlike some earlier versions of the onboard system which included a delayed power shutdown. The `PWR` LED lights when the computer has power.

The `HDD` LED lights when the onboard hard-disk drive is active. The RESET button restarts the PC.

Operating the Onboard PC

This is a brief overview of operating the onboard PC. Please consult the *Computer Systems Documentation* and the OS manufacturer's documentation for more detail. *Activ*Media Robotics' software runs over either Microsoft Windows (currently Windows 2000®) or RedHat® Linux (currently version 7). Accordingly, we prefer (the latter, in particular) and support those OSes on the onboard PC.

When we perform the installation and configuration, we install our robotics and accessory software typically in `/usr/local` on Linux systems, or in `C:\Program Files\ActivMedia Robotics` under Windows. Of course, we install the appropriate drivers for the various accessory expansion cards, such as for a framegrabber or sound card. Please consult the respective *Activ*Media Robotics application software manuals, such as the ActivMedia Color Tracking System (ACTS) for the video framegrabber or Festival for the sound card.

The first time you access the onboard PC, we recommend that you put the robot up on blocks so that it cannot inadvertently move and wreak havoc with external connections. Then attach a keyboard, monitor, and mouse to their respective sockets on the Computer Control Panel. Switch `Main Power` and then the computer power switch on.

After boot up, log in to the system. We've already created two users: one with common systems and file read/write permissions (`guest`) and one with full-access to the PC software and OS—`root` (Linux) or `administrator` (Windows). If there is a password (usually not) it's `activmedia`. When connected directly, we recommend you log in with full-access capabilities so that you can do systems set up and maintenance, such as change passwords, add users, and set up the network. Do note that with Linux systems, you cannot log in remotely over the network as `root`; you must log in as a common user and use the `'su -'` command thereafter to attain superuser (root) status.

Once logged into a Windows system, it's simply a matter of clicking the mouse to select programs and applications. With Linux, use the 'startx' command to enable the X-Windows desktop and GUI environment. You might perform some of the *QuickStart* activities this way, although motion is impractical because of the monitor, mouse, and keyboard tethers. You may remove these while the system is active at your own risk.

Rather, we suggest that you run the *QuickStart* activities from an offboard computer first (onboard PC off), and then tackle the networking issues to establish a remote, preferably wireless connection with your robot.

PC Networking

The RJ-45 connector on the Computer Control Panel provides wired 10/100Base-T Ethernet networking directly with the onboard PC. With the purchased option, we also install a PCMCIA adaptor card on the PC's accessory stack and insert a 10GHz 11Mbps

802.11b-compatible wireless Ethernet card in one of its slots. The wireless Ethernet antenna sits atop the top deck.

To complete the wireless installation, you will need to provide an Access Point module (comes as an accessory with most units). Attach the Access Point to one of your LAN hubs or switches with a standard CAT5 100Base-T cable. No configuration required. We use the default operating mode: "managed" client-server.

We ship installed PC systems' preset and tested at a fixed IP address with Class-C network configuration. We allocate the same IP to both the wired and wireless Ethernet ports, typically 192.168.1.32. Although you need not fuss with drivers or low-level device settings, before you may establish a network connection with the onboard PC (*not* the robot's controller!), even if just through a "cross-over" Ethernet cable to another PC, you'll need to reconfigure the robot's PC network settings. Please consult with your network systems administrator for networking details.

Briefly, with Windows, go to the Control Panel's `Network and Dialup Connections` wizard and choose the networking device's `Properties` to change the IP address and other details. Under Linux, there are similar, GUI-based tools under X-Windows to help you set up the network, such as `netcfg`, but we prefer to edit (*emacs* or *vi*) the salient network settings in `/etc/sysconfig/network` and in the specific device configuration files found in `/etc/sysconfig/network-scripts/`, such as `ifcfg-eth0` (wired Ethernet) and `ifcfg-eth1` or `ifcfg-wvlan0` (wireless).

From Windows, use the Control Panel `Network and Dialup Connections` tool to enable or disable a particular device. From Linux, use `ifup` and `ifdown` to enable or disable an Ethernet device. For example, as superuser, type '`ifdown eth0; ifup eth1`' to switch from a tethered to a wireless Ethernet connection.

For remote connections over Ethernet to your onboard PC, simply use *telnet* or the more secure *ssh* to log in to your Linux system. Allow X-windows server connections at your remote PC (`xhost`) if you plan to export the X-Windows display from the robot PC for remote GUI-based controls (`export DISPLAY=remote's hostname or IP:0`, for example).

With Windows, you will need a special remote-control application to establish a GUI-based connection from a remote computer to the onboard PC over the network; *VNCserver*, for example, or *XWin32*.

Please note that you may ***not*** connect with the robot's microcontroller directly over the network: That is, you cannot run a client application, such as the ARIA demo or Saphira, on the remote PC and choose to directly connect with the robot server by selecting the robot PC's IP address. Rather, either run the client application on the onboard PC and export the display and controls over the network to the remote PC (preferred), or use the ARIA-based `IPTHRU` programs (see program sources in `Aria/examples`) to negotiate the IP-to-serial conversions needed by the client-server connection.

UPS and Genpowerd

To protect your robot's onboard PC data, we've enabled a detection scheme in AROS and UPS-like software on the computer that invoke shutdown of the operating system in the event of a persistent low-battery condition.[9]

AROS versions 1.6 and later raises the HOST serial port's RI pin 9 to RS232-level high when the P2-H8 controller is operating normally, but when your robot's battery power drops the

[9] The original Pioneer 2 Motor-Power boards implemented a similar strategy in hardware.

below safe operating level of ~11 VDC.[10,11] *Genpowerd* running on the onboard Linux system or *ups.exe* running under Windows, detects the change of state and initiates OS shutdown after a short wait, during which the shutdown may be canceled by raising the battery voltage, such as by attaching a charger.

Genpowerd monitors the HOST serial RI port on `/dev/ttyS0`. Windows' *ups.exe* requires a dedicated serial port—`COM2` on current systems, and prefers to monitor the CTS line. Consequently, we wire the onboard PC serial connector differently for Linux versus Windows PC. Please consult the AROS chapters for more detail.

SAFETY AROS WATCHDOGS

AROS contains a communications `watchdog` that will halt the robot's motion if communications between a PC client and the robot 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.

AROS 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.

There also is the `LowBattery` FLASH parameter that sets off an audible warning when the batteries fall below a safe charge level. To avoid systems corruptions, the AROS servers force a soft system shutdown, possibly including the onboard PC (Linux *genpowerd* or Windows' *ups.exe*), when the batteries fall below approximately 11 volts.

All these "failsafe" mechanisms help ensure that your robot will not cause damage or be damaged during operation. You may reconfigure the various FLASH-based parameter values to suit your application. See Chapter 7, *Updating & Reconfiguring AROS*, for details.

[10] RI and DSR on the HOST serial port are RS232 low during reset or when the controller is in Maintenance Mode.
[11] AROS versions 1.5 and earlier raised the HOST serial port's DSR and RI to RS232-level high and lowered the RI for low-power condition, which worked fine for Linux genpowerd, but was incompatible with Windows' ups.

Chapter 4 Quick Start

This chapter describes how to quickly set up and operate your new *Activ*Media robot with the ARIA demonstration software. For more details about programming and operating your *Activ*Media mobile robot with ARIA, Saphira, or other client software, see their respective programming manuals.

PREPARATIVE ASSEMBLY

Your *Activ*Media robot comes fully assembled and ready for out-of-the-box operation. However, you may need to attach some accessories that were shipped separately for safety. The procedures we describe herein are for control of the basic robot.

If you have the onboard PC option, we recommend that you leave it off and perform the following tests first with a laptop or desktop computer tethered to the robot's serial port on the User Control Panel, then attack the many networking issues before you establish a remote-control connection with the onboard PC.

Install ARIA

The ARIA client software-development environment, including the ARIA demonstration program and robot simulator, come on CD-ROM with your new robot. They also come installed in your robot's onboard PC, if you purchased this option.

*Activ*Media Robotics customers also may obtain ARIA and related software and updates from our support website:

`http://robots.activmedia.com`

When installed, ARIA typically requires ten or more megabytes of hard-disk space.

The Windows version of ARIA is a self-extracting InstallShield® archive. Simply double-click its `.exe` icon and follow the extraction program's instructions. Normally, ARIA is put into a directory named `C:\Program Files\ActivMedia Robotics\ARIA`. The demonstration program and simulator get put into the `bin\` subdirectory. For convenience, you may access all these from the `Start Menu's Programs` option. The demonstration program's source code and MSVC++ project and workspace files are in the `examples\` subdirectory.

Linux users must have `superuser` (`root`) permissions in order to install ARIA. It comes as an RPM installation archive:

`rpm -ihv aria...`

and gets installed in `/usr/local/Aria`. The ARIA demonstration program and simulator get put into the `bin/` subdirectory. The demonstration sources and makefile are in the `examples/` subdirectory.

Linux users should also be sure they have permission to read/write through their PC's serial port that connects with the robot. The default is `/dev/ttyS0`. ARIA is a terminal application that does not include a GUI, so its programs do not require X-Windows.

CAREFUL
Slide the batteries into the robot TERMINALS LAST.
Otherwise, you will damage the robot.

Install Batteries

Out of the box, your *Activ*Media robot comes with its batteries fully charged, although shipped separately, unless you have the automated docking/charging system. For most models, slide one or up to three batteries into robot's battery box through the back door. Balance them: one in the center; if two, then one on each side.

Client-Server Communications

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

- ✓ A tether cable from the robot's 9-pin serial connector on the User Control Panel to a computer
- ✓ A piggyback laptop cabled to the User Control Panel
- ✓ Serial Ethernet
- ✓ Radio Modem
- ✓ An integrated onboard PC wired internally for direct onboard control

STARTING UP CLIENT AND SERVER

We recommend that you first test your robot and are confident of its operation before putting it together with and controlling it from the ARIA demonstration client.

Drive Self-Test

Position your *Activ*Media robot on the floor or ground in an open space, or up on blocks if you have attachments to the Computer Control Panel. Slide the `Main Power` switch to `ON`. You should hear an audible beep, and the `Power` light and `Battery` light should glow while the `Status` light blinks rhythmically on the User Control Panel. The same AROS initialization sequence also occurs whenever you press the red `RESET` button.

Now press the white `MOTORS` button twice to engage the motor's self-test. If your robot is working properly, it should move or rotate the wheels in four brief, but distinctive turns, forward and back, left and right. If not, please contact support@activmedia.com for assistance.

Press the red `RESET` button to prepare for the client connection.

Client Server Connection

ARIA's examples are text-based "terminal" applications that do not include a GUI, so its programs do not require X-Windows over Linux or special software on a remote PC client—a simple telnet session will do the trick.

First, please note well that you cannot connect with and control your *Activ*Media robot through its controller directly from a remote client over the network without special hardware (new radio Ethernet-to-serial device) or, alternatively, special software that runs on the onboard computer and converts IP packets into serial data.[12] Otherwise, you must run the client software on the robot's PC or on a PC that is connected to the robot's controller HOST serial port. You may, of course, export the controls and display over the network from X-windows or with special Windows software, such as VNCserver.

To start the ARIA client demonstration program and connect with the robot, we presume that you have completed the preparatory stages of this chapter by installing ARIA (as

[12] Look in the ARIA/examples directory for a program called ipthru. It converts IP to serial and back again for remote-control clients connected through the onboard PC.

needed), by starting and testing the robot, and by connecting the client PC with the AROS controller via a serial link. Now it is time to connect the ARIA demonstration program with your robot.

If you are using radio modems or the new Low-Speed Ethernet-to-serial device to communicate wirelessly from a desktop PC to the robot controller, now is a good time to power the units. The RADIO power switch for the integrated radio is on the User Control Panel. The other radio modem should be attached to your PC and powered via the module that came with the unit. If using the Ethernet-to-serial radio, be sure you have a connection with a local access point, or have a peer-to-peer radio Ethernet installed in your client computer.

Windows users may select the ARIA `demo` from the `Start` menu, in the `ActivMedia Robotics` program group. Otherwise, start if from the ARIA `bin\` directory.

Linux users will find the compiled `demo` in `/usr/local/Aria/bin/` or in `examples/`. Start it:

```
% ./demo
```

Demo Startup Options

Table 1. ARIA demo command line arguments

`--remoteHost <Host Name or IP>`	Connect with robot through a remote host over the network instead of a serial port; requires special hardware or IPTHRU software mediation.
`--robotPort <Serial Port>`	Connect with robot through specified serial port name; COM3, for example.
`--remoteRobotTcpPort <Number>`	Remote TCP host-to-robot connection port number; default is 8080.
`--laserPort <Serial Port>`	Connect with laser rangefinder through the specified serial port name; /dev/ttyS3, for example.
`--remoteLaserTcpPort <Number>`	Remote TCP host-to-laser connection port number; default is 8081.

By default, the ARIA demo program connects with the robot through the serial port `COM1` under Windows or `/dev/ttyS0` under Linux. And, by default, the demo connects with the laser rangefinder accessory through serial port `COM3` or `/dev/ttyS2`. To change those connection options, either modify the ARIA source code (`examples/demo.cpp` and related files in `src/`) and recompile the demo application, or use a startup argument on the command line. See Table 1.

For example, from the Windows `Start:Run` dialog, choose `Browse…` and select the ARIA demo program: `C:\Program Files\ActivMedia Robotics\ARIA\bin\demo.exe`. Then, type a command line argument at the end of the text in the `Run` dialog as described in Table 1. To connect through the new Ethernet-to-serial radio device over the wireless network, for example, try the command:

C:\Program Files\ActivMedia Robotics\ARIA\bin\demo.exe --remoteHost 192.168.1.32

A Successful Connection

ARIA prints out lots of diagnostic text as it negotiates a connection with the robot. If successful, the client requests various AROS 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. In addition, the motors-associated `STATUS` LED on the User Control Panel should light continuously (was flashing slowly while awaiting connection). Note that the ARIA demo automatically engages your robot's motors though a special client command. Normally, the motors are disengaged when first connecting.

The amber `SERIAL` port indicator LEDs on the robot's User Control Panel should blink to indicate ARIA-client to AROS-server communications, too.

Table 2. ARIA demo operation modes

MODE	HOT KEY	DESCRIPTION
`laser`	l	Displays the closest and furthest readings from the robot's laser range finder
`io`	i	Displays the state of the robot's digital and analog-to-digital I/O ports
`position`	p	Displays the coordinates of the robot's position relative to its starting location
`bumps`	b	Displays the status of the robot's bumpers
`sonar`	s	Displays the robot's sonar readings
`camera`	c	Controls and exercises the robot's pan-tilt-zoom robotic camera
`gripper`	g	Controls, exercises, and displays status of the robot's Gripper
`wander`	w	Sends the robot to move around at its own whim, while avoiding obstacles
`teleop`	t	Allows the user to drive and steer the robot via the keyboard or a joystick connected to the computer

OPERATING THE ARIA DEMONSTRATION CLIENT

When connected with the ARIA demo client, your robot becomes responsive and intelligent. For example, it moves cautiously. Although it may drive toward an obstacle, your *Activ*Media robot will not crash because the ARIA demo includes obstacle-avoidance behaviors which enable the robot to detect and actively avoid collisions.

The ARIA demo displays a menu of robot operation options. The default mode of operation is `teleop`. In `teleop` mode, you drive the robot manually, using the arrow keys on your keyboard or a joystick connected to the client PC's joystick port (as opposed to a joystick port on the robot).

Table 2. Keyboard teleoperation

KEY	ACTION
↑	Increment forward velocity
↓	Decrement forward velocity
←	Incremental left turn
→	Incremental right turn
space	All stop

While driving from the keyboard, each keypress speeds the robot forward or backward or incrementally changes its direction incrementally. 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.

