Pioneer 2
MOBILE ROBOTS
with Pioneer 2 Operating System Servers

OPERATIONS MANUAL
# Table of Contents

**CONGRATULATIONS** .................................................................................................................. 1

**PIONEER PACKAGE** .................................................................................................................... 2
  - Basic Components (all shipments) .................................................................................. 1
  - Optional Components and Attachments (partial list) ................................................... 1
  - User-Supplied Components / System Requirements .................................................. 2

**ADDITIONAL RESOURCES** ....................................................................................................... 2
  - Pioneer Software ........................................................................................................ 2
  - Pioneer Newsgroup ..................................................................................................... 2
  - Support ....................................................................................................................... 3

**WHAT IS PIONEER?** ............................................................................................................... 4
  - Client Software .......................................................................................................... 4
  - The Pioneer Legacy .................................................................................................... 5
  - Modes of Operation .................................................................................................... 6

**SPECIFICATIONS & CONTROLS** ............................................................................................... 8
  - Physical Characteristics ............................................................................................. 8
  - Main Components ...................................................................................................... 8
    - Console and Deck .................................................................................................... 9
    - Body ...................................................................................................................... 9
    - Nose ..................................................................................................................... 10
    - Accessory Panels .................................................................................................. 10
    - Sonar Arrays ......................................................................................................... 10

**MOTORS AND POSITION ENCODERS** .................................................................................... 10

**SONARS** .................................................................................................................................. 10

**BATTERIES AND POWER** ....................................................................................................... 11

**ELECTRONICS** .......................................................................................................................... 11
  - Power/Motor Board ..................................................................................................... 12
  - Microcontroller ......................................................................................................... 12
  - Sonar Boards ........................................................................................................... 12

**CONTROLS, PORTS, AND INDICATORS** .................................................................................. 12
  - Main Power, Fuse, and Indicator .................................................................................. 12
  - Recharge/Power Port ................................................................................................... 12
  - Liquid-Crystal Display ............................................................................................... 13
  - Contrast .................................................................................................................... 13
  - SERIAL ...................................................................................................................... 15
  - RADIO ....................................................................................................................... 15
  - FLASH ....................................................................................................................... 15

**SAFETY WATCHDOGS AND CONFIGURATION** .................................................................... 15

**QUICK START** ............................................................................................................................ 16
  - Preparative Assembly ................................................................................................. 16
    - Saphira Client Installation ...................................................................................... 16
  - Saphira Client Start-Up .............................................................................................. 17
  - Pioneer Cold Start-Up ............................................................................................... 17
  - RADIO ON .................................................................................................................. 17
  - Starting Saphira Client/Pioneer Server Communications ......................................... 17
  - A Successful Connection .......................................................................................... 18
  - Operating the Saphira Demo Client .......................................................................... 18
  - Disconnecting Serial Communications (intentionally or unintentionally) ............. 20
  - Quickstart Troubleshooting ...................................................................................... 20


JOYSTICK DRIVE AND SELF-TEST MODES ................................................................. 22

JOYSTICK DRIVE .................................................................................................................. 22
JOYSTICK OPERATION .......................................................................................................... 22
ENGAGING SELF-TESTS ..................................................................................................... 22
MOTORS TEST ....................................................................................................................... 23
SONAR TEST ........................................................................................................................ 23
DIGIN TEST ........................................................................................................................ 23
DIGOUT TEST ....................................................................................................................... 24
ANALOG TESTS ................................................................................................................... 24
BUMPS ................................................................................................................................ 24
USER PWMs ........................................................................................................................ 25
P2 Gripper .............................................................................................................................. 25
COMPASS ............................................................................................................................. 25

PIONEER 2 OPERATING SYSTEM ......................................................................................... 26

COMMUNICATION PACKET PROTOCOL ............................................................................. 26
Packet Data Types .............................................................................................................. 27
Packet Checksum ............................................................................................................... 27
Packet Errors ..................................................................................................................... 27
SERVER INFORMATION PACKETS .................................................................................... 28

CLIENT COMMANDS .......................................................................................................... 29
Client Command Argument Types ..................................................................................... 30
Saphira Client Command Support .................................................................................... 31

PROGRAMMING P2OS ........................................................................................................ 31
Synchronization—SYNC .................................................................................................... 31
Autoconfiguration .............................................................................................................. 32
Opening the Servers—OPEN ............................................................................................. 32
Keeping the Beat—PULSE ................................................................................................. 32
Closing the Connection—CLOSE ...................................................................................... 32

MOVEMENT COMMANDS .................................................................................................. 32
PIONEER IN MOTION .......................................................................................................... 33
Position Integration ............................................................................................................ 33

SONARS ............................................................................................................................... 34
INPUT / OUTPUT (I/O) ......................................................................................................... 34
Digin, Timer, and ADSEL .................................................................................................... 34
DIGOUT and PSUPOS ........................................................................................................ 35