The other modes of ARIA demo operation give you access to your robot's various sensors and accessories, including encoders, sonar, laser, Gripper, a pan-tilt-zoom robotic camera, I/O port states, bumpers, and more. Accordingly, use the ARIA demo not only

as a demonstration tool, but as a diagnostic one, as well, if you suspect a sensor or effector has failed or is working poorly.

Access each ARIA demo mode by pressing its related hot-key;'t', for instance, to select teleoperation. Each mode includes onscreen instructions and may have sub-menus for operating of the respective device.

DISCONNECTING

When you finish, press the `Esc` key to disconnect the ARIA client from your robot server and exit the ARIA demonstration program. Your ActivMedia robot should disengage its drive motors and stop moving, and its sonar should stop firing. You may now slide the robot's `Main Power` switch to `OFF`.

QUICKSTART TROUBLESHOOTING

Most problems occur when attempting to connect the ARIA client with a robot for the first time. The process can be daunting if you don't make the right connections and installations.

ATTENTION!

The ARIA-to-robot connection is ***SERIAL*** *only*. Accordingly, run the ARIA demo client with the onboard or piggyback computer, over radio modems, or over the network with the radio Ethernet-to-serial device.

Proper Connections

Make sure you have ARIA properly installed and that your robot and connections are correct. A common mistake with Linux is not having the proper permissions on the connecting serial port.

Make sure your robot's batteries are fully charged (battery LED green). The robot servers shut down and won't allow a connection at under 10.5 volts.

If you are using the onboard PC or radios, the serial connection is internal and established at the factory; you should not have problems with those cables. Simply make sure the `RADIO` switch is ON, for example. And remove any serial cable that is plugged into the User Control Panel as it may interfere with internal serial communication.

With other serial connections, make sure to use the proper cable: a "pass-through" one, minimally connecting pins 2, 3, and 5 of your PC's serial port to the HOST radio modem of the pair or to the robot's serial port on the User Control Panel.

If you access the wrong serial port, the ARIA demonstration program will display an error message. If the robot server isn't listening, or if the serial link is severed somewhere between the client and server (cable loose or the radio is off, for instance), the client will attempt "Syncing 0" several times and fail. In that case, `RESET` the robot and check your serial connections. For instance, if you are using radio modems, the DCD lamp on the HOST unit next to your PC should light up. If it doesn't, it means it cannot find the one in the robot.

If for some reason communications get severed between the ARIA client and AROS 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 or access point. 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 ARIA demo or other client application is still active, simply press esc and restart.

SRIsim

To verify proper installation of the software, you might run the robot simulator, SRISim. It is in the same directory as the ARIA demonstration program. Start SRIsim first, then the ARIA demo program. ARIA should successfully connect with the simulator if the software has been installed correctly.

SRIsim looks like a real robot to the ARIA client, so you can operate the demo as you do your own *Activ*Media robot. SRIsim includes simulated worlds and different robot profiles which you select from the Files menu, too, so you can see how different robots might navigate in a real or imagined space.

Chapter 5 Joydrive and Self-Tests

Although not all models come standard with a joystick port, your robot's H8S-based controller has a joystick connector and AROS contains a joydrive server for manual operation.[13] And AROS comes with a short self-test routine for your robot's drive system.

To run in either joydrive or self-test mode, start up or RESET the robot into its AROS wait state. You may press the RESET button at any time to disable self-test and joydrive modes. You have about 10 seconds to engage and complete the joystick calibration and begin driving the robot or to enter into motors self-test mode before the controller automatically cancels joydrive mode and reverts to waiting for a client connection.

You may also enable AROS' joydrive server while connected with a client by sending the client command number 47 with the integer argument 1.

JOYDRIVE MODE

To joydrive your robot when not connected with a client program, switch the robot's Main Power ON or RESET the controller, then press the white MOTORS button on the User Control Panel once. Listen for a rhythmic, low-tone beeping indicating joydrive mode.

To joydrive your robot while it is connected with a client (overrides client-based drive commands for manual operation while recording a map, for instance), you must have the client software send the AROS command #47 with an integer argument 1 to enable the joydrive servers. The first time you press and release the joystick fire button after AROS receives the command, you engage self-calibration mode (see below). Have your client send the AROS joydrive command #47 with an integer argument of 0 to disable the joystick drive-override.

The joystick is self-calibrating: When you first enable joydrive mode, either by the client command or when in self-test joydrive mode, AROS 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 handle find its default centered position before pressing the fire button and starting to joydrive the robot. Try exiting (RESET or client command 47, depending on mode) and restarting joydrive mode if the joystick doesn't seem to function well.

The joystick's fire 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 robot. 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 and even disable joydrive mode, through special AROS FLASH configuration parameters. See Chapter 7, *Updating & Reconfiguring AROS*, for details.

[13] The joystick adaptor kit, including the 15-pin DSUB joystick connector and pull-up resistors, if not installed on your robot, is available for nominal fee through sales@activmedia.com. Also note that this port is different than the USB-based joystick port found on the back of the Laser bracket for the optional equipment and used to manually drive from ARIA-based clients.

ENGAGING SELF-TESTS

To enable self-test mode, press the white MOTORS button twice after startup or RESET.[14]

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

Currently, the only AROS 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 before engaging the self-test.

The motor's 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 H8S-AROS controller reverts to its client-connection wait state after completing the motors self-test.

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

Chapter 6 ActivMedia Robotics Operating System

All *Activ*Media robots use a client-server mobile robot-control architecture originally developed at SRI International, Inc. and Stanford University. In the model, the robot's controller servers 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.

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 *Activ*Media Robotics Operating System (AROS) client-server interface. The same AROS functions and commands are supported in the various client-programming environments that accompany your robot or are available for separate license.

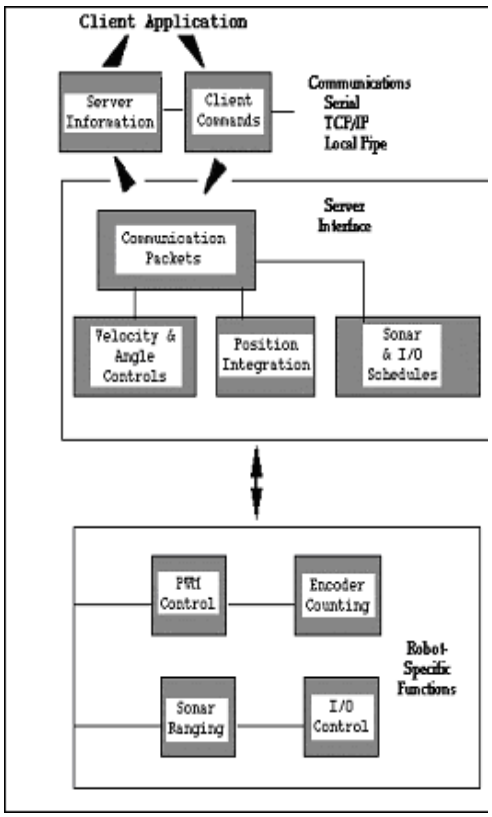

Figure 16. ActivMedia Robotics client-server control architecture

Experienced *Activ*Media robot users can be assured that AROS is upwardly compatible with all *Activ*Media robots, implementing the same commands and information packets that first appeared in the Pioneer 1-based PSOS and in the original Pioneer 2-based P2OS. AROS, of course, extends the servers to add new functionality, improve performance, and provide additional information about the robot's state and sensing.

CLIENT-SERVER COMMUNICATION PACKET PROTOCOLS

*Activ*Media robots communicate with a control client using special client-server communication packet protocols, one for command packets from client to server and another for server information packets (SIPs) from the server to client. Both protocols are bit streams consisting of five main elements: a two-byte header, a one-byte count of the number of subsequent packet bytes, the client command or SIP packet type, command data types and argument values or SIP data bytes, and, finally, a two-byte checksum. Packets are limited to a maximum of 206 bytes each.

The two byte header which signals the start of a packet is the same for both client-command packets and SIPs: 0xFA, 0xFB. The byte count value counts the number of all subsequent bytes in the packet including the checksum, but not including the byte count value itself or the header bytes.

Data types are simple and depend on the element (see descriptions below): client commands, SIP types, and so on, are single 8-bit bytes, for example. Command arguments and SIP values may be 2-byte integers, ordered as least-significant byte

always first. Some data are strings of up to a maximum 200 bytes, prefaced by a length byte. Unlike common data integers, the two-byte checksum appears with its most-significant byte first (opposite order).

Packet Checksum

Calculate the PSOS/P2OS/AROS client-server packet checksum by successively adding data byte pairs (high byte first) to a running checksum (initially zero), disregarding sign and overflow. If there are 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: The checksum integer is placed at the end of the packet, with its bytes in the reverse order of that used for data integers; that is, b_0 is the high byte and b_1 is the low byte.

Packet Errors

AROS ignores a client command packet whose byte count exceeds 204 (total packet size of 206 bytes) or has an erroneous checksum. The client should similarly ignore erroneous SIPs.

AROS does not acknowledge receipt of a command packet nor does it have any facility to handle client acknowledgment of a SIP. Accordingly, when designing client applications, keep in mind serial communication limitations, particularly data rates and physical linkage. Communication between an onboard PC client connected with the server via a signal cable is much more reliable than over radios, for example. And don't expect to send a client command every millisecond if the HOST serial port's baud rate is set to 9,600 kbps.

Because of the real-time nature of client-server mobile-robotics interactions, we made a conscious decision to provide an unacknowledged communication packet interface. Retransmitting server information or command packets would serve no useful purpose, because old data would be virtually useless in maintaining responsive robot behaviors.

Nonetheless, the 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 respective SIPs, client software may detect ignored commands and re-issue them until achieving the correct state.

SERVER INFORMATION PACKETS

Like its PSOS and P2OS predecessors, AROS automatically and repeatedly sends a packet of information over its HOST serial port to a connected client. The standard AROS SIP informs the client about a number of operating states and readings, using the order and data types described in the nearby Table.

Table 3. Standard Server Information Packet

NAME	VALUE	DESCRIPTION
HEADER	int	Exactly 0xFA, 0xFB
BYTE COUNT	byte	Number of data bytes + 2 (checksum), not including header or byte-count bytes
STATUS/PACKET TYPE	0x3S =	Motors status
	2	Motors stopped
	3	Robot moving
XPOS	int	Wheel-encoder integrated coordinates; platform-dependent units; multiply by *DistConvFactor*‡ to convert to millimeters.
YPOS	int	
THPOS	sint‡	Orientation in platform-dependent units—multiply by *AngleConvFactor*‡ for degrees.
L VEL	sint	Wheel velocities in mm/sec (*VelConvFactor*‡ = 1.0)
R VEL	sint	
BATTERY	byte	Battery charge in tenths of volts (101 = 10.1 volts, for example)
STALL AND BUMPERS	int	Motor stall and bumper indicators. Bit 0 is the left wheel stall indicator, set to 1 if stalled. Bits 1-7 correspond to the first bumper I/O digital input states (accessory dependent). Bit 8 is the right wheel stall, and bits 9-15 correspond the second bumper I/O states, also accessory and application dependent.
CONTROL	sint	Setpoint of the server's angular position servo—multiply by *AngleConvFactor*‡ for degrees
FLAGS	sint	Bit 0 motors status; bits 1-4 sonar array status; bits 5,6 M-STOP; bits 7,8 ledge-sense IRs; bit 9 joystick button; bit 10 auto—charger power-good.
COMPASS	byte	Electronic compass accessory heading in 2-degree units
SONAR COUNT	byte	Number of new sonar readings included in SIP
NUMBER	byte	If Sonar Count>0, is sonar disc number 0-31; reading follows
RANGE	int	Sonar range value; multiply by RangeConvFactor‡
…REST OF THE SONAR READINGS…		
GRIP_STATE	byte	Gripper state byte.
ANPORT	byte	Selected analog port number 1-5
ANALOG	byte	User Analog input (0-255=0-5 VDC) reading on selected port
DIGIN	byte	Byte-encoded User I/O digital input
DIGOUT	byte	Byte-endcoded User I/O digital output
CHECKSUM	integer	Packet-integrity checksum

‡ Client-side data-conversion factor. Consult the ARIA parameter file your robot.
‡ Explicitly, a signed integer.

AROS also supports several additional SIP types. These include an "alternative" SIP that currently is not supported by Saphira or ARIA.[15] See following sections in this chapter for a description of the extended SIP types.

[15] Indeed, if enabled, the alternative SIP apparently will "break" the client software. Read carefully.

CLIENT COMMANDS

AROS has a structured command format for receiving and responding to directions from a client for control and operation of your *Activ*Media robot or the simulator. Client commands are comprised of a one-byte command number optionally followed, if required by the command, by a one-byte description of the argument type and then the argument value.

Table 4. AROS/P2OS/PSOS client command packet protocol

COMPONENT	BYTES	VALUE	DESCRIPTION
HEADER	2	0xFA, 0xFB	Packet header; same for client and server
BYTE COUNT	1	N	Number of command/argument bytes plus Checksum's two bytes, but not including Byte Count itself or the header bytes. Maximum of 249.
COMMAND NUMBER	1	0 - 255	Client command number; see Table 7.
ARGUMENT TYPE	1	0x3B or 0x1B or 0x2B	Required data type of command argument: positive integer, negative or absolute integer, or string (ARGSTR)
ARGUMENT	n	data	Command argument; always 2-byte integer or string containing length prefix
CHECKSUM	2	computed	Packet integrity checksum

Table 5. AROS/P2OS/PSOS command set

COMMAND	#	ARGS	DESCRIPTION	AROS	P2OS	PSOS
			Before Client Connection			
SYNC0	0	none	Start connection. Send in sequence. AROS echoes synchronization commands back to client, and robot-specific autosynchronization after SYNC2.	1.0	1.0	3.x
SYNC1	1	none				
SYNC2	2	none				
			After Established Connection			
PULSE	0	none	Resets server watchdog	1.0	1.0	3.x
OPEN	1	none	Starts the AROS servers	1.0	1.0	3.x
CLOSE	2	none	Close servers and client connection	1.0	1.0	3.x
POLLING	3	string	Change sonar polling sequence (see text)	1.0	1.0	3.9
ENABLE	4	int	1=enable; 0=disable the motors	1.0	1.0	–
SETA	5	sint	Translational acceleration, if positive, or deceleration, if negative; mm/sec/sec	1.0	1.0	–
SETV	6	int	Sets maximum translational velocity; mm/sec	1.0	1.0	4.8
SETO	7	none	Resets local position to 0,0,0 origin	1.0	1.0	3.x
MOVE	8	sint	Translate (+) forward or (-) back mm distance	1.0	1.0	–
ROTATE	9	sint	Rotate (+) counter- or (-) clockwise degrees/sec	1.0	1.0	–
SETRV	10	int	Sets maximum rotational velocity; degrees/sec	1.0	1.0	4.8
VEL	11	sint	Translate at mm/sec forward (+) or backward (-)	1.0	1.0	3.x
HEAD	12	sint	Turn to absolute heading; ±degrees (+ = ccw)	1.0	1.0	4.2
DHEAD	13	sint	Turn relative to current heading; (+) counter- or (–) clockwise degrees	1.0	1.0	3.x
SAY	15	string	As many as 20 pairs of duration (20 ms increments) /tone (half-cycle) pairs	1.0	1.0	4.2
CONFIG	18	none	Request configuration SIP	1.0	1.4	–
ENCODER	19	int	Request one (1), a continuous stream (>1), or stop (0) encoder SIPs	1.0	1.4	–
RVEL	21	sint	Rotate at (+) counter- or (–) clockwise; degrees/sec	1.0	1.0	4.2
DCHEAD	22	sint	Heading setpoint relative to last setpoint;	1.0		

DCHEAD	22	sint	Heading setpoint relative to last setpoint; ± degrees (+ = ccw)	1.0		
SETRA	23	sint	Rotational (+)acceleration or (-)deceleration, in degrees/sec/sec	1.0	1.0	–
SONAR	28	int	1=enable, 0=disable all the sonar; otherwise, use bit 0 to enable (1) or disable (0) a particular array 1-4, as specified in argument bits 1-4.	1.0	1.0	–
STOP	29	none	Stops robot; motors remain enabled	1.0	1.0	–
DIGOUT	30	2 bytes	Bits 8-15 is a byte mask that selects the output port(s); Bits 0-7 set (1) or reset (0) the selected port(s).	1.7[16]	1.2	4.2
VEL2	32	2 bytes	Independent wheel velocities; Bits 0-7 for right wheel, Bits 8-15 for left wheel; PSOS is in ±4mm/sec; AROS/P2OS in 20mm/sec increments	1.0	1.0	4.1
GRIPPER	33	int	Gripper server commands. See the Pioneer 2 Gripper or PeopleBot manual for details.	1.0	1.3	4.0
ADSEL	35	int	Selects the A/D port number for reporting Anport value in standard SIP.	1.0	1.2	–
GRIPPERVAL	36	int	Gripper server values. See Pioneer 2 Gripper or PeopleBot manual for details.	1.0		–
GRIPREQUEST	37	none	Request one (1), a continuous stream (>1), or stop (0) Gripper SIPs. See Pioneer 2 Gripper or PeopleBot manual for details.	1.0	1.E	–
IOREQUEST	40	none	Request one (1), a continuous stream (>1), or stop (0) IO SIPs	1.0	1.E	–
PTUPOS	41	2 bytes	Msbyte is port number (1-4), lsbyte is pulse width in 100µsec units PSOS or 10µsec units P2OS	–	1.2	4.5
TTY2	42	string	Sends string argument to serial device connected to AUX (AUX1 on H8S) port	1.0	1.0	4.2
GETAUX	43	int	Request to retrieve 1-200 bytes from the AUX (AUX1 on H8S) serial port; 0 flushes the buffer.	1.0	1.4	–
BUMP_STALL	44	int	Stall robot if front (1), rear (2) or either (3) bumps contacted. Off is 0. See BumpStall FLASH for default.	1.0	1.5	–
TCM2	45	int	TCM2 Module commands; see *TCM2 Manual* for details.	1.0	1.6	–
DOCK	46	int	Default is 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 while also connected with a client	1.0	1.G	–
SONAR_CYCLE	48	int	Change the sonar cycle time; arg in milliseconds	1.8	–	–
HOSTBAUD	50	int	Reset the HOST serial port baud rate to 0=9600, 1=19200, 2=38400, 3=57600, or 4=115200	1.8	–	–
AUX1BAUD	51	int	Resets the AUX1 serial port baud rate	1.8	–	–
AUX2BAUD	52	int	Resets the AUX2 serial port baud rate	1.8	–	–
E_STOP	55	none	Emergency stop, overrides deceleration	1.0	1.8	–
M_STALL	56	int	1 (default)=Motors stop button causes stall; 0 (P2OS default)=off	1.0	1.E	–
LEDGE	57	int	0 if inactive; 1 if stop when near-IRs triggered; 2 if impose speed control only; 3 if both stop and speed control	1.5	–	–
STEP	64	none	Single-step mode (simulator only)	1.0	1.0	3.x
TTY3	66	string	Sends string argument to serial device connected	1.0	–	–

[16] No, this isn't a misprint—the DIGOUT command was mistakenly omitted until version 1.7.

			to AUX2 H8S serial port			
GETAUX2	67	int	Request to retrieve 1-200 bytes from the AUX2 H8S serial port; 0 flushes the buffer.	1.1	–	–
CHARGE	68	int	1 to deploy autocharging mechanism; 0 to retract	1.7	–	–
ARM	70 - 81	int	Arm-related commands; see manual for details	1.3	–	–
ROTKP	82	int	Change working rotation Proportional PID value (not FLASH default)	1.1	1.M	–
ROTKV	83	int	Change working rotation Derivative PID value (not FLASH default)	1.1	1.M	–
ROTKI	84	int	Change working rotation Integral PID value (not FLASH default)	1.1	1.M	–
TRANSKP	85	int	Change working translation Proportional PID value (not FLASH default)	1.1	1.M	–
TRANSKV	86	int	Change working translation Derivative PID value (not FLASH default)	1.1	1.M	–
TRANSKI	87	int	Change working translation Integral PID value (not FLASH default)	1.1	1.M	–
REVCOUNT	88	int	Change working differential encoder count (not FLASH default)	1.1	1.M	
PLAYLIST	91	none	Request AmigoBot sounds playlist packet	1.0	1.E	–
SOUNDTOG	92	int	0=mute or 1=enable buzzer	1.0	–	–
SHUTDOWN	250	int	0=cancel shutdown; 1=simulate low-power condition; 2=initiate computer shutdown	1.6	–	–

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. AROS' command processor runs on a one millisecond interrupt cycle, but the server response speed depends on the command. Typically, limit client commands to a maximum of one every 3-5 milliseconds or be prepared to recover from lost commands.

THE CLIENT-SERVER CONNECTION

Before exerting any control, a client application must first establish a connection with the robot server via a serial link through the robot controller's `HOST` port. After establishing the communication link, the client then sends commands to and receives operating information from the server.

When first started or reset, AROS is in a special wait state, listening for communication packets to establish a client-server connection.[17] To establish a connection, the client application must send a series of three synchronization packets containing the `SYNC0`, `SYNC1`, and `SYNC2` commands in succession, and retrieve the server responses.

Specifically, and as examples of the client command protocol, the synchronization sequence of bytes is (in hexadecimal notation):

```
SYNC0:  0xFA, 0xFB, 0x03, 0x00, 0x00, 0x00
SYNC1:  0xFA, 0xFB, 0x03, 0x01, 0x00, 0x01
SYNC2:  0xFA, 0xFB, 0x03, 0x02, 0x00, 0x02
```

When in wait mode, AROS echoes the packets verbatim 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.

[17] There also is monitor mode for AROS downloads and parameter updates; see next chapter for details.

Autoconfiguration (SYNC2)

AROS automatically sends robot configuration information back to the client following the last synchronization packet (`SYNC2`). The configuration values are three NULL-terminated strings that comprise the robot's FLASH-stored `name`, `class`, and `subclass`. You may uniquely `name` your *Activ*Media robot with the FLASH configuration tool we provide. The `class` and `subclass` are constants normally set at the factory and not changed thereafter. (See next chapter for details.)

The `class` string typically is `Pioneer`. The `subclass` depends on your robot model; `P2D8` or `P2AT8`, for example. Clients may use these identifying strings to self-configure their own operating parameters. ARIA, for instance, loads and uses the robot's related parameters file found in the special `Aria/params` directory.

Opening the Servers—OPEN

Once you've established a connection with AROS, your client should send the `OPEN` command #1 (no argument; 0xFA, 0xFB, 0x03, 0x01, 0x00, 0x01) to the server, which causes the *Activ*Media robot controller to perform a few housekeeping functions, start its various servers, such as for the sonar and motor controllers, listen for client commands, and begin transmitting server information to the client.

Note that once connected, your robot's motors are disabled, regardless of their state when last connected. To enable the motors after starting a connection, you must either do it manually (press the black MOTORS/TEST button) or have your client send an ENABLE client command #4 with an integer argument of 1 (0xFA, 0xFB, 0x06, 0x04, 0x3B, 0x01, 0x00, 0x05, 0x3B).

Keeping the Beat—PULSE

A safety watchdog expects that, once connected, your robot's 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.

Some clients—ARIA-based ones, for instance—use the good practice of sending a `PULSE` command #0 (no argument; 0xFA, 0xFB, 0x03, 0x00, 0x00, 0x00) just after opening the AROS servers. And if your client application will be otherwise distracted for some time, periodically issue the `PULSE` command to let your robot server know that your client is indeed alive and well. It has no other effect.

If the robot shuts down due to lack of communication with the client, 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, which automatically disables the motors and other server functions like sonar, simply issue the client `CLOSE` command #2 (no argument; 0xFA, 0xFB, 0x03, 0x02, 0x00, 0x02).

Once connected, send the `ENABLE` command
or press the white `MOTORS` button on the User Control Panel
to enable your robot's motors.

With AROS versions 1.3 and later, many of the controller's operating parameters return to their FLASH-based default values upon disconnection with the client.[18] For example, if the FLASH default for the maximum velocity is 1000 millimeters per second, and your client uses the SETV command #6 to reset the maximum velocity to 500 millimeters per second, the maximum velocity automatically will revert back to 1000 after your client disconnects and then reconnects for a subsequent session.

MOTION COMMANDS

The AROS motor-control servers accept several different client-motion commands of two mutually exclusive types: either independent-wheel or platform translational/rotational movements. The AROS servers automatically abandon any translational or rotational setpoints and switch to independent wheel-velocity controls when your client issues the independent-wheel `VEL2` command #32, and vice versa.

Note that once connected, *Activ*Media robots' motors are *disabled*, regardless of their state when last connected. Accordingly, you must either enable the motors manually (white `MOTORS` button on the User Control Panel) or send the motors `ENABLE` client command #4 with the argument value of one.[19] Monitor the status of the motors with bit 0 of the `Flags` integer in the standard SIP.

When in independent-wheel velocity mode (`VEL2`), the robot's motion-control servers do their best to maintain precise wheel velocities. In practice, wheel slippage and uneven terrain will cause the robot to change heading, which your client must detect and compensate. When in translational/rotational (TR) motion control mode (recommended), your robot's servers work to maintain both platform speed and heading.

Table 6. AROS motion commands

Rotation	
HEAD (#12)	Turn to absolute heading at SETRV max velocity
DHEAD (#13), DCHEAD (#22)	Turn to heading relative to control point at SETRV max velocity
ROTATE (#9)	Rotate at SETRV velocity
Translation	
VEL (#11)	Translate forward/reverse at prescribed velocity (SETV maximum)
MOVE (#8)	Translate distance at SETV max velocity
Independent	**Wheel**
VEL2 (#32)	Set velocity for each side of robot (SETV maximum)

[18] With earlier versions, the changes persisted between sessions, and reverted to the FLASH defaults only after the controller was reset.

[19] Alternatively, disable the motors with the `ENABLE` command argument of zero.

ActivMedia Robots in Motion

*Activ*Media robots use position, as opposed to velocity, motion controls to translate the platform a certain distance and turn it to a particular heading. To achieve constant translational (VEL), rotational (ROTATE), or independent-wheel (VEL2) velocities, the servers simply set the target position well ahead of the robot's current position.

When the robot controller receives a motion command, it accelerates or decelerates the robot at the translational SETA (#5) (TR and VEL2 modes) and rotational SETRA (#23; TR mode only) rates until the platform either achieves its SETV (#6) maximum translational and SETRV (#10) maximum rotational speeds, or nears its goal. Accordingly, rotational headings and translational setpoints are achieved by a trapezoidal velocity function, which AROS recomputes each time a new motion command is received.[20]

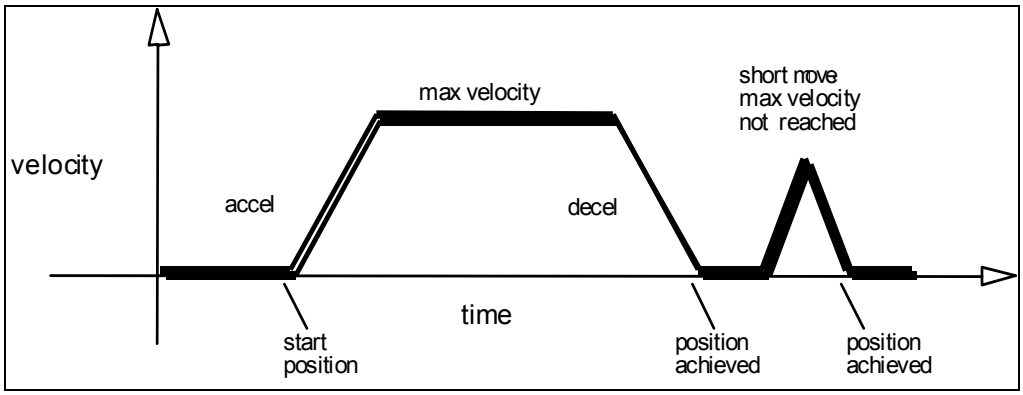

Figure 17. ActivMedia robot's trapezoidal velocity profile

AROS automatically limits VEL2-, VEL-, and RVEL-specified velocities to previously imposed, client-modifiable SETVEL and SETRV maximums, and ultimately by absolute, platform-dependent, FLASH-embedded constants. Similarly, the distinct acceleration and deceleration parameters for both translation and rotation are limited by FLASH constants. AROS initializes these values upon controller startup or reset from related FLASH parameters. The speed limits, either from FLASH or when changed by SETV or SETRV commands, take effect on subsequent commands, not previously established velocity or heading setpoints. And the maximums persist across client-server connection sessions until the controller is reset.

Note that the E_STOP command #55 or the STOP button that is found on some *Activ*Media robots override deceleration and immediately stop the robot in the shortest distance and time possible. 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 or until the STOP button is reset. (See *E_STOP and E_STALL* later in this chapter.)

Platform Dependent and Independent Variables

All client-side motion command arguments use robot-independent units of measure, in millimeters or degrees. AROS converts these command arguments into robot-dependent, wheel encoder-related motion values using two, user-settable parameters: ticksmm, for translation, and revcount for rotation.

[20] Note that acceleration and deceleration are distinct values, settable via SETA for translation and SETRA for rotation.

At the same time, AROS reports back to the client in the standard SIP the robot's position and speed. Not all robots convert these values into platform-independent units. ARIA and Saphira clients rely on conversion factors found in your robot's respective ".p" parameter file to make the necessary conversion.

So when you tell the robot to `move` a certain number of millimeters forward, measure its actual travel with a meter tape and adjust `ticksmm` accordingly. Similarly, turn the robot and adjust `revcount` to achieve the correct heading.

Then, when you are satisfied that the robot moves and turns precisely, adjust the various parameter file-based conversion factors, such as `DistConvFactor`, so that the client reports the robot's position and speeds in platform-independent units.

Please see the next chapter for a detailed description of these platform-dependent variables.

PID Controls

The AROS drive servers use a common Proportional-Integral-Derivative (PID) control system to adjust the PWM pulse width at the motor drivers and subsequent power to the motors. The motor-duty cycle is 200 microseconds; pulse-width is proportional 0-500 for 0-100% of the duty cycle.

The AROS drive servers recalculate and adjust your robot's trajectory and speed every five milliseconds based on feedback from the wheel encoders.

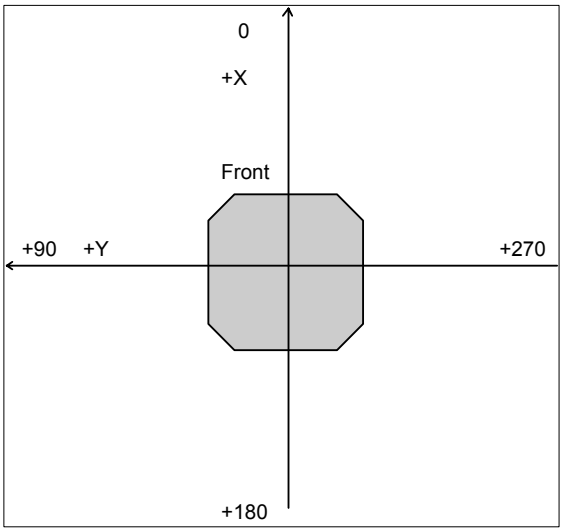

Figure 18. Internal coordinate system

The default PID values for translation and rotation and maximum PWM are stored as FLASH parameters in your robot's H8S microcontroller and may be changed. You also may temporarily update the PID values with the AROS client commands #84 through #87. On-the-fly changes persist until the controller is reset. The translational PID values apply to independent wheel-velocity mode.