BUMPSTALL ......................................................................................................................... 35
EXTENDED PACKETS ......................................................................................................... 36
Packet Processing .............................................................................................................. 36
Configuration Packets ...................................................................................................... 36
Encoder Packets ................................................................................................................ 36
AUX Serial Packets ........................................................................................................... 36

STANDALONE MODE .......................................................................................................... 38

UPDATING & RECONFIGURING P2OS ................................................................................ 39

WHERE TO GET P2OS SOFTWARE .................................................................................... 39
INSTALLING THE P2OS UTILITIES .................................................................................... 39

UPDATING P2OS .................................................................................................................. 39

Step 1: Serial Connection from Computer to Pioneer ....................................................... 39
Step 2: Enable FLASH ...................................................................................................... 39
Step 3: Put Pioneer into Boot Mode .................................................................................. 39
Step 4: Run p2osdl ............................................................................................................ 40

Download Troubleshooting ............................................................................................... 40

CONFIGURING P2OS OPERATING PARAMETERS ............................................................... 40

Steps 1–3: Preparing Pioneer 2 for Configuration ................................................................. 40
Step 4: Run p2oscf ............................................................................................................. 40
Step 5: Changing the Configuration Parameters ................................................................. 40
Editing P2OS Parameters .................................................................................................... 41

Saving and Restoring ......................................................................................................... 43
Congratulations

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

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

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

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

Pioneer Package

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

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

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

Basic Components (all shipments)

- One fully assembled Pioneer 2 Mobile Robot
- CD-ROM containing licensed copies of Pioneer 2 software and documentation
- Hex wrenches and assorted replacement screws
- Replacement fuse
- Set of manuals
- Registration and Account Sheet

Optional Components and Attachments (partial list)

- Battery charger (some contain power receptacle and 220VAC adapters)
- Onboard PC computer with PC104+ bus, hard-drive and other accessories
- Radio Ethernet and/or Modems
  - One mounted inside Pioneer 2
  - Companion radio for LAN or basestation connection
- Supplementary and replacement batteries
- 3-Battery Simultaneous Charge Station (110/220 VAC)
- P2 Gripper
- ActivMedia Color Trackings System
- Range-Finding StereoCam System
- Pan-Tilt-Zoom Surveillance System
- Custom Vision System
Congratulations

- Range-finding Laser
- GPS System
- Compass
- Bumper Ring
- Serial cables for external connections

**User-Supplied Components / System Requirements**

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

**Additional Resources**

Every new Pioneer customer gets three additional and valuable resources:

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

**Pioneer Software**

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

http://robots.activmedia.com

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

**Pioneer Newsgroup**

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

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

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

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

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

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

pioneer-support@activmedia.com

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

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

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

(See Chapter 8, Maintenance & Repair, for more details.)
What Is Pioneer?

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

These are truly off-the-shelf, “plug and play” mobile robots, containing all of the basic components for sensing and navigation in a real-world environment, including battery power, drive motors and wheels, position / speed encoders, and integrated sensors and accessories. They are all managed via an onboard microcontroller and mobile-robot server software.

Your Pioneer 2 also has a variety of expansion power and I/O ports for optional, custom attachments. These include an addressable I/O bus for up to 16 devices, two RS-232 serial ports, eight digital I/O ports, five A/D ports, PSU controllers and more — all accessible through a common application interface to the robot server software, P2OS.

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

![Figure 2-1. The Basic Components of a Pioneer 2 DX](image)

Client Software

Pioneer 2 comes with a variety of mobile robotics development software. Currently available Pioneer client software for the computing platform of your choice (UNIX, Linux, Macintosh, Win32) includes:

- The Saphira client development suite with Colbert
- Ayllu subsumption-like system, with multi-agent extensions
- Pioneer simulator
- Pioneer LOGO (P-LOGO)
- Pioneer Application Interface (PAI)

Versions and all updates for supported computing platforms are available to password-registered customers for download from our software website (see Congratulations, Chapter 1):

[http://robots.activmedia.com](http://robots.activmedia.com)
Currently supported client operating systems include most UNIX, Apple Macintosh, SunOS and Solaris from Sun Microsystems, Linux, Silicon Graphics’ IRIX, and Microsoft’s Win32 (32-bit) Windows.

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

Ayllu is a tool for development of behavior-based control systems for mobile robots. It extends subsumption-style message passing to the multi-robot domain, provides for a wide variety of behavior-arbitration techniques, and allows a great deal of run-time system flexibility. This includes a dynamic reconfiguration of behavior structure and redistribution of tasks across a group of robots as determined by either task constraints or changing availability of resources. See the *Ayllu Programmer’s Manual* for details.

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

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

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

![Image of client/server mode]

**Figure 2-2. In client/server mode, Pioneer 2 requires a host computer connection.**

**The Pioneer Legacy**

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

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

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

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

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

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

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

Sporting a more holonomic body, larger wheels and stronger motors for better indoor performance, the Pioneer 2-DX and CE models, like Pioneer 1, are two-wheel, differential-drive mobile robots. (The least expensive CE model comes with fewer options and expansion capabilities than the 2-DX.)

The four-wheel drive Pioneer 2-AT has independent motor-drivers, as well as a stall-detection systems. Unlike its predecessor, the Pioneer 2-AT comes with inflatable tires and metal wheels for much more robust operation in rough terrain, as well as the ability to carry nearly 45 kilograms (90 lbs) of payload and climb a 60-percent grade.

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

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

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

**Modes of Operation**

You may operate Pioneer 2 in one of four modes:

- Joystick Drive
- Self-test
- Standalone
- Server Mode

The Pioneer 2 controller comes with 32K flash-programmable, read-only memory (flash-ROM) as part of its Siemens 88C166 microprocessor, and an additional 32K of dynamic RAM: 64K total memory space for your standalone robotics programs.

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

In conjunction with client software, such as Saphira, Ayllu, or PAI running on the onboard PC or a user-supplied computer, P2OS lets you take advantage of modern client/server and robot-control technologies to perform advanced robot tasks. (See Chapter 6, *Pioneer 2 Operating System*, for details.)
Most users run Pioneer in this server mode, because it gives them quick, easy access to the robotics functionality while working in high-level software on a familiar host computer.

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

In fact, the download utilities we provide for you to reprogram the 88C166-based controller's flash-ROM and RAM also are used to update and upgrade your P2OS. We typically provide the upgrades free for download from our website, so be sure to sign up for the pioneer-users newslis. That's where we notify our customers of the upgrades, as well as where we provide access to Pioneer users worldwide.

Finally, we provide onboard software that lets you drive the robot from a tethered joystick. And we provide some test programs that exercise Pioneer 2 microcontroller hardware and software. We examine these modes in some detail in Chapter 5, Joystick Drive and Self-Test Modes.
Specifications & Controls

Pioneer robots may be smaller than most, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and more expensive machines. For example, the Pioneer 2DX and CE’s modest size lends itself to navigation in tight quarters or cluttered spaces, such as classrooms, laboratories and small offices (see Figure 3-1).

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

**Physical Characteristics**

For a complete comparison of Pioneer 2’s physical and operational specifications, see *Appendix E*.

![Figure 3-1. The Pioneer 2 DX/CE physical dimensions and swing radius](image)

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

These characteristics also permit Pioneer 2 to carry extraordinary payloads: The DX can carry up to 23 Kg (50 lbs.) additional weight; the Pioneer 2 AT can carry over 30 Kg (66 lbs.) more! Yet the Pioneer 2 is light-weight enough that it is also as easy to transport as a suitcase—a task made even easier by the Pioneer 2’s built-in handle. And because Pioneer's main components are assembled with Allen hex screws (metric sizes; tools included with robot), you have quick access to interior components and can attach accessories with a minimum of effort.

**Main Components**

All Pioneer 2’s are composed of six main sections:

- Console
- Deck
- Body
- Nose
- Sonar Arrays
- Accessory Panels
**Console and Deck**

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

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

**Figure 3-2. Pioneer 2’s Console and Deck**

Every Pioneer 2 has a Console that consists of a liquid-crystal display (LCD), MOTORS, and RESET control buttons and indicators, and an RS232-compatible serial port at the 9-pin DSUB connector on the front and top of the Deck.

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

**Body**

Pioneer 2’s sturdy, but lightweight aluminum Body houses the robot’s batteries, drive motors, electronics, and other standard components, including the forward and rear sonar arrays. The Body also has sufficient room, with power and signal connectors, to support a variety of robotics accessories inside, including the Fast-Track Color Vision System, radio modems and radio Ethernet, onboard computer, and many more.

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

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

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

Accessory Panels

The Pioneer 2 DX comes with a removable panel on each side of the robot on which you can install accessory connectors and controls. A special sidepanel comes with the onboard PC option, for example, which gives users monitor, keyboard, mouse (serial port), and 10base-T Ethernet access, as well as the means to reset and power the computer.

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

Sonar Arrays

Pioneer 2 supports up to two sonar range-finding arrays. One array, affixed under the front of the Deck and atop the Nose, provides forward- and side-range sensing. The other, an optional sonar array is attached just beneath the rear Deck and provides rearward, as well as side sensing. All arrays contain eight sonars, for a total of 16 sonars around the robot.

Motors and Position Encoders

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

Sonars

Natively, the Pioneer 2 AT and DX support both front and rear sonar arrays (CE front only), each with eight transducers that provide object detection and range information for features recognition, as well as navigation around obstacles. The sonar positions are fixed in both arrays: one on each side, and six facing outward at 20-degree intervals, together providing 360 degrees of nearly seamless sensing. An additional 16 sonars in two banks also are available with accessory hardware.