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 a 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.

Position Integration

*Activ*Media robots, including Pioneer 2s and 3s, track their position and orientation based on dead-reckoning from wheel motion derived from encoder readings. The robot maintains its *internal coordinate position* in platform-dependent units, as reported in the standard SIP (Xpos, Ypos, and Thpos).

Be aware that with the simulator as well as with real robots, 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 with respect to 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.

You may reset the internal coordinates to 0,0,0 with the `SETO` command #7.

SONAR

When connected with and opened by the client, AROS automatically begins firing your robot's sonar, one disc each simultaneously for each array, as initially sequenced and enabled in your robot's FLASH parameters. The sonar servers also begin sending the sonar-ranging results to the client via the standard SIP.

Enable/Disabling Sonar

Use the `SONAR` client command #28 to enable or disable all or individual sonar arrays. Set ("1") bit zero of the SONAR argument to enable or reset it ("0") to disable the sonar pinging. Set argument bits two through four to an individual array number one through four to enable or disable only that array. Array zero, the form of the P2OS command, affects all the arrays at once.

For example, an argument value of one enables all the sonar arrays, whereas an argument value of six silences array number three. Monitor the status of the sonar arrays in the `FLAGS` integer of the standard SIP.

Polling Sequence and Rate

Each array's sonar fire at a rate and in the sequence defined in your robot's FLASH parameters. (Consult the next chapter on how to change the FLASH settings.) Use the sonar `POLLING` command #3 to have your client change the firing sequence, and the `SONAR_CYCLE` command #48 to change the rate. The changes persist until you reset the controller or restart the client-server connection.

The `POLLING` command string argument consists of a sequence of sonar numbers one through 32. Sonar numbers one through eight get added to the polling sequence for

sonar array number one; numbers nine through 16 get added to the sequence for sonar array number two; 17-24 specify the sequence for array three; and 25-32 are for array four. You may include up to 16 sonar numbers in the sequence for any single array. Only those arrays whose sonar numbers appear in the argument get re-sequenced. You may repeat a sonar number two or more times in a sequence. If a sonar number does not appear in an otherwise altered sequence, the disc will not fire.

Note that for compatibility with earlier *Activ*Media robot operating systems, if the string is empty, all the sonar get disabled, but their polling sequences remain unaltered, just as if you had sent the SONAR command with an argument value of zero.

In earlier versions of AROS and P2OS, the sonar polling rate is fixed: one sonar per array gets polled every 40 milliseconds. That common cycle timing accommodates ranging out to the maximum of the sonar of several meters for general applications, including features recognition and localization. For other applications, such as close-in obstacle avoidance, a shorter range but faster rate of update is better.

Hence, we introduce in AROS v1.8 the SonarCycle FLASH parameter which lets you set, through AROScf, the default sonar cycle time, in milliseconds. Use the SONAR_CYCLE client command #48 to change the cycle timing on the fly to the command integer's argument value in milliseconds.

STALLS AND EMERGENCIES

With a robot equipped with forward and/or rear bumpers, by default AROS immediately stops the robot and notifies the client of a stall if any one or more of the contact sensors get triggered and the robot is going in the direction of the bump (forward/front or backward/rear). Send the BUMPSTALL command #44 with an integer argument of zero to disable that bump-stall behavior. Give the argument value of one to re-enable BUMPSTALL only when a forward bump sensor gets triggered; two for rear-only BUMPSTALLs; or three for both rear and forward bump contact-activated stalls.

Change AROS' bump-stall behavior default with the BumpStall FLASH parameter.

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 BUMPSTALL, use AROS' built-in E_STALL feature to simulate a stall when someone presses the robot's STOP button.[21] An integrated switch in the STOP button toggles a dedicated digital I/O port (Port A, bit 3) on the microcontroller thereby notifying AROS of the condition. AROS stops the robot's motors, puts on the brakes, and throws continuous stalls.

Unlike other stalls, E_STALL also disables the motors. You must either re-enable the motors manually (MOTORS button) or programmatically (ENABLE com-mand #4).

Table 7. The FLAGS bits in the standard SIP

BIT	CONDITION IF SET
0	Motors enabled
1	Sonar array #1 enabled
2	Sonar array #2 enabled
3	Sonar array #3 enabled
4	Sonar array #4 enabled
5	STOP button pressed
6	E_stall engaged
7	Far ledge detected (IR)
8	Near ledge detected (IR)
9	Joystick button 1 pressed
10	Recharging "power-good"
11-15	Reserved

The E_STALL server notifies your client software through the stall bytes and in bit 5 of the FLAGS byte in the standard so that your client may respond to a STOP E_STALL differently than a regular stall.

[21] Available only on some robots.

Normally enabled (default was disabled in P2OS), change E_STALL by sending the AROS command #56. With argument of zero, E_STALL gets disabled. An argument value of one re-enables E_STALL.

ACCESSORY COMMANDS AND PACKETS

Several types of alternative server information packets (SIPs) come with AROS to better support the *Activ*Media Robotics community. On request from the client by a related AROS command, the AROS server packages and sends one or a continuous stream of information packets to the client over the HOST serial communication line. Extended packets get sent immediately after the standard SIP that AROS sends to your client every SIP milliseconds (typically 100).[22]

The standard SIP takes priority and gets sent as soon as the communication port is free and the cycle timer expires. So you may have to adjust the communications baud rate to accommodate all data packets in the allotted cycle time, or some packets may never get sent.

Packet Processing

Identical with the standard SIP, all AROS 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.

ARIA, for example, comes with a framework for packet parsing and has an internal parser for the PSOS/P2OS/AROS packet type 0x3S (S=0-2; aka "standard" SIP), as well as for some of the extended packets we describe in this section.

Table 8. CONFIGpac contents (AROS v1.5 and later)

LABEL	DATA	DESCRIPTION
HEADER	int	Common packet header = 0xfAFB
TYPE	byte	IDs ENCODERpac = 0x20
BYTE COUNT	byte	Number of following data bytes
ROBOT TYPE	str	Typically "Pioneer"
SUBTYPE	str	Identifies the *Activ*Media robot model; e.g. "p3dx",
SERIALNUM	str	Serial number for the robot.
4MOTS	byte	Antiquated (=1 if AT with P2OS)
ROTVELTOP	int	Maximum rotational velocity; deg/sec
TRANSVELTOP	int	Maximum translation speed; mm/sec
ROTACCTOP	int	Maximum rotation (de)acceleration; deg/sec^2
TRANSACCTOP	int	Maximum translational (de)acceleration; mm/sec^2
PWMMAX	int	Maximum motor PWM (500=fully on).
NAME	str	Unique name given to your robot.
SIP	byte	Server information packet cycle time
HOSTBAUD	byte	Baud rate for client-server HOST serial: 0=9.6k, 1=19.2k, 2=38.4k, 3=56.8k, 4=115.2k.
AUXBAUD	byte	Baud rate for AUX serial port 1; see HostBaud
GRIPPER	int	0 if no Gripper; else 1
FRONT SONAR	int	1 if robot has front sonar array enabled, else 0
REAR SONAR	byte	1 if robot has rear sonar enabled, else 0
LOWBATTERY	int	In 1/10 volts; alarm activated when battery charge falls below this value.
REVCOUNT	int	Current number of differential encoder ticks for a 360 degree revolution of the robot.
WATCHDOG	int	Ms time before robot automatically stops if it has not

[22] You may have to adjust the HOST serial baud rate to accommodate the additional communications traffic.

		received a command from a client. Restarts on restoration of connection.
P2MPACS	byte	1 enables alternative SIP.
STALLVAL	int	Maximum PWM before stall. If > PwmMax, never.
STALLCOUNT	int	Ms time after a stall for recovery. Motors not engaged during this time.
JOYVEL	int	Joystick translation velocity setting, mm/sec
JOYRVEL	int	Joystick rotation velocity setting in deg/sec
ROTVELMAX	int	Current max rotational speed; deg/sec.
TRANSVELMAX	int	Current max translational speed; mm/sec.
ROTACC	int	Current rotational acceleration; deg/sec^2
ROTDECEL	int	Current rotational deceleration; deg/sec^2
ROTKP	int	Current Proportional PID for rotation
ROTKV	int	Current Derivative PID for rotation
ROTKI	int	Current Integral PID for rotation
TRANSACC	int	Current translational acceleration; mm/sec^2
TRANSDECEL	int	Current translational deceleration; mm/sec^2
TRANSKP	int	Current Proportional PID for translation
TRANSKV	int	Current Derivative PID for translation
TRANSKI	int	Current Integral PID for translation
ADDED IN AROS 1.6		
FRONTBUMPS	byte	Number of front bumper segments
REARBUMPS	byte	Number of rear bumper segments
ADDED IN AROS 1.7		
CHARGER	byte	1 if P3 or 2 if PowerBot automated charger mechanism and circuitry installed in robot; otherwise 0
ADDED IN AROS 1.8		
SONARCYCLE	byte	Sonar duty cycle time in milliseconds
AUTOBAUD	byte	1 if can change baud rates; 2 if auto-baud implemented
HASGYRO	byte	1 if has the gyro device; otherwise 0

CONFIGpac and CONFIG Command

Send the CONFIG command #18 without an argument to have AROS send back a CONFIGpac SIP packet type 32 (0x20) server information packet containing the robot's operational parameters. Use the CONFIGpac to examine many of your robot's default FLASH_based settings or their working values, when appropriate, as changed by other client commands, such as SETV and ROTKV. A table nearby gives details about the configuration packet data.

SERIAL PORT COMMUNICATIONS

AROS provides two-way communications through the HOST client-server communication port and to and through two auxillary serial ports on the microcontroller, AUX1 and AUX2.

Changing Baud Rates and Autobauding

The baud rates for the HOST, AUX1, and AUX2 ports initially are set from their respective FLASH-based defaults and get reset to those values whenever the controller is reset or upon client disconnection. For advanced serial port management from the client side, in AROS 1.8 and later we provide three client commands which let your software reset the HOST (HOSTBAUD #50), AUX1 (AUX1BAUD #51), and AUX2 (AUX2BAUD #52) serial port baud rates, respectively. Use the integer command argument values: 0=9600, 1=19.2K, 2=38.8K, 3=57.6K or 4=115.2K baud.

For auto-baud, the HOST serial port automatically reverts to its FLASH default baud rate if, after being reset by the HOSTBAUD client command, it does not receive a subsequent and valid client-command packet within 500 milliseconds.

HOST-to-AUX Serial Transfers

Use the client-side TTY2 command #42 with a string argument to have that string sent out the AUX1 port to the attached serial device, such as a robotic camera. Similarly, use the TTY3 command #66 to send a string argument to the AUX2 port.

AROS also maintains two circular buffers for incoming serial data from the AUX1 and AUX2 ports. On request, AROS sends successive portions of the buffers to your client via the HOST port in the respective SERAUXpac (type = 176; 0xB0) and SERAUX2pac (type = 184; 0xB8) SIPs. Use the GETAUX command 43 for AUX1 or GETAUX2 command number 67 for AUX2. Use the integer argument value of zero to flush the contents of the respective buffer. Use an argument value of up to 253 bytes to have AROS wait to collect the requested number of incoming AUX-port serial bytes and them send them in the respective SERAUXpac or SERAUX2pac SIP.

ENCODER PACKETS

Issue the ENCODER command #19 with an argument of one for a single, or with an argument value of two or more for a continuous stream of ENCODERpac (type 144; 0x90) SIPs. Discontinue the packets with the ENCODER command #19 with an argument of zero.

Table 9. ENCODERpac SIP 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 accumulated encoder counts from the left wheel
Right Encoder	integer	Least significant, most significant portion of the
	integer	current accumulated encoder counts from the left wheel
Checksum	integer	Checksum for packet integrity

Gripper packets

AROS controls the Gripper accessory for the Pioneer and Performance PeopleBot robots. The client sends commands to the Gripper servers and gets Gripper status information from the standard SIP. Please consult the respective manuals for details.

Table 10. GRIPPERpac packet contents

HEADER	int	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=PeopleBot
GRIP_STATE	byte	See *nearby Table*
GRASP_TIME	byte	MS time controls grasping pressure
CHECKSUM	integer	Computed checksum

AROS supports a GRIPPERpac (type=224; 0xE0) packet type and related GRIPREQUEST P2OS command #37 to retrieve setup and status information from the servers.

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

Table 11. 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

Note that the Gripper status information bits 0-5 also may be obtained from the respective `digin` and `digout` values of the standard SIP as related to the User I/O port states. See *Appendix A* for connection details.

Sounds

Unlike its *Activ*Media robot cousins, the AmigoBot mobile robot has onboard sound reproduction hardware and software that includes a playlist of contents. To support the *Activ*Media Robotics Interface for Applications (ARIA) that includes all *Activ*Media's robots, we've included the `PLAYLISTpac` (type = 208; 0xD0) and `PLAYLIST` request command 91 in AROS. We document the command and packet here for completeness, but they have no effect on the operation or performance of your ActivMedia mobile robot.

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. Upon receipt of the `PLAYLIST` command 91 with any or no argument, AROS responds with a PLAYLISTpac SIP containing 25 NULL bytes, telling the client that your AROS-based robot does not have any onboard sounds.

Whereas the AmigoBot has a high-fidelity sound system, AROS- and P2OS-based robots have a piezo buzzer that aurally notifies you of system conditions, such as low battery or stalls. For stealthy operation, issue the SOUNTOG command number 92 with an argument of zero to mute the controller's buzzer; argument of one to re-enable it. (See also the SOUNDTOG FLASH parameter in the next chapter to set its default state.)

The SAY command number 15 lets you play your own sounds through the buzzer. The argument consists of a length-specified string of duration/frequency tone pair bytes. The duration is measured in 20 millisecond increments. Frequencies are half-tones, limited by the 8-bit timer. You'll have to experiment with tones. Here is the sequence that generates the AROS tone when the robot stalls (in octal):

```
\012\001\012\000\012\010\012\000\012\001
```

TCM2

The TCM2 accessory is an integrated inclinometer, magnetometer, thermometer, and compass that attaches to one of the `AUX` serial ports of the AROS 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 45 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.

Onboard PC

Communication between the onboard PC and the H8S microcontroller is RS232 serial through the respective `COM1` (Windows) or `/dev/ttyS0` (Linux) and internal HOST ports. Set the `HostBaud` FLASH communication rate to match the PC client-software's serial port rate.

Beginning with AROS version 1.6, the `RI` pin 9 on the HOST port initializes to low and goes high when the batteries discharge to below 11 VDC. We use the *genpowerd* software under Linux to detect that low-power signal and automatically shut down the PC. Windows PCs are a bit more problematic.

The Windows *genpowerd*-like *ups.exe* program requires a dedicated serial port and prefers to use the CTS line to indicate low power. Accordingly, we jumper the RI signal of HOST `COM1` to the CTS signal pin of the adjacent `COM2` port of the onboard PC for the feature. For convenience, the Versalogic VSBC8 PC found onboard most recent Pioneer 2s shares its 20-pin connector on the PC's motherboard with `COM1` and `COM2`. So, to implement Windows *ups.exe*-enabled low-power shutdown, we jumper pin 8 (`COM1` RI) to pin 16 (`COM2` CTS) on that VSBC8 serial connector. Use a similar strategy for other implementations; the UPS configuration dialog lets you select COM1-4.

Once the port is wired, start up Windows and, as Administrator, go to the `Start:Settings:Control Panel:Power Options` dialog and select the `UPS` tab. Click `Select` and in the UPS Selection dialog, select `COM2` (or other) port, `Generic` manufacturer, and `Custom` model. Then click Next.

In the UPS Interface Configuration On: COM2 dialog, check the `Power Fail/On Battery` and its related `Position` options. Uncheck to disable the Low Battery and UPS Shutdown options. Then click Finish to save the settings and close the dialog. Click OK or Apply to enable the UPS shutdown programs.