![Figure 3-3. Pioneer 2 sonar array](image)

---

1 When attached, you will need to first remove the P2 Gripper before removing the Nose.
The sonar firing rate is 25 Hz (40 milliseconds per sonar per array) and sensitivity ranges from ten cm (six inches) to more than five meters (16 feet). (Objects closer than ten cm are not detected.) You may control the sonar’s firing pattern through software; the default is left-to-right in sequence for the forward array (Figure 3-3) and right-to-left on the rear. One sonar from each array “ping” simultaneously.

The driver electronics for the forward and rear sonar sensor arrays each are calibrated at the factory. However, you may adjust their sensitivity and range to accommodate differing Pioneer 2 operating environments. The sonar gain control is on the underside of the sonar driver board, which is attached to the floor of each sonar module.

Both banks’ sensitivity adjustment controls are accessible directly, although you will need to remove the P2 Gripper to access the front sonar, if you have that accessory attached. For the front sonar, for instance, from the underside, locate a hole near the front of the sonar module through which you can see the cap of the sonar-gain adjustment potentiometer. Using a flat-blade screwdriver, turn the gain control counterclockwise to make the sonar less sensitive to external noise and false echoes.

Low sonar-gain settings reduce the 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 sonars are too sensitive, they will “see” the carpet immediately ahead of the robot as an obstacle.

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

**Batteries and Power**

The Pioneer 2 CE has a single, twelve-volt, direct-current, seven ampere-hour (84 watt-hour) sealed lead/acid battery which supplies ample power for its drives, electronics, and accessories. Typical intermittent operation of the motors gives six or more hours of Pioneer 2-CE use.

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

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

If you don’t use the motors, Pioneer 2’s native electronics will run for several days on a single battery charge.

**Balance the batteries in your Pioneer 2.**

IMPORTANT: Batteries have a significant impact on the balance and operation of your robot. For the Pioneer 2-DX and 2-AT, under most conditions, we recommend operating with three batteries. Otherwise, a single battery should be mounted in the center, or two batteries inserted on each side of the battery container.

Typical recharge time using the recommended accessory recharger varies according to the battery’s discharge state; it is roughly equal to three hours per volt. The optional Charge Cube allows simultaneous recharge of three swappable batteries outside the robot.

With the optional high-speed Charger, recharge time is greatly reduced, but care must be taken not to overcharge and thereby damage batteries.

**Electronics**

The Pioneer 2’s standard electronics reside on three main boards: The microcontroller is mounted under the Console Deck; a power / motor controller board is mounted to the battery box inside the robot; and a sonar controller (one for each array) is mounted in the base of the sonar array.
Specifications and Controls

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

Power/Motor Board

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

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

Microcontroller

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

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

Sonar Boards

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

Controls, Ports, and Indicators

Main Power, Fuse, and Indicator

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

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

Recharge/Power Port

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

You should maintain Pioneer 2’s batteries in a charged state above 11 VDC, as indicated on the robot’s LCD. We recommend recharging the battery when it falls below 11 VDC, even though the robot may continue to operate below 10 VDC. The microcontroller will sound a warning when the battery voltage falls below that set level (see Chapter 7, Updating and Reconfiguring P2OS), and the optional computer
power circuitry will automatically shut down the onboard PC. Discharging the batteries to below 10 VDC damages them.

---

**Disengage the motors when recharging the robot.**

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

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

The Pioneer 2 chargers also may be used with the Pioneer 2 Charge Cube accessory.

**Liquid-Crystal Display**

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

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

**Contrast**

A small, contrast-adjustment potentiometer for the LCD is inset next to the display. Make sure the Main Power switch is on and the battery is well charged. Then, using a small, flat-blade screwdriver, turn the adjustment screw to darken or lighten the screen so that the characters are clearly visible under your lighting conditions.
Specifications and Controls

**Figure 3-3. Pioneer 2 Console**

**RESET and MOTORS**

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

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

The MOTORS pushbutton is *NOT* a power switch—it does not directly control power to the motors. P2OS does and your standalone software must.

The MOTORS button and its associated green LED are under software control. Normally, Pioneer 2's motors are disabled when not connected to a client, or when not running in joystick-drive or self-test mode, when running a standalone program on the controller.

Pressing and releasing the white MOTORS button enables/disables the motors when connected with a P2OS client. When first connected with a client, the motors remain disabled (LED flashes) until you press and release the MOTORS button. The green LED should light continuously when the motors are enabled and blink on / off when disengaged.
Under P2OS control when not connected with a client, pressing and releasing the MOTORS button puts Pioneer 2 into joystick-drive mode. A subsequent press/release puts it into self-test mode that exercises the robot's drive, controller, and I/O systems. (See Chapter 5, Joystick and Self-Test Modes, for details.)

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

**SERIAL**

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

Two amber LEDs on either side of the Serial Port flash during actual data activity transmitted from or received by the microcontroller.

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

**RADIO**

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

**FLASH**

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

**Safety Watchdogs and Configuration**

Pioneer’s standard onboard software, P2OS, contains a communications watchdog that will halt motion if communications between a client computer and the server are disrupted for a set time interval, nominally two seconds (watchdog). The robot will automatically resume activity, including motion, as soon as communications are restored.

Also, Pioneer's server software contains a stall monitor. If the drive exerts a PWM pulse that equals or exceeds a configurable level and the wheels fail to turn (stallval), motor power is cut off for a configurable amount of time (stallwait). The server software also notifies the client which side is stalled. When the time elapses, motor power automatically switches back on and motion continues under server control.

Both these “failsafe” mechanisms help ensure that the robot will not damage objects or be electrically damaged during operation. You may reconfigure the communications, drive current, and stall-wait values to suit your Pioneer’s application. (See Chapter 7, Updating & Reconfiguring P2OS, for details.)
Quick Start

Pioneer 2 comes fully assembled and ready for action. This chapter describes how to operate the mobile robot with the Saphira demonstration software. (For more details about programming and operating your Pioneer with Saphira, PAI, Ayllu, or P-LOGO, see their respective programming manuals.)

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

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

![Figure 4-1. Pioneer 2’s P2OS servers need a serial link with a client.](image)

Preparative Assembly

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

Saphira Client Installation

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

http://robots.activmedia.com

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

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

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

For instance, with the mouse, double-click on the $saphira.exe$ icon inside $C:\Saphira\ver61\bin$

on your Windows 95, 98, or NT desktop, or navigate $(cd)$ to the Saphira directory on your UNIX or Linux machine via an X-Window-based shell and type $saphira$, or $./saphira$ to execute the Saphira client software there.

(We’ve set the $\{\text{PATH}\}$ for onboard, Linux-based PC systems that we install for you, so that you need only type $saphira$ in an X-Window shell.)

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

Pioneer Cold Start-Up

Place your Pioneer 2 on the floor in an open space. Slide the Main Power switch up to $\text{ON}$. The red power indicator LED beneath the $\text{FLASH}$ switch on the Console should light. After a short P2OS initialization phase, the LCD on the control panel then displays the current status, the Console serial port baud rate, the current P2OS version number, and the battery charge, in volts. For example,

```
no conn 19.2 kB
P2OS v1.2 13.8*
```

The "heartbeat" asterisk character following the battery voltage should be flashing on and off.

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

$\text{RADIO ON}$

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

Starting Saphira Client/Pioneer Server Communications

After it starts up, resets, or completes the self-test, P2OS enters server disconnect ("noconn") mode — a quiet state in which P2OS awaits communication with a client computer.

To establish a connection with the Saphira client, pull down the $\text{Connect}$ menu of Saphira’s main window on your computer and engage the appropriate serial port: It’s the one that you connected to the robot via a direct cable or through modems (Figure 4-1). With the onboard PC, it is the first serial port.

The port name varies by computer platform, such as $/dev/ttyx$ on a Sun workstation, the modem port on a Macintosh, one of the $\text{COM}$ ports (1–4) on a PC, $/dev/ttySx$ with Linux, or any of the alternative serial ports that hosts the robot-to-computer connection. (See Figure 4-2 on next page.)

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

As synchronization packets are received and echoed by the communications server, they appear sequentially next to the word $\text{sync}$ on the top line of the LCD display on the robot. If these numbers do not appear, the communication line is down or the client is malfunctioning. RESET Pioneer to return it to its
waiting state. The \texttt{sync} number 3 is a special error code indicating you have a noisy communication line; you must reduce the noise level before operating Saphira.

![Saphira connection screen](image)

\textbf{Figure 4-2. Connecting Saphira with Pioneer}

\textbf{A Successful Connection}

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

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

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

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Connected} & \textbf{P2OS v1.8 13.2*} \\
\hline
\end{tabular}
\end{center}

\textbf{Operating the Saphira Demo Client}

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

Collision avoidance is just one of the many mobile robot behaviors available through Pioneer's suite of robotics application-development software. Remember, this section is meant to familiarize you with Pioneer. Please read the respective Pioneer software manuals for details.
Engage Pioneer 2’s motors (white MOTORS button) after connecting with a client or the robot won’t move, no matter how excited you get.

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

In Figure 4-3, Pioneer (center octagon) has identified a corridor and several doors. Notice the small dots, which are recent sonar reflections. The long lines through the sonar reflections are the calculated corridor’s geometry. The rectangles directly ahead of the robot represent an obstacle “detected and of interest” to Pioneer. One of Saphira’s behaviors, by the way, is to have Pioneer seek and traverse the center of a corridor.