Change a registry value so that the PC shuts down one minute instead of two minutes after low-power notification by the controller: Use *regedit* and navigate to `[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\UPS\Config`. Change the `ShutdownOnBatteryWait` dword value to 1 (from 2).

Use the AROS client maintenance command #250 to test your *genpowerd* or `ups.exe` setup. Send the `COMshutdown` command #250 with an integer argument of 1 to simulate the low battery condition, in which AROS issues warnings first, then disconnects from the client after about a minute and sets the PC-shutdown signal on `RI`. An argument of 2 forces the computer shutdown signal (RI high); 0 cancels the shutdown/test. Resetting the controller cancels shutdown, too, unless battery power really is very low.

Put the controller into maintenance mode and fix your onboard PC settings if the computer falsely engages *genpowerd* or *ups.exe*.

Heading Correction Gyro

With the new rate-gyro accessory, your client software may detect and compensate for robot heading changes that aren't detected by the wheel encoders, such as from slipping wheels. AROS version 1.8 and later supports the gyro via its attachment to the AN6 and AN7 analog-to-digital input ports on the H8S microcontroller.

AROS collects 10-bit (0-1023) gyro rate and 8-bit (0-255) temperature data and will, upon request, send the collected data to a connected client in a new `GYROpac` (type=0x98) server information packet for processing. Analysis of the gyro data and subsequent

modifications to the robot's heading are done on the client side, as supported in the latest versions (1.3 and later) of ARIA.

To enable the gyro, you must set the `HasGyro` FLASH parameter to 1 using the AROScf tool (see next chapter). Set it to 0 if the gyro isn't attached. Then to acquire gyro data, send the `GYRO` client command #58 with integer argument of one; zero disables the gyro SIP. The gyro SIP is stopped upon client disconnection or controller reset, too.

AROS collects the gyro rate and temperature readings at the maximum rate of once every 25 milliseconds and reports each of these values to the client, when enabled, in the `GYROpac` SIP that gets sent just before the standard Server Information Packet every `sInfoCycle`, typically every 100ms. `GYROpac` consists of a count byte of the rate and temperature data pairs accumulated since the last cycle (typically 4 for a 100ms cycle time), followed by that number of rate/temperature integer/byte pairs.

Gyro rates are 10-bit integers of value 0-1023. When not moving, the rate is centered around 512 or so, depending on the gyro's temperature and other calibration factors which drift with use and should be corrected on the fly. Values below that center point indicate counter-clockwise rotational rates; values above the resting center measure clockwise rotational rates.

Table 12. GYROpac SIP contents

LABEL	BYTES	CURRENT VALUE	DESCRIPTION
HEADER	2	0xFA, 0xFB	Common header
BYTE COUNT	1	xx	Varies
TYPE	1	0x98	Packet type
N PAIRS	1	x	Number of gyro data pairs
FOR N PAIRS			
RATE	2	varies 0-1023	Gyro rate
TEMPERATURE	1	varies 0-255	Gyro temperature
CHECKSUM	2	varies	Computed checksum

INPUT OUTPUT (I/O)

Your AROS-based robot comes with a number of I/O ports that you may use for sensor and other custom accessories and attachments. See Appendix A for port locations and specifications. Some I/O states and readings appear in the standard SIP and may be manipulated with AROS client commands. There also is an `IOpac` SIP for convenient access to all of your robot's I/O.

User I/O

The User I/O connector on the H8S controller contains eight digital input and eight digital output ports, as well as an analog-to-digital (A/D) port.[23] The bit-mapped states of the sixteen digital ports and analog port automatically and continuously appear in the standard SIP, in their respective `DIGIN`, `DIGOUT`, `ANALOG` bytes. When not physically connected, the digital input and A/D port values may vary and change without warning.

Use the AROS client command number 30 to set one or more of the eight `DIGOUT` ports on the AROS controller. Electrically, the ports are digital high (1) at ~5 VDC (V_{cc}) and low (0) at ~0 VDC (`GND`). `DIGOUT` uses a two-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.

[23] Many of these ports are used by the Gripper accessory. Alternative I/O also is available.

For example, here's the AROS client command to set digital output ports one and three (OD1 and OD3), reset port four (OD4), and leave all the rest alone (hexadecimal notation):

```
0xFA, 0xFB, 0x06, 0x1E, 0x1B, 0x19, 0x09, 0x37, 0x24
```

Bumper and IR I/O

Two 10-position latching IDC connectors on the H8S controller provide 16 digital input ports, normally used for the bumper accessory, but also available for your own attachments. See *Appendix A* for connector details.

Similarly, the Motor-Power connector on the H8S controller contains eight digital inputs that we normally use for IR sensors on the Performance PeopleBot and PowerBot, and whose states are digitally mapped. See *Appendix B* for connector details.

Normally pulled high (digital 1), all the bumper and IR bit-mapped switches go low (digital 0) when the respective port gets triggered. Bumper inputs also appear with the stall bits in the standard SIP, but unlike in the IOpac, are modified by the InvertBumps mask. All the bumper and IR data bits appear in the IOpac packet.

IO packets

Table 13. IOpac packet contents

LABEL	BYTES	CURRENT VALUE	DESCRIPTION
HEADER	2	0xFA, 0xFB	Common header
BYTE COUNT	1	22	Number of data bytes + 2
TYPE	1	0xF0	Packet type
N DIGIN	1	4	Number of digital input bytes
DIGIN	1	varies 0-255	ID0-8 bits mapped
FRONTBUMP*	1	varies 0-255	Front bumper bits mapped
REARBUMP*	1	varies 0-255	Rear bumper bits mapped
IR	1	varies 0-255	IR inputs
N DIGOUT	1	1	Number of digital output bytes
DIGOUT	1	varies 0-255	Digital output byte(s)
AN	1	5	Number of A/D values
A/D	10	5 integers varying 0-2047	A/D ports 1-5 input values at 12-bit resolution = 0-5 VDC
CHECKSUM	2	varies	Computed checksum

Not all analog and digital I/O appears in the standard SIP. Accordingly, your client software may request the IOpac SIP (type = 240; 0xF0), which contains all common I/O associated with the H8S controller and which appear on the various connectors, including User IO, General IO, Bumpers, and IRs.

Use the AROS client IOREQUEST command number 40 with an argument value of zero, one, or two. The argument value one requests a single packet to be sent by the next client-server communications cycle. The request argument value of two tells AROS to send IOpac packets continuously, at approximately one per cycle depending on serial port speed and other pending SIPs. Use the IOREQUEST argument value zero to stop continuous IOpac packets.

* Actual bits, not affected by InvertBumps since bumper bits may be used for other digital input besides bumpers.

Expansion I/O

Four alternative A/D ports appear at the 40-position Expansion I/O connector of the H8S microcontroller.[24] Use the `ADSEL` client command number 35 to select and subsequently have the A/D value from one of the alternative ports `AN2-5` appear in the standard SIP. The default port is `AN0` (ADSEL argument value of zero), the A/D port also on the User I/O connector.

DOCKING/CHARGING SYSTEM I/O

The docking/charging system's mechanism and associated charge-management circuitry on the robot may be controlled from the robot's H8-microcontroller and AROS servers.

Digital Port Controls

When set digital high (1), the "inhibit" port OD4 on pin 10 of the User I/O connector (see Appendix A) causes the charging mechanism to disengage and retract from the charging platform and inhibits its future deployment. The "deploy" port OD5 pin 12, when set high with port OD4 low, deploys the charging mechanism with full force to seat it onto the charging platform.[25]

At the fully deployed position, the mechanism is mechanically stabilized and requires much less force to maintain contact. If in positive contact with the charger base, the robot's onboard circuitry activates and thereafter maintains the actuated mechanism at that lower force as long as it receives power. To minimize heat and eventual damage to the actuator, the deploy line should be activated for only short periods; maximally for 10 seconds at a time.

Your client software may run the charging mechanism by individually activating/ deactivating the digital output ports, such as with the AROS COMdigout (#30) command. However, for best results, we recommend using the automated charging control commands and systems we provide with the latest AROS.

Docking/Charging Servers

To use AROS' docking/charging system servers (version 1.7 or later), you must first enable the H8-microcontroller's automated charger servers through your robot's FLASH parameters. Use the AROScf configuration tool and set the Charger parameter value to 1 (0 to disable) and save the value.

Thereafter, for autonomous operation of the robot with the charging platform, establish a client-server connection between an ARIA- or similar client-enabled PC and the robot's controller. Use the AROS `CHARGE` command #68 with an integer argument of 1 to automatically halt robot motion and deploy the docking mechanism. The docking mechanism automatically retracts after five seconds if the robot does not engage with the docking platform, during which time the robot's drive system is unresponsive. So your client should wait at least that long before attempting to resume activity.

Although disengaged while recharging, AROS remembers if your robot's motors were engaged just before deploying the docking mechanism. This way, your robot may discontinue charging, retract the robot's charging mechanism, and go on its merry way automatically by having the client send any motion command that normally would cause the robot to drive away from its current position. However, if you purposely

[24] Many other ports also appear at that connector, but are not yet supported in AROS.
[25] These output ports and the charge-sensing User I/O-based digital input ports (see below) do not interfere with the Pioneer/PeopleBot Gripper.

disengage the motors while charging, such as by disconnecting, you will have to re-engage them from the client or by manually pressing the MOTORS button on the controller. Re-engaging the motors automatically retracts the charging mechanism.

While the motors are engaged, the charging mechanism cannot be deployed, except by the `CHARGE` command. For best control and safety, consider also using the AROS `CHARGE` command number 68 with integer argument 0 to gracefully cancel charging, retract the charging mechanism, and restore motor state.

In addition to the client-mediated commands, you also may cancel recharging and retract the charging mechanism manually with the Charge Deploy button, as described in the earlier sections. Do note that client-mediated docking/charging behaviors may act to reverse your actions.

For example, the client may, upon untimely loss of recharging power resulting from someone pressing the Charge Deploy button, may re-engage the motors and have the robot automatically attempt to re-dock with the charging platform and restart charging.

Your client software may disengage and re-engage the client-server connection without disrupting recharging, as long as the robot remains positively engaged with the charging platform and you don't do anything else to otherwise disrupt recharging. Once disengaged from the client, the rules for engaging and disengaging the recharge mechanism and power manually apply.

Monitoring the Recharge Cycle

Three digital signals indicate battery recharging states of the docking/recharging system. All appear in the standard SIP.

Table 14. Recharging cycle states

Charge State	Overcharge (ID7)	~Volts	Charge current
Bulk	1	discharge-~14V	6A
Overcharge	0	~14-14.7	6A
Float	1	~13.5	< 1A

The "power-good" signal appears as both User I/O `DIGIN` bit 6 and as bit 10 of the `FLAGS` integer in the standard SIP, but their states are inversely related: `DIGIN` bit 6, normally high (1) when not charging or when the charging system is not installed, goes low (0) when the recharge system is engaged on the charge platform. Conversely, the power-good bit 10 in `FLAGS` normally is low and goes high when the robot is docked and charging. For compatibility with future docking systems, we recommend that your client monitor the power-good FLAGS bit and not the DIGIN line to determine if the robot is getting power from the charging platform.

The `DIGIN` and `DIGOUT` bytes of the Standard SIP also reflect the states of the associated charging digital input and output bits. `DIGOUT` bits 4 and 5 are the inhibit and deploy output ports described earlier. `DIGIN` bit 7, corresponding to the User I/O connector digital input port ID7, pin 15, reflects the battery recharge cycle and, with the `Battery Voltage` SIP value, helps the autonomous robot client determine immediate battery life and operation times.

The "overcharge" bit ID7 is set (1) when the batteries are well below full charge and the charger is at full charging current. During this bulk-charging period, the battery voltage rises to around 13.8-14V. The overcharge bit ID7 then drops to low (0) while the batteries charge from approximately 80% to 90% of full charge: from ~13.8 to 14.7V. The charger then reverts to "float mode", maintaining full charge at much lower current and charger voltage (~13.5V).

In float mode, the overcharge bit ID7 is set.

Accordingly, by monitoring the power-good and overcharge bits, as well as the battery voltage, your client may make recharging strategy decisions. The thing to remember is that lead-acid batteries last longest when routinely charged into float mode, typically once per day.

Chapter 7 Updating & Reconfiguring AROS

The AROS software and a set of operating parameters for your *Activ*Media robot get stored in the H8S microcontroller's FLASH ROM. With special upload and configuration software tools, you change and update the FLASH memory image. No hardware modification is required.

WHERE TO GET AROS SOFTWARE

Your *Activ*Media robot comes preinstalled with the latest version of AROS. And the various AROS configuration and update tools come with the robot on CD-ROM. Thereafter, stay tuned to the `pioneer-users` newsgroup or periodically visit our support website to obtain the latest AROS software and related documentation:

`http://robots.activmedia.com`

AROS tools come in two flavors: One (`dl_AROSV_v`) simply updates the AROS servers in FLASH. The other utility, `AROScf`, is a multi-functional application for both uploading new AROS versions as well as modifying your robot's onboard FLASH-based parameters.

AROS MAINTENANCE MODE

To connect with and update your robot's AROS servers and its FLASH-based operating parameters, you need to first connect a serial port on the PC from which you will run the AROS tool(s) to the HOST port of your robot's microcontroller:

- ✓ If you are running from an onboard PC, the computer-to-`HOST` connection already is made.
- ✓ If you have an onboard PC, but prefer to use an external computer for maintenance, simply power down the onboard computer.
- ✓ If you use radio or Ethernet wireless, switch `RADIO` power OFF.
- ✓ When connecting from an external PC, directly tether (no radios) its serial port to the 9-pin DSUB serial connector on the User Control Panel.

Now start up your robot and put its controller into the special Maintenance Mode:

1. Press and hold the white `MOTORS` button on the User Control Panel
2. Press and release the adjacent red `RESET` button
3. Release the `MOTORS` button.

The `STATUS` LED on the User Control Panel should flash twice the rate than when in server ("wait") mode and the `BATTERY` LED should shine bright red.

SIMPLE AROS UPDATES

The simple AROS update application is just that: a standalone program that, when run, updates the AROS servers to the indicated version `V_v` (1_0, for example) in your robot's microcontroller. Although it may *add* parameters to your current FLASH values, the `dl_AROSV_v` application *never* changes your current parameters.

To use this convenient utility, simply download the "dl_"-prefixed executable for your PC's operating system ("`.lin`" filename suffix for Linux or "`.exe`" for Windows). Connect the PC to your robot's `HOST` serial port and put its controller into Maintenance Mode (see section above). Then run the `dl_...` program.

Text prompts will help you get connected with your *Activ*Media robot's H8S-based controller and update its AROS servers. No fuss. No muss.

AROSCF

The AROS update and configuration program, `AROScf`, is part of a collection of utilities and files for comprehensive management of your *Activ*Media robot's onboard servers and FLASH-based operating parameters. The distribution archive for the software is simply named `AROSV_v` (`V` and `v` are the version major and minor numbers, such as 1_0), with a "`.tgz`" suffix for Linux-based PCs or "`.exe`" for Windows computers.

Install the utilities and files on the PC you plan to use for maintaining your robot's operating system and parameters by double-clicking the distribution software's onscreen icon or otherwise executing the self-extracting, self-installing package. For Linux, `uncompress` and `untar` the files:

```
% tar -zxvf AROS1_0.tgz
```

The expanded archive creates an `AROS/` directory in the selected Windows or current Linux path and stores the AROS software within.

STARTING AROSCF

`AROScf` is a text-based console application, as opposed to a graphical-user one. It runs in two stages: Startup Mode followed by Interactive Mode. When invoked, you may start `AROScf` with various command-line options. With an X-terminal under Linux, for example, navigate to the AROS directory and invoke the program:

```
% cd /usr/local/AROS
% ./AROScf <options>
```

With Windows PCs, you may double-click the `AROScf` icon to automatically open a console window and start the program without any options. To start up with command-line options, `Run` the program from the `Start` menu, or run `Command` from the `Start` menu, then navigate to the AROS directory and start `AROScf` with options.

For example (after invoking the MSDOS-like command window):

```
C:\> cd AROS
C:AROS\> AROScf <options>
```

Normally (without any command-line arguments), AROScf starts up expecting to connect your PC's `COM1` or `/dev/ttyS0` serial port with your robot's microcontroller which you've put into Maintenance Mode. If successfully connected, the program automatically retrieves your robot's FLASH-stored operating parameters and enters interactive mode.

If the initial connection fails, AROScf still starts up into its Interactive Mode, but with empty, and thereby useless parameter values. You may still operate many of AROScf's interactive features without a connection, such as maintain disk-based copies of your robot's operating parameters. And there is an interactive `connect` command that lets you establish a maintenance connection with your robot. See the next section for AROScf commands and operating features.

Include each of the selected AROScf's startup-mode options as a key letter with a dash ("-") prefix, followed by any required arguments, separated by spaces. For example, to start up AROScf and make a connection with a serial port other than the default `COM1` or `ttyS0`:

```
C:\AROS> AROScf -p COM3
```

Similarly, this Linux `xterm` command uploads a fresh copy of AROS to your robot's H8S-based microcontroller and then exits, much like the simple `dl_AROS1_0` program:

```
% ./AROScf -d AROS1_0.hex -n -b
```

Table 15. AROScf startup options

KEY	ARGUMENT	DESCRIPTION
-b	command arguments	Batch mode executes list of AROScf Interactive mode commands with arguments
-d	hexfile	Automatically upload AROS hex image file after connecting with the controller
-h	none	Print help message and exit
-l	paramsfile	Load the disk-stored parameter file instead of the robot's copy
-n	none	Don't automatically connect with controller
-p	serial-device	Uses specified serial port for connection
-s	paramsfile	On exit from AROScf, automatically save the current parameter values to the named paramsfile

CONFIGURING AROS OPERATING PARAMETERS

Your *Activ*Media robot has several parameters stored in FLASH that AROS uses to configure its servers and auxiliary attachments and to uniquely identify your robot. For instance, the default maximum translational velocity is stored in the `TransVelMax` parameter. Its value takes effect when starting your robot or after resetting the microcontroller, and may be changed temporarily by a client command. Use AROScf's interactive mode to modify these operating parameters, and hence your robot's default operating characteristics.

Start up AROScf as described in the previous section. As discussed earlier, AROScf normally downloads the set of operating parameters from your robot's FLASH for your review and modification. Or you may load a disk-stored version of those parameters. Some of the parameters, "Constants", should not to be changed. The others, "Variables", are the identifying and operating parameters that you may edit.

Interactive Commands

To operate AROScf in interactive mode, simply type a keyword at the command line. Some keywords affect the operation of AROScf, the status of the parameters file as a whole, or the connection between AROScf and your robot's microcontroller. For instance, to review the list of current AROS constants or variables, type **`'c'`** or **`'v'`**, respectively, followed by a return (`Enter`). Similarly, type `'?'` or `'help'` to see a list of AROScf interactive commands.