![Figure 4-3. The main window of Pioneer’s Saphira client](image)

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

<table>
<thead>
<tr>
<th>KEY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>i, ↑</td>
<td>Increment forward velocity</td>
</tr>
<tr>
<td>m, ↓</td>
<td>Decrement forward velocity</td>
</tr>
<tr>
<td>j, ←</td>
<td>Incremental left turn</td>
</tr>
<tr>
<td>l, →</td>
<td>Incremental right turn</td>
</tr>
<tr>
<td>k, space</td>
<td>All stop</td>
</tr>
<tr>
<td>g</td>
<td>Constant velocity ON/OFF</td>
</tr>
<tr>
<td>c</td>
<td>Obstacle avoidance ON/OFF</td>
</tr>
</tbody>
</table>
When manually joysticking the robot, each keypress moves the robot forward or backward faster or slower and incrementally changes its direction. For instance, when turning, it is often useful to push the left- or right-turn key rapidly several times in a row, because the turn increment is small.

**Disconnecting Serial Communications (intentionally or unintentionally)**

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

![Figure 4-4. Gracefully disconnect the Saphira client from the Pioneer server](image)

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

**Quickstart Troubleshooting**

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

Licensed versions of Saphira come with the robot and can be downloaded by Pioneer customers from the ActivMEDIA ROBOTICS support website, http://robots.activmedia.com.

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

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

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

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

If for some reason, communications sever between the Saphira client and Pioneer server, but both the client
and server remain active, you may revive the connection with little effort: If you are using radio modems,
first check and see if the robot is out of range.

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

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

Beginning with P2OS version 1.1c, the Pioneer 2 may be tethered and manually driven via a common PC joystick. And, P2OS comes with a series of short test routines for the Pioneer 2’s drive motors, sonar, User I/O functions, and accessories. To run either, start up or reset the robot into its P2OS wait state (LCD "no conn"). You may press the Reset button at any time to disable self-test and joystick-drive modes.

**Joystick Drive**

You may attach a common PC joystick to the User I/O port and manually drive the robot. Newer Pioneer 2 ATs come with the joystick connector and internal wiring. An accessory kit is available to adapt older Pioneer 2 ATs and for Pioneer 2 DX and CE models. See Appendix D for details.

**Beware:**

The Joystick and older P2 Gripper mutually interfere at the User I/O port.

Because they share ports on the User I/O connector of the microcontroller, older versions of the P2 Gripper (hasgripper = 1; see Chapter 7, *Updating & Reconfiguring P2OS*, and the *Pioneer 2 Gripper Manual*, version 2 or later) interfere with each other’s operation. One solution is to switch the control lines with a 4PDT toggle switch. See Appendix D for details. Alternatively, there also is an accessory kit available that moves the P2 Gripper controls to the General I/O bus, liberating the User I/O ports for other uses, include the joystick drive. Visit the Pioneer webpages (http://www.activrobots.com) and/or contact sales@activmedia.com for details.

**Joystick Operation**

To drive the Pioneer 2 with a tethered joystick, attach a common PC joystick—one that you might connect to the joyport on a standard sound/game interface card—to the 15-pin DSUB socket of the joystick port. Then, after power ON, press the white MOTORS button once. Listen for a rhythmic, low-tone beep, beep, beep, indicating joystick mode.

The joystick is self-calibrating: When you first enable joystick-drive mode, P2OS detects the joystick’s initial position and saves its values as the centered/stopped position. Accordingly, let the joystick handle find its default centered position before enabling joydrive mode. Also, exit and restart joystick-drive mode after you manually adjust the joystick’s center position.

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

You may adjust the maximum translated and rotational speeds, even disable joystick-drive, through special P2OS configuration parameters. See Chapter 7, *Updating & Reconfiguring P2OS*, for details.

**Engaging Self-Tests**

To enable self-test mode, press the white MOTORS button twice after startup or resetting the robot.

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

As described above, the first press and release puts the robot into joystick-drive mode. A message should appear in the LCD:

```
Press again to begin tests 13.2*
```
And there should be a rhythmic beeping sound coming from the Console speaker. This is to alert you that you may have inadvertently engaged the self-tests. If so, press the red RESET button. If you really intended to run the self-tests, move on...

Note that you may interrupt any test in the sequence and move on to the next test by pressing the white MOTORS button. And you may discontinue testing at any time and return to the P2OS noconn wait state by cycling power or, better, by pressing the red RESET button.

**Motors Test**

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

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

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

```
Left forward
13.2*
```

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

**Sonar Test**

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

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

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

```
FS:#3 034A
RS:#3 7D00 12.4V*
```

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

**DIGIN Test**

A third self-test lets you examine the values of the eight digital input ports (ID0–7) that come with the Pioneer. The state of each input port is mapped into a series of eight digits, whose values each get displayed in the LCD. Each digit, 0 or 1, represents the digital on/off state of a single input port, numbered right-to-left from 0 to 7. For instance, in the example below, digital input ports 1 and 5 are on; the rest are off:

```
DIGIN: 00100010
13.2*
```

The basic Pioneer has no attachments to the I/O ports. Because their normal state is floating, the DIGIN reading will vary. The DIGIN self-test becomes useful, of course, when you have equipment attached to one or more of the digital input ports.
See Appendix A, “C166 Ports and Connections” for the location of the digital input ports (IDO-7) on the Pioneer microcontroller.