Changing Parameters

Other keywords refer to the operating parameters themselves. Alone, a parameter's keyword simply asks AROScf to display the parameter's value. Provide an argument with the parameter keyword separated by a space to change its 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
```

See the respective control command and parameter Tables nearby for a full description of AROScf operation.

Table 16. AROScf control commands

COMMAND	DESCRIPTION
KEYWORD <value>	Alone, a keyword displays current, edited value. Add argument to change current value.
c or **constants**	Display all constant parameters. You cannot edit these.
v or **variables**	Display all variable parameter values which you may edit and eventually save to your robot's FLASH.
r or **restore** <paramsfile>	Restores variables to values currently stored in FLASH or from a paramsfile on disk
save <paramsfile>	Saves current edited values to FLASH or saves current edited values to pathname on disk for later reference.
q or **quit**	Exits AROScf.
connect <portname>	Connects AROScf with microcontroller through serial port (COM1 or /dev/ttyS0 default)
disconnect	Disconnects AROScf from your robot's microcontroller
? or **help**	Displays these commands and descriptions.

SAVE YOUR WORK

While changing parameter values in AROScf Interactive Mode, you are editing a temporary copy; your changes are not put into effect in your robot's FLASH until you explicitly "save" them to the microcontroller.

Also use the AROScf save command to save a copy of the parameters to a disk file for later upload. We strongly recommend that you save each version of your robot's parameter values to disk for later retrieval should your microcontroller get damaged or its FLASH inadvertently erased. Default parameter files come with each AROS distribution, but it is tedious to reconstruct an individual robot's unique configuration.

PID PARAMETERS

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

Table 17. AROS FLASH configuration parameters with values for Pioneer 3–DX

KEYWORD	Type	Default	Description
CONSTANTS			*Should not be changed*
PTYPE	str	Pioneer	Identifies the robot type.
PSTYPE	str	P3DX	Identifies the *Activ*Media robot model.
SERNUM	str	factory	Serial number for the robot.
VERNO	str	1.x	AROS version number
TOPRV	int	360	Maximum rotational velocity; deg/sec
TOPTV	int	2200	Maximum translation speed; mm/sec
TOPRA	int	600	Maximum rotation (de)acceleration; deg/sec^2
TOPTA	int	4000	Maximum translational (de)acceleration; mm/sec^2
TICKSMM	int	132	Encoder ticks/mm: $\frac{\text{(ticks per rev x gear-ratio)}}{\text{(wheel\_diameter x PI)}}$
BATTCONV	byte	0	0 if a 12V system; 1 if 24V
VARIABLES			*Parameters that you may change*
NAME	str	not_set	Unique name for your robot. Maximum of 20 characters, no spaces.
SIP	byte	100	Server information packet cycle time in 1 ms increments. Default is classic 100 ms.
PWMMAX	int	400	Maximum motor PWM (500=fully on).
HOSTBAUD	byte	0	Baud rate for client-server HOST serial: 0=9.6k, 1=19.2k, 2=38.4k, 3=56.8k, 4=115.2k.
AUXBAUD1	byte	0	Baud rate for AUX serial port 1; see HostBaud
AUXBAUD2	byte	0	Baud rate for AUX serial port 2; see HostBaud
SONARCYCLE	byte	40	Sonar cycle time in milliseconds
SONAR1	str	12345678	Ping sequence for sonar array #1. Up to 16 number characters 1-8; 0 to disable the array
SONAR2	str	0	Ping sequence for array #2. See sonar1 above
SONAR3	str	0	Ping sequence for array #3. See sonar1 above
SONAR4	str	0	Ping sequence for array #4. See sonar1 above
LOWBATTERY	int	110	In 1/10 volts; microcontroller alarm activated when battery charge falls below this value.
WATCHDOG	int	2000	Ms time before robot automatically stops if it has not received a command from a client. Restarts on restoration of connection.
REVCOUNT	int	36300	The number of differential encoder ticks for a 360 degree revolution of the robot.
SOUNDTOG	byte	1	0 disables the buzzer
P2MPACS	byte	0	1 enables alternative SIP.
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.
HASGYRO	byte	0	Set to 1 if you have the gyro accessory
CHARGER	byte	0	Set to 1 if P3 or 2 if PowerBot autocharger mechanism and circuitry installed; otherwise 0
GRIPPER	byte	0	Set to 1 if P2/P3 Gripper; 2 if Gripper on Performance PeopleBot
TCM2	byte	0	TCM2 module connected to 1=AUX1 or 2=AUX2
LEDGESENSE	byte	0	0=none; 1=stop on detect; 2=limit speed; 3=stop and limit speed
BUMPSTALL	byte	3	0=disable bump stall; 1=enable rear; 2=enable front; 3=enable both front and rear bump stalls
INVERTBUMP	byte	0	0=none; 1=front; 2=rear; or 3=invert both front and rear bumper signals
FRONTBUMPS	byte	0	Number of front bumper segments
REARBUMPS	byte	0	Number of rear bumper segments
ROTVELMAX	int	200	Max rotational speed; deg/sec.
TRANSVELMAX	int	2000	Max translational speed; mm/sec.
ROTACC	int	100	Rotational acceleration; deg/sec^2
ROTDECEL	int	100	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	450	Differential PID for translation
TRANSKI	int	4	Integral PID for translation
JOYVELMAX	int	1000	Joydrive maximum translation velocity
JOYRVELMAX	int	50	Joydrive maximum rotational velocity

The Proportional PID (Kp) values control the responsiveness of your robot. Lower values make for a slower system; higher values make the robot "zippier", but can lead to overshoot and oscillation.

The Derivative PID (Kv) dampens oscillation and overshoot. Increasing values gives better control of oscillation and overshoot, but they also make the robot's movements more sluggish.

The Integral PID (Ki) adjusts residual error in turning and velocity. Higher values make the robot correct increasingly smaller errors between its desired and actual angular position and speed.

TICKSMM AND REVCOUNT

AROS uses the `ticksmm` and `revcount` parameters to convert your platform-independent speed and rotation commands—typically expressed in millimeters or degrees, respectively—into platform-dependent units.

The `ticksmm` value is the number of encoder pulses ("ticks") per millimeter of wheel rotation. The value is, of course, dependent upon the wheel encoder's resolution, the motor-to-wheel gear ratio, and the wheel's diameter. These don't normally change, and so are considered constants and not editable for your robot.

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.

Table 18. Some platform-dependent robot parameter values

PARAMETER	Model						
	DX	**DXE**	**CE**	**PB V1**	**P3DX, PerfPB, DX8, DX8 and PerfPB *Plus***	**AT, AT8**	**P3AT & AT8 *Plus***
ENCODER TICKS/REV	500	500	500	500	500	100	100
GEAR RATIO	19.7	19.7	19.7	38.3	38.3	85.5	57.5
WHEEL DIAM (MM)	165	191	165	165	191	220	220
ENCODER TICKS/MM	76	66	76	148	132	49	138
DISTCONVFACTOR	0.840	0.969	0.826	0.413	0.424	1.32	0.465
DIFFCONVFACTOR	0.0056	0.0057	0.0056	0.0056	0.0060	0.0034	0.0060

Ticksmm and revcount affect only the conversion of your motion command arguments into platform-dependent values. Your client must independently convert values reported back from the server, such as X-Pos and Th, into platform-independent values. ARIA clients use the conversion factors found in your robot's respective ARIA\params file (p3dx.p, for example).

To adjust both the server and client parameter values for your robot, first connect the robot with a client and have the robot move a certain distance, preferably one to three or more meters. Measure the actual distance moved, not the client-reported value and adjust ticksmm accordingly.

Similarly, rotate your robot from the client and measure the actual achieved heading. Adjust revcount (the measure of differential encoder ticks to achieve 360-degrees rotation) accordingly.

When you are satisfied that the robot moves and rotates the proper distances and headings, adjust the related client-side parameters in your robot's params .p file, so that your client responds accurately.

STALLVAL AND STALLCOUNT

An AROS stall monitor maintains a running average of PWM values for each wheel over a 500 millisecond integration period. PWM values get added to the sum if the wheel speed is below 100 mm/sec. The average is then compared with the stallval FLASH value. If it exceeds that value, in other words the motors are being given lots of power but are barely moving if at all, a stall occurs. Once stalled, power is removed and the motors relax for the stallwait period, after which power gets reapplied.

BUMPERS

Introduced in AROS version 1.6, use the BumpStall FLASH parameter to set the default for the robots behavior when its front and/or rear bumper gets triggered. Normally, BumpStall is engaged for both front and rear (default value of 0) bumpers. Reset it to 3 to disengage bump stalls altogether; 1 to trigger stalls only when the rear bumpers engage; or 2 for front bumps only.

You may over-ride the BumpStall FLASH default with the bump_stall client command number 44, although the command arguments are the reverse: enabling versus disabling the various bumper-stall combinations.

Your robot's BumpStall behavior reverts to the FLASH default on reset and up disconnection from the client.

Next-generation client-side software will need to know if you have bumpers or not and how they are configured. And new bumper hardware inverts the Pioneer 2's bumper signal bits which confuses the client-server software. Moreover, different AROS-enabled robots have different numbers of bumper segments, front and rear. Accordingly, the new AROS v1.6 implements three new FLASH parameters that specify states (invert or not) and numbers of front and rear bumper segments. Unfortunately, we have no way of knowing automatically what bumpers your robot may have, if any, so we are forced to assume you DON'T have bumpers or that you have the old-style (non-inverting) bumpers.

Use AROScf to indicate the type and number of bumper segments. Set the new InvertBump FLASH parameter's value to 1 if you have new bumpers in front, which signals need to be inverted; 2 if in the rear; or 3 if both front and rear bumper signals need inverting. Set to the default 0 if your robot has no bumpers or has the original style (non-inverting) bumpers.

Set the FrontBump and RearBump parameters to the number of bumper segments for the front and rear bumpers, repectively; or to 0 if you don't have a particular bumper. For pre-AROS 1.6 robots, you don't need to set these values to have them work with AROS 1.6. The number of segments is used to isolate the bumper bits, if any, and to apply InvertBump as needed, so that a triggered bumper is reported as digital 1 regardless of the hardware, and is reported as such in the standard SIP. IOpac, on the other hand, simply reports the bit-mapped states of the input bytes associated with the bumpers, regardless of hardware.

The FrontBump and RearBump byte values also are reported near the end of the CONFIGpac.

If for any reason you remove a new-style bumper from your robot, you MUST reset the InvertBump FLASH value or disable BumpStall for that bumper. Otherwise, the robot will stall incessantly.

Chapter 8 Maintenance & Repair

Your *Activ*Media robot is built to last a lifetime and requires little maintenance.

TIRE INFLATION

Maintain even tire inflation for proper navigation of your Pioneer 3 or 2 robot. We ship with each pneumatic tire inflated to 23 psi. If you change the inflation, remember to adjust the `ticksmm` and `revcount` FLASH values.

DRIVE LUBRICATION

Pioneer 3 and 2 drive motors and gearboxes 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 for those equipped with the automated docking/charging system, your Pioneer robot has 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 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 some of your *Activ*Media robots 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 Batteries

If you have the standard charger accessory, insert it into a standard 110 or 220 (Europe/South America/Asia) VAC 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 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.

Automated Docking/Charging System

The automated docking/charging system accessory optimally conditions power to charge the three 21-Ahr, 12 VDC lead-acid batteries (6 A charging current max) and provides sufficient power (up to 5A) for operation of all onboard systems.

The charging mechanism and onboard power conditioning circuitry can be retrofitted to all Pioneer 3 and some Pioneer 2 and PeopleBot robots. All require return to the factory.

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 charger, 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.

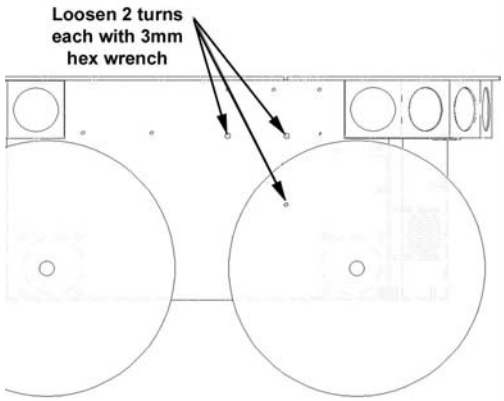

Figure 19. Loosen the AT drive belt retainer screws first.

TIGHTENING THE AT DRIVE BELT

Occasionally, particularly after heavy use, the Pioneer 3- or 2-AT drive belts that mechanically link the front and rear motors on each side will loosen and slip, resulting in a load popping noise. To start, use a 3mm hex key to loosen, but not remove, the three screws on the side of the robot near the front wheel. One screw is partly behind the wheel, so with our parts kit, we included a 3mm hex key with a shortened "L" section to fit behind the wheel.

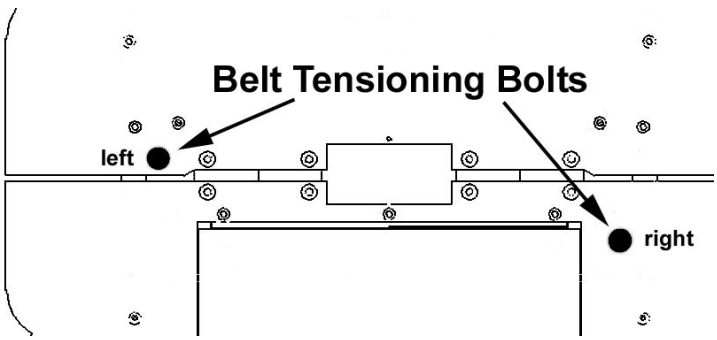

Figure 20. Locations of Pioneer 2- and 3-AT's belt-tensioning bolts

Remove the small plastic plug which is near the hinge on the top plate and near the edge by the wheel. Under it, you will see the head of a large hex bolt. This bolt tightens (clockwise) or loosens (counter-clockwise) the drive belt for that side of the robot. Turn it using a 5mm hex key probably not more than 1 full rotation. Avoid over tightening.

Test to make sure that it is tight enough by holding the wheel while running the self test. When adjusted satisfactorily, re-tighten the screws on the side and replace the plug.

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.

REMOVE THE BATTERIES FIRST!

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 3 and 2 DX or AT robot.

Removing the Nose

The Pioneer 3- and 2-DX and –AT 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. Two screws hold the nose to the front sonar (or blank) array. The AT also has a screw at the bottom of the nose that attaches to the body; the DX's nose is hinged at the bottom.

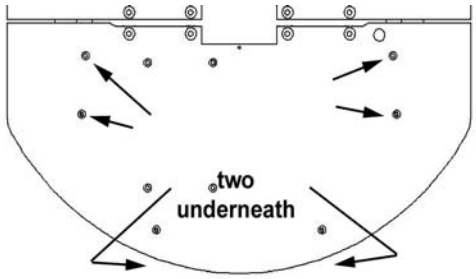

Figure 21. Remove indicated screws to access front plate of Pioneer 2- and 3-DX and -AT robots.

Remove all nose retaining screws with the 3mm hex wrench supplied with your robot. Unlike earlier Pioneer 2 models, you do not have to remove the Gripper or the front Bump Ring accessories.

Once loosened, the DX nose pivots down on a hinge. For the AT model, four pins along the nose's back edges guide it onto the front of the robot. Simply pry the nose out and away from the body.

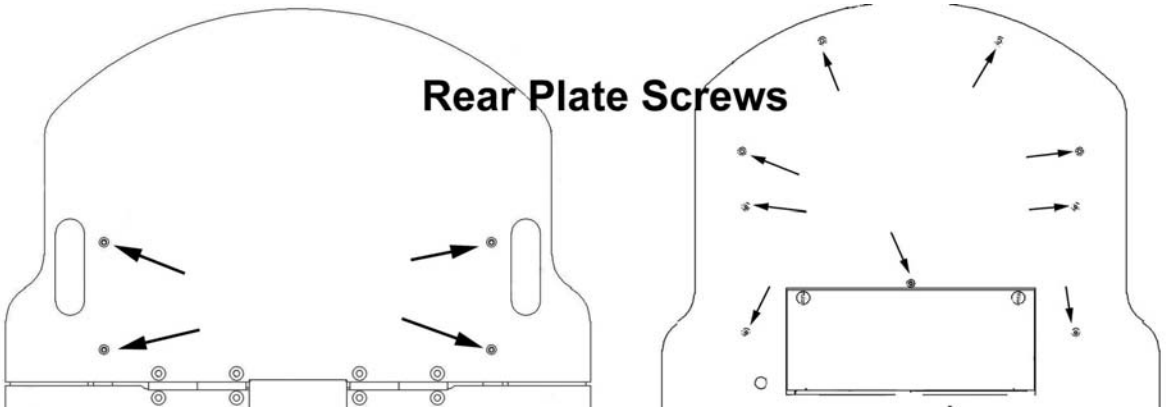

Figure 22. Remove indicated screws from Pioneer 2- or 3-DX or -AT rear deck to open plate.

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

All the H8S-based Pioneer robots have a center hinge in the deck which let you easily open and access internal components without completely removing the top plate. Simply remove the indicated 3mm screws shown in the Figures nearby from the section of the deck that you want to access. You may need to first remove any accessories that are bolted to the top plate through the indicated holes.

Remove the batteries BEFORE opening the robot.

FACTORY REPAIRS

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

support@activmedia.com
(603) 881-3818 (fax)

Tell us your robot's SERIAL NUMBER

In the body of your email or fax message, describe the problem in as much detail as possible. Also include your **robot's serial number** (IMPORTANT!) as well as name, email and mail addresses, along with 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

H8S PORTS & CONNECTIONS

This Appendix contains pinout and electrical specifications for the external and internal ports and connectors on the H8S microcontroller, motor-power interface, and User Control boards.

Note that layered connectors are numbered differently, depending on the socket type. IDC ones are odd and even layers; mini- and micro-fit connectors use successive-position numbering. See the Figures nearby for examples.

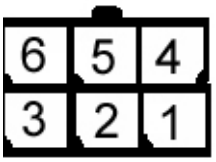

Figure 23. Mini- and micro-fit style connector numbering

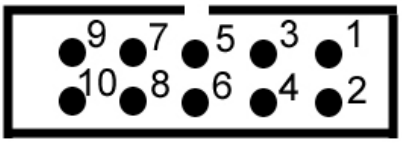

Figure 24. IDC-type connector

H8S MICROCONTROLLER

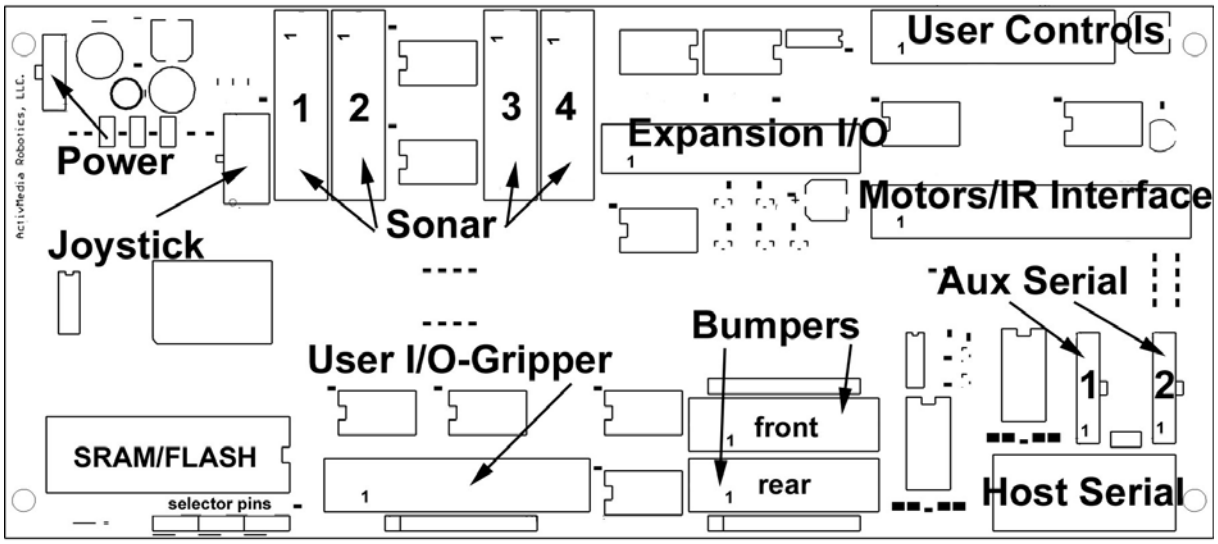

Figure 25. ActivMedia's H8S-based microcontroller

Power Connector

The power connector is a 3-pin microfit socket that delivers 12VDC (battery) to the microcontroller circuitry and separate, conditioned 5 VDC to the sonar, including power grounds.

Table 19. H8S Controller Power Connector

PIN	DESCRIPTION
1	12 VDC battery
2	GND
3	Sonar 5VDC

Serial Ports

Two DSUB-9 and two 5-pin microfit sockets provide the HOST and AUX1/AUX2 auxiliary serial ports for the H8S controller. All are RS-232 compatible. The HOST port is shared on both the User Control Panel as well as on the H8S controller board and is for AROS client-server and maintenance connections.[26] The internal HOST serial connector also has signal lines for detecting an attached device (DTR pin 4) and for notifying the attached PC of low-power condition (HRNG pin 9). The HOST serial connectors are wired DCE for direct connection (straight-through cable, not NULL-modem) to a standard PC serial port or to a radio modem set to DTE mode. See the nearby Tables for details.

The AUX1 and AUX2 serial ports are for RS232-compatible serial device connections, such as for the TCM2 Modules or any of several pan-tilt-zoom robotic systems.