**DIGOUT Test**

Press the white MOTORS button to exit the DIGIN self-test and activate the DIGOUT self-test. (If you have a P2 Gripper installed, the DIGOUT self-test gets bypassed.) Like DIGIN, the DIGOUTself-test software maps into series of eight ones and zeros the current state of each the eight digital output ports (OD0-7) and displays the result in the LCD:

```
DIGOUT: 01010101
```

The Digout self-test currently cycles the output bits automatically, setting and resetting alternate ports; consequently, the display cycles between 01010101 and 10101010.

See Appendix A, *C166 Ports and Connections* for the location of the digital output ports (OD0-7) on the Pioneer microcontroller.

**ANALOG Tests**

P2OS versions 1.2 and later let you reuse four of the digital input ports as analog-to-digital (A/D) input ports. And there is one dedicated A/D port on the User I/O connector. The ANALOG self-tests measure the voltage (0-5 VDC) applied to each of the A/D ports and converts it to single byte, 0-255 (0x00 to 0xFF hexadecimal).

Press the white MOTORS button on the Console once again to enable the ANALOG tests. The port number and value of the current reading at that A/D port on the Pioneer microcontroller gets displayed as the two-digit hexadecimal value, beginning with port #1. For example,

```
ANALOG #1: A9
```

Press the white MOTORS button to move on to the next analog port, 1-5. Port #5 is the dedicated and default A/D port. Like DIGIN, the measured value is unreliable unless you connect something to the port. See Appendix A, “C166 Ports & Connections” for the location of the A/D ports on the Pioneer 2 microcontroller's User I/O connector.

**Bumps**

The next two self-tests display the states of the eight bump inputs from the front and rear bumper accessories available for your Pioneer 2. Press the white MOTORS button to move on to the next test.

The state of each input port is mapped into a series of eight digits, whose values each get displayed in the LCD. Each digit, 0 or 1, represents the digital on/off state of a single bumper switch, numbered right-to-left from 0 to 7. For instance in the example below, the Front Bump's input ports 1 and 5 are on; the rest are off, whereas in the Rear Bumps test example, only the second and third bumps are on.

```
Front Bumps 00100010 13.2*
```

```
Rear Bumps 00101100 13.2*
```
The basic Pioneer 2 has no attachments to the bump ports. Because their normal state is floating, the Bump readings may vary (usually all "1"s). If you do have bumpers, only the ports 1-5 currently are active; the others should be 0.

**User PWMs**

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

See Appendix A, *C166 Ports & Connections* for the location of the A/D ports on the Pioneer 2 microcontroller's User I/O and General I/O connectors.

**P2 Gripper**

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

**Compass**

If you own a Pioneer 2 TCM2 or V2XG compass module, this self-test displays its compass heading in the LCD. See the respective manuals for details.
Pioneer 2 Operating System

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

With this client/server architecture, high-level robotics applications developers do not need to know many details about a particular robot server, because the client typically insulates them from this lowest level of control. Some of you, however, may want to write your own robotics control and reactive planning programs, or just would like to have a closer programming relationship with your robot. This chapter explains how to communicate with Pioneer via the P2OS client/server interface. The same P2OS functions and commands are supported in the various client-programming libraries accompanying Pioneer.

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

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

Communication Packet Protocol

P2OS communicates with a client application using a special packet protocols: command packets from client to server, and server information packets (SIPs) from the server to client. Both are bit streams consisting of four main elements (Table 6-1): a two-byte header, a one-byte count of the number of command/data bytes, the client command and its arguments or the server information data, and finally, a two-byte checksum.
Table 6-1. Main elements of PSOS communication packet protocol

<table>
<thead>
<tr>
<th>Component</th>
<th>Bytes</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>2</td>
<td>0xFA, 0xFB</td>
<td>Packet header; same for client and server</td>
</tr>
<tr>
<td>Byte Count</td>
<td>1</td>
<td>N + 2</td>
<td>Number of subsequent data bytes, including checksum word, but not Byte Count. Maximum 200 total bytes.</td>
</tr>
<tr>
<td>Data</td>
<td>N</td>
<td>command or SIB</td>
<td>Client command or server information block (SIB: discussed in subsequent sections)</td>
</tr>
<tr>
<td>Checksum</td>
<td>2</td>
<td>computed</td>
<td>Packet integrity checksum</td>
</tr>
</tbody>
</table>

Packet Data Types

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

Table 6-2. P2OS Communication Packet Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Bytes</th>
<th>Order</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>2</td>
<td>b₀ low byte; b₁ high byte</td>
<td></td>
</tr>
<tr>
<td>word</td>
<td>4</td>
<td>b₀ low byte; b₁ high byte</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td>up to ~200, length-prefixed</td>
<td>b₀ length of string; b₁ first byte of string</td>
<td></td>
</tr>
</tbody>
</table>

Packet Checksum

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

NOTE: The checksum word is placed at the end of the packet, with its bytes in the reverse order of that used for arguments and data; that is, b₀ is the high byte, and b₁ is the low byte.