AROS operates the serial ports at any of the common data rates: 9,600, 19,200, 38,400, 57,800, or 115,200 bits per second; and at eight data bits, one stop bit, no parity or hardware handshaking.

Table 20. HOST serial ports on H8S board and on User Control (*) (DSUB-9 socket)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	nc			**2**	*TXD	output
3	*RCV	Input		**4**	DTR	Input detects attached device and switches TxD and RxD into the uC
5	*GND	Common		**6**	*DSR	Output when controller powered
7	nc	May be jumpered to pin 8		**8**	nc	Jumper to pin 7 for radio modem handshaking
9	†RI	Output lowered to signal PC shutdown				

† Shared on Motors interface

Table 21. AUX1 and AUX2 serial ports (5-pos microfit sockets)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	DTR	Input		**2**	TXD	output
3	RCV	Input		**4**	DSR	output
5	GND	common				

User I/O, Gripper, Docking/Charging Port

A 20-pin latching IDC socket on the H8S microcontroller provides the digital, analog, and power ports for user connections and for the Gripper and automated docking/charging accessories, if installed. Indicated ports (*) are shared on other connectors. Digital inputs are buffered and pulled high (digital 1); outputs are buffered and normally low (digital 0).

Table 22. User I/O – Gripper (20-pos latching IDC)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	OD0	DIGOUT bit 0; Gripper enable		**2**	ID0	DIGIN bit 0; Paddles open limit
3	OD1	DIGOUT bit 1; Gripper direction		**4**	ID1	DIGIN bit 1; Lift limit
5	OD2	DIGOUT bit 2; Lift enable		**6**	ID2	DIGIN bit 2; Outer breakbeam IR
7	OD3	DIGOUT bit 3;		**8**	ID3	DIGIN bit 3;

[26] Unlike with earlier P2 controllers, HOST does not interfere with the User Control Panel serial connections if its attached device—PC or radio modem—is OFF.

		Lift direction				Inner breakbeam IR
9	ID4	DIGIN bit 4; Left paddle contact		**10**	OD4	DIGOUT bit 4; Automated docking/charging "inhibit"
11	ID5	DIGIN bit 5; Right paddle contact		**12**	OD5	DIGOUT bit 5; Automated docking/charging "deploy"
13	ID6	DIGIN bit 6; Automated docking/charging "power good"		**14**	OD6	DIGOUT bit 6; User only
15	ID7	DIGIN bit 7; Automated docking/charging "overcharge"		**16**	OD7	DIGOUT bit 7; User only
17	*AN0	A/D port 0 (default) (0-5VDC = 0-255)		**18**	Vcc	5VDC < 1A
19	Vpp	Battery 12VDC < 1A		**20**	Gnd	Signal/power common

The Expansion I/O Bus

- ✓ A 40-pin high-density IDC socket on the H8S microcontroller provides a general-purpose connector for future I/O expansion. Digital lines, including 8-bit bus address, data, read/write, and other general-purpose ones, are buffered with inputs pulled high. Indicated ports (*) appear on other connectors.

Table 23. General-purpose I/O and data bus (40-pos high-density IDC)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	AD0	Address bit 0		**2**	D7	Data bit 7
3	AD1	Address bit 1		**4**	D6	Data bit 6
5	AD2	Address bit 2		**6**	D5	Data bit 5
7	AD3	Address bit 3		**8**	D4	Data bit 4
9	AD4	Address bit 4		**10**	D3	Data bit 3
11	AD5	Address bit 5		**12**	D2	Data bit 2
13	ID6	Address bit 6		**14**	D1	Data bit 1
15	ID7	Address bit 7		**16**	D0	Data bit 0
17	AN6	A/D port 6; gyro temp		**18**	CS4	Chip select 4
19	AN5	A/D port 5; gyro rate		**20**	CS3	Chip select 3
21	*AN4	A/D port 4; Joystick Y		**22**	CS2	Chip select 2
23	*AN3	A/D port 3; Joystick X		**24**	WR	Data write
25	AN2	A/D port 2		**26**	RD	Data read
27	AN1	A/D port 1		**28**	CS6	Chip select 6 or digital I/O
29	*AN0	A/D port 0		**30**	CS7	Chip select 6 or digital I/O
31	GND	Signal common		**32**	RST	Controller reset
33	GND	Signal common		**34**	P1.1	Digital I/O
35	GND	Signal common		**36**	Vcc	Controller 5VDC (<200ma)
37	GND	Signal common		**38**	Vcc	Controller 5VDC (<200ma)
39	GND	Signal common		**40**	Vpp	Battery 12VDC (<0.5A)

Table 24. Bumper ports (10-pos latching IDC)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	BP0	Bumper bit 0		2	BP1	Bumper bit 1
3	BP2	Bumper bit 2		4	BP3	Bumper bit 3
5	BP4	Bumper bit 4		6	BP5	Bumper bit 5
7	BP6	Bumper bit 6		8	BP7	Bumper bit 7
9	Gnd	Common		10	Gnd	Common

Bumper Ports

Two 10-position latching IDC connectors provide general-purpose digital inputs, typically used for the robot's bumpers. All inputs are buffered and pulled high (digital 1).

Motors, Encoders, and IR Sensors

A 26-position latching IDC connector on the H8S microcontroller provides interface through an intermediate board to the Motor-Power Board (Appendix B). Line descriptions also can be found in the following Motor-Power Interface section.

Table 25. Motors, encoders, and IRs interface (26-pos latching IDC)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	LPWM	Left motors PWM		2	LDIR	Left motors direction
3	RPWM	Right motors PWM		4	RDIR	Right motors direction
5	MEN	Motors enable		6	LEA	Left encoder channel A
7	E-STOP	E-Stop detect input		8	REA	Right encoder channel A
9	RPWR	Radio power enable		10	REB	Right encoder channel B
11	APWR	Aux power enable		12	LEB	Left encoder channel B
13	CHRG	Charge port detect		14	IR6	IR input bit 6
15	IR7	IR input bit 7		16	IR4	IR input bit 4
17	IR5	IR input bit 5		18	IR2	IR input bit 2
19	IR3	IR input bit 3		20	IR0	IR input bit 0
21	IR1	IR input bit 1		22	VBAT	Battery voltage detect
23	Gnd	Signal common		24	AN1*	Analog input
25	Gnd	Signal common		26	AN2*	Analog input

* Board versions C and earlier pin 24 HOST RI and pin 26 ground.

User Control Interface

A 16-position latching IDC connector provides interface with the User Control Panel board and functions. See description in a following section.

Table 26. User Control Panel interface

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	Vcc	5 VDC power		2	Vcc	5 VDC power
3	RST	RESET button		4	MOT	MOTORS button
5	RPWR	Radio power switch		6	APWR	Aux power switch
7	CHRG	Charging indicator		8	BZR	Buzzer PWM
9	PLED	Main power		10	SLED	Status
11	Vpp	Battery 12 VDC		12	Gnd	Signal/power common
13	Gnd	Signal/power common		14	HTXD	HOST serial transmit
15	HDSR	HOST serial enabled		16	HRCV	HOST serial receive

Joystick Port

An 8-position microfit socket provides signal lines for connection to an analog joystick. Indicated lines (*) are shared on other connectors.

Table 27. Joystick connector (8-pos microfit)

PIN	SIGNAL	DESCRIPTION		PIN	SIGNAL	DESCRIPTION
1	Vcc	5 VDC		**2**	FB0	Fire button 0
3	*AN4	A/D port 4; Y-axis		**4**	Gnd	Signal common
5	*AN3	A/D port 3; X-axis		**6**	FB1	Fire button 1
7		nc		**8**		nc

Appendix B

Power Distribution

*Activ*Media Robotics' original H8S-based Pioneer 2 robots have two separate boards which interface with the H8S microcontroller and provide power for the motors as well as conditioned power and signal paths for the standard and accessory onboard electronics. The new *Plus*-series Pioneer 2 robots and the Pioneer 3s have just a single Motor-Power Board. Consult *Appendix A* for H8S-controller and User Control Panel interface details.

PIONEER 3 AND 2-PLUS MOTOR-POWER BOARD

The new Motor-Power Board for the Pioneer 2-AT8 *Plus,* –DX8 *Plus,* and all Pioneer 3 robots contains all the features of the two-board legacy system and lots more.

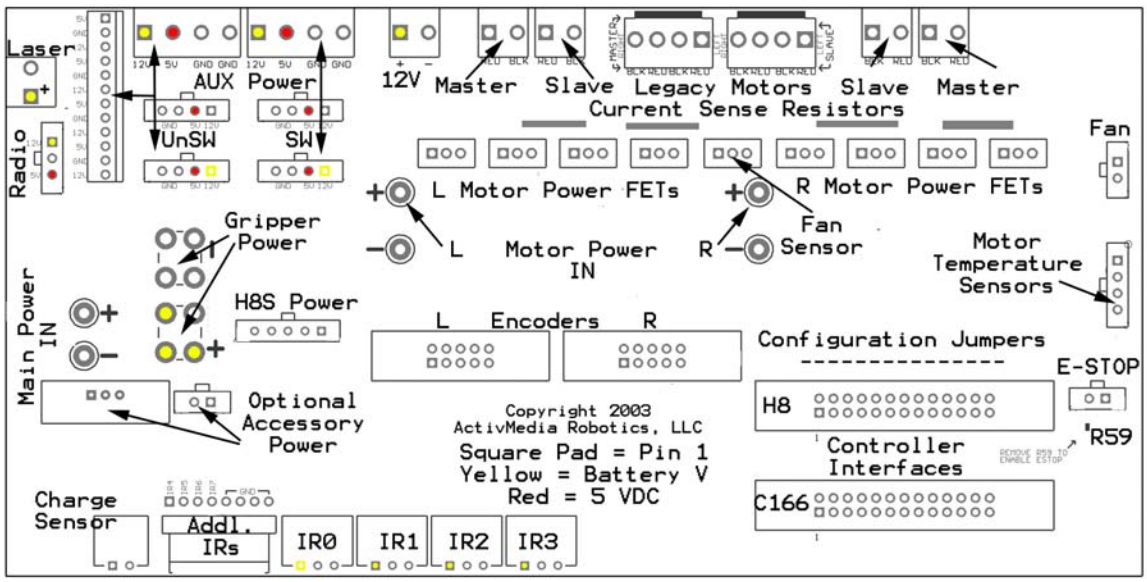

Figure 26. New Pioneer Motor-Power Board

Configuration for Current and Temperature Sensing

The motor drivers are configured to limit 10A per motor, and to share the drivers with both motors on each side of the AT. Accordingly, there are two additional motor-current sense resistors added to the AT versus DX board: R3 and R26, as well as R1 and R2.

The new Motor-Power board also has a set of 0-ohm resistor pads that may be configured to engage the analog-to-digital input ports AN1 and AN2. By adding jumpers to R60 and R62, for example, the board is configured to sense motor current draw on AN1 and AN2, respectively.

Instead, by jumpering R77 and R78 and by attaching temperature sensors to two motors via the Motor Temperature Sensors connector, the AN1 and AN2 ports respectively may be used to protect against motor overheating. This configuration is currently enabled in the new ATs, but not yet supported in AROS.

Table 28. Motor Temperature Sensors Connector (4-pos microfit)

PIN	SIGNAL	DESCRIPTION
1	Vcc	5 VDC
2	T2	To AN2-based temp sensor circuit
3	T1	To AN1-based temp sensor circuit
4	GND	Signal/power common

Otherwise, a jumper across R76 connects the AN1 port to the Fan Sensor system that is attached to the FET heat sink. Note, too, that with or without attachment of AN1 via R76, but with the heat sensor in place, a fan may be attached and activated whenever the motor-driver FETs get overheated, as implemented in all new AT systems.

Controller Power and Interface

Individual 26-pos IDC connectors and cables provide signal for the new H8S-based microcontroller or the legacy C166-based microcontrollers. A separate cable and connector provides for the H8S microcontroller and sonar power. Power and signal are shared on the C166 controller connector.

Table 29. H8S Power connector (5-pos microfit)

PIN	SIGNAL	DESCRIPTION
1	Vbat	Battery power
2	Gnd	Power common
3	Vcc	5 VDC for sonar
4	Vcc	5 VDC for sonar
5	nc	No connection

Radio, Auxiliary, and User Power Connectors

Various connectors provide conditioned 5 VDC @ 1.5A total and unconditioned battery power for the variety of accessories and custom user attachments. Some are AUX and RADIO power switched from the User Control Panel. And some are for Use the 12-position latchlock connector for legacy installations. Otherwise, screw-down auxiliary user-power connectors make custom attachments easy. Four-position microfit connectors also provide AUX power for standard accessories.

Table 30. User Control Panel-switched radio power connector (3-pos microfit)

PIN	SIGNAL	DESCRIPTION
1	Vpp	Radio switched battery 12 VDC
2	Gnd	Power common
3	Vcc	Radio switched 5 VDC

Table 31. User Control Panel-switched and unswitched Aux power connectors (4-pos microfits and screw-down terminal blocks)

PIN	SIGNAL	DESCRIPTION
1	Vpp	Aux switched battery 12 VDC
2	Vcc	Aux switched 5 VDC
3	Gnd	Power common
4	Gnd	Power common

Table 32. User Power connector (12-pos latchlock; unswitched)

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

IR Signal and Power

Originally available on the Motor-Power Interface Board and now integrated on the new Motor-Power board, four connectors provide power and signal for fixed-range IR sensors. A separate connector provides signal path for an additional four IR sensors.

Table 33. IR power and signal connectors (3-pos microfits)

PIN	SIGNAL	DESCRIPTION
1	Vpp	Battery 12 VDC
2	IRn	Switching signal
3	Gnd	Power/signal ground

Table 34. Additional IR connector (8-pos latchlock 0.1 header)

PIN	SIGNAL	DESCRIPTION
1-4	IR4-7	IR signals
5-8	GND	Signal common

LEGACY MOTOR-POWER

The legacy Motor-Power system is a two-board set that connects the H8S controller's control signals to the original P2 Motor-Power board, and provides connections for switched radio and auxiliary power, power and digital inputs for IRs, charge-detection port, and emergency stop detector. See the H8S-controller board in *Appendix A* for interface connection specifications.

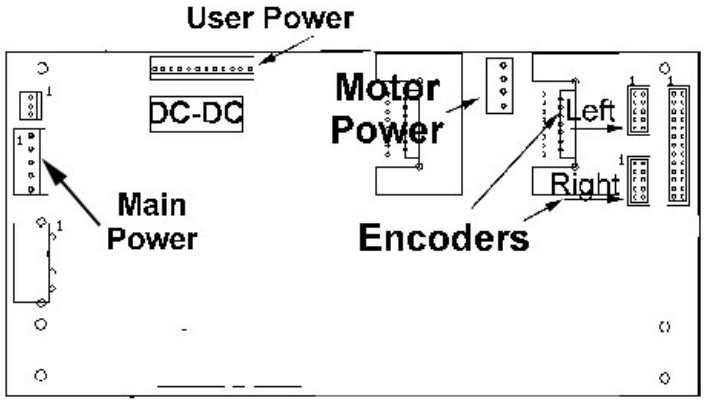

Figure 27. The Original P2 Motor-Power Board

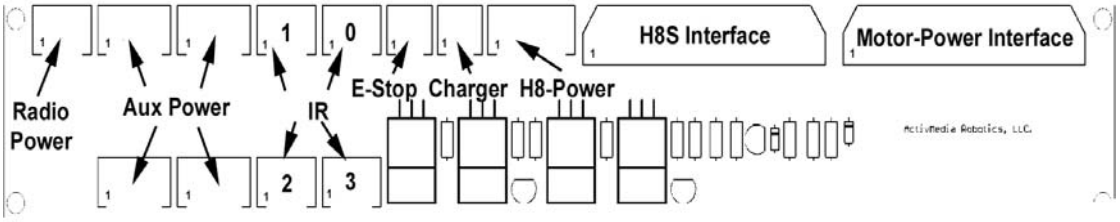

Figure 28. Motor-Power interface board

Appendix C

RADIO MODEM SETTINGS

The radio modem-based wireless serial accessory comes pre-configured for use with your *Activ*Media robot for client-server connections. One modem comes installed in the robot (robot's HOST serial port pins 7 and 8 jumpered; powered 5 VDC from RADIO switch). All you need to do is attach the other radio modem to a free serial port on your PC and provide power—no other setup is required.

You may examine and alter your radio modem settings, such as to match a new baud rate. Use Hyperterminal, minicom, or other simple terminal program. Default settings are DCE for the host and DTE for the H8S-based Pioneers, and 9,600 baud, 8 bits data, 1 stop bit, no parity. Once connected, all modem control commands begin with "WM". For example, "WMS2" at the host connects the host modem to the robot's modem.

Command	Description
WMBx	Set up the default baud rate. x=1 : 115200 , 2 : 57600 , 3 : 38400 , 4 : 19200 , 5 : 9600.
WMD	Disconnect the radio link established previously.
WMEx	Set up echo and response function. x= 'A' ~ 'P'.
WMFxxxx	Set up the maximum frame length. xxxx must be at most a 4-digit decimal number and ranging from 1 to 1024.
WMIxxxxxx	Set up the group identification code. xxxxxx must be exactly a 6-digit hexadecimal number. The group ID is used to ensure that each connection within the group can be created successfully only if the group ID is the same.
WMJxxx…	Change the identification name to xxx…. The length of xxx… cannot exceed 32 letters.
WML	List current setting. The format is as follows:
WMMxxx	Set up my address. xxx must be at most a 3-digit decimal number and ranging from 1 to 255.
WMN	From command mode return to data mode.
WMOxxx…	Set up the partner PN code. See WMP.
WMPxxx…	Older units have to set up your own PN codes. xxx... must be exactly a 32-digit hexadecimal number. Newer units xxx is a number 1-23; match with pair modem.
WMQx	Query remote setting.
WMRx	Set up the remote output destination. x=P : printer port, x=R : RS-232 port.
WMSxxx	Create a radio link with the partner addressed by xxx. Xxx must be at most a 3-digit decimal number and ranging from 1 to 255. After establishing the link, the async. interface will enter data transmission mode until receiving ESCAPE sequence. The ESCAPE sequence consists of three contiguous '|' characters and a <CR>. After the reception of ESCAPE sequence, the async. interface will re-enter into command mode. Note that robot's modem xxx is 2.
| | | followed by <CR> key	From data mode escape to command mode. A delay of 100 ms is needed between the return and any following data input.

Appendix D

SERIAL ETHERNET SETTINGS

The Ethernet-to-Serial device settings are made at the factory and stored in FLASH. Pressing and holding the `test` button for more than five seconds restores those settings. Server name: AMR-EW-1

Wireless

SSID: WaveLAN Network

Mode: Infrastructure

Speed: 1 Mbps

TCP/IP

Address: 192.168.1.11 (.12, .13, … for successive units on a single order)

Gateway: 192.168.1.1

Subnet mask: 255.255.255.0

Boot protocol: static

Serial Port (S1)

Disable console mode

Disable flow control

Serial port service (AMR-EW-1_S1)

Disable queuing

TCP port 8101

Disable NetWare

Misc Network

Disable AppleTalk

Disable POP3

Disable SMTP

LAN IP SETTINGS

You need to modify your Ethernet-to-Serial device network setting in order to use it with your own LAN and Access Point. You have two ways to change those settings: From a serial console or from the device's support webpage:

Console mode:

1. Power off

2. Attach a cross-over serial cable between your PC and the serial port on th e device
3. Start minicom (Linux), HyperTerminal (Windows) or comparable serial console on your PC
4. Serial settings are 115,200 baud, 8 bits, one stop, no parity and hardware handshaking.
5. Hold in the test button and power the device
6. Press the Return key to get the Local> prompt
7. Type:
    8. set ip address aa.bb.cc.dd
    9. set ip router aa.bb.cc.dd
    10. set ip subnet aa.bb.cc.dd
    11. save
    12. init
    13. exit
14. Restart the device

Webpage

1. Start web browser and access http://192.168.1.11
2. The default password is access
3. Select Configure TCP/IP
4. Change the fields for the IP address, subnet mask, and gateway
5. Click submit
6. Restart the device

Peer-to-Peer Networking

If you don't have an established LAN or access to the wireless network, you may operate your robot wirelessly directly from a PC that contains wireless Ethernet in what is known as peer-to-peer mode.

1. From console mode (see above), at the Local> prompt
2. Type:
    3. set enet mode adhoc
    4. save
    5. init
    6. exit
7. Restart the device

1. Or from the webpage (access as above)
2. Select Configure WiFi
3. Choose Ad-hoc from the Mode menu
4. Submit
5. Restart the device

Appendix E

SPECIFICATIONS

	DXe	DX8/P3DX	AT/AT8	Perf PB	PB V1	CE
Physical Characteristics						
Length (cm)	44.5	44.5	50	47	47	44
Width (cm)	40	40	49	38	38	33
Height (cm)	24.5	24.5	24	124	104	22
Clearance (cm)	6.5	6.5	5.5	3.5	3.5	5.1
Weight (kg)	9	9	14	21	19	9
Payload (kg)	23	25	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	8-10	8-10	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 pneumatic	4 pneumatic	2 pneumatic	2 solid rubber	2 solid rubber
diam (mm)	191	191	220	191	165	165
width (mm)	50	50	75	50	37	37
Caster (mm)	75	75	na	75	75	75
Steering	Differential	Differential	Skid	Differential	Differential	Differential
Gear ratio	19.7:1	38.3:1	85.2:1	38.3:1	38.3:1	19.7:1
Swing (cm)	32	32	40	33	32	32
Turn (cm)	0	0	0	0	0	0
Translate speed max (mm/sec)	1,800	1,400	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**	**DX8/P3DX**	**AT/AT8**	**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 forward @ 20° intervals)	na	na	na	8	8	na
Encoders (2 ea) counts/rev	39,400	76,600	34,000	76,600	76,600	39,400
counts/mm	66	128	49	128	148	76
counts/rotation	18,400	33,500	22,500	33,500	39,000	18,400
Controls and Ports						
Main Power	✓	✓	✓	✓	✓	✓
Charge Port	✓	✓	✓	✓	✓	✓
Joydrive	Optional	Optional	Standard	Standard	Optional	Optional

Warranty & Liabilities

Your *Activ*Media robot is fully warranted against defective parts or assembly for one year after it is shipped to you from the factory. Accessories are warranted 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.

19 Columbia Drive
Amherst, NH 03031
(603) 881-7960
(603) 881-3818 fax
http://www.mobilerobots.com