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

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

Packet Errors

Currently, P2OS ignores a client command packet whose byte count exceeds 200 or has an erroneous checksum. The client should similarly ignore erroneous server information packets.
Pioneer 2 Operating System

P2OS does not acknowledge receipt of a command packet nor does it have any facility to handle client acknowledgment of a server information packet. Consequently, Pioneer client/server communications are as reliable as the physical communication link. A cable tether between the robot and client computer, such as a piggyback laptop, provides very reliable links; radio modem-mediated communication is much less reliable. Accordingly, when designing client applications that may use radio modems, do not expect to receive every information packet intact, nor can you expect the server to accept every command.

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

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

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

Server Information Packets

Like its Pioneer 1 PSOS predecessor, P2OS automatically sends a packet of information over the serial communication port back to the client every 100 milliseconds. The standard P2OS Server Information Packet (SIP) informs the client about a number of the robot’s operating parameters and readings, using the order and data types shown in Table 6-3.

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

Table 6-3. Standard P2OS Server Information Packet (SIP)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>integer</td>
<td>Exactly 0xFA, 0xFB</td>
</tr>
<tr>
<td>Byte Count</td>
<td>byte</td>
<td>Number of data bytes + 2; must be less than 201 (0xC9)</td>
</tr>
<tr>
<td>Status</td>
<td>byte = 0x3S; where S = sfSTATUSSTOPPED (2) Motors stopped</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sfSTATUSMOVING (3) Motors moving</td>
</tr>
<tr>
<td>Xpos</td>
<td>unsigned integer (15 ls-bits) Wheel-encoder integrated coordinates; platform-dependent units; multiply by DistConvFactor in the parameter file to convert to millimeters</td>
<td></td>
</tr>
<tr>
<td>Ypos</td>
<td>unsigned integer (15 ls-bits)</td>
<td></td>
</tr>
<tr>
<td>Th pos</td>
<td>signed integer</td>
<td>Orientation in platform-dependent units—multiply by AngleConvFactor for degrees.</td>
</tr>
<tr>
<td>L vel</td>
<td>signed integer</td>
<td>Wheel velocities (respective Left and Right) in platform-dependent units—Multiply by VelConvFactor to convert to millimeters per second</td>
</tr>
<tr>
<td>R vel</td>
<td>signed integer</td>
<td></td>
</tr>
<tr>
<td>Battery</td>
<td>byte</td>
<td>Battery charge in tenths of volts</td>
</tr>
</tbody>
</table>
Motor stall and bumper accessory indicators. Bit 0 of the lsbyte is the left wheel stall indicator = 1 if stalled. Bits 1-5 of that same byte correspond to the bump switch states (1=on) for the rear bumpers accessory. Bit 0 of the msbyte is the right wheel stall; the bits 1-5 of that same msbyte correspond to the front bumpers switch states.

Control

signed integer

Setpoint of the server’s angular position servo—multiply by AngleConvFactor for degrees

PTU

unsigned integer

Pulse width of position servo #1

Compass

byte

Compass heading in 2-degree units

Sonar readings

byte

Number of new sonar readings included in information packet; readings follow:

Sonar number

byte

Sonar number

Sonar range

unsigned integer

Sonar reading—multiply by RangeConvFactor for mm

…rest of the sonar readings…

Timer

unsigned int

Currently selected analog port number 1-5

Analog

byte

User analog input (0-255=0-5 VDC) reading on selected port

Digin

byte

User digital input

Digout

byte

User digital output

Checksum

integer

Checksum (see previous section)

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

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

Client Commands

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

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

Table 6-4. P2OS client command packet

<table>
<thead>
<tr>
<th>Component</th>
<th>Bytes</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>2</td>
<td>0xFA, 0xFB</td>
<td>Packet header; same for client and server</td>
</tr>
<tr>
<td>Byte Count</td>
<td>1</td>
<td>N + 2</td>
<td>Number of subsequent command bytes plus checksum, not including Byte Count. Maximum of 200 bytes.</td>
</tr>
<tr>
<td>Command Number</td>
<td>1</td>
<td>0 - 255</td>
<td>Client command number; see Table 4-4</td>
</tr>
<tr>
<td>Argument Type (command dependent)</td>
<td>1</td>
<td>0x3B or 0x1B or 0x2B</td>
<td>Required data type of command argument: positive integer (sfARGINT), negative integer or absolute value (sfARGNINT), or string (sfARGSTR)</td>
</tr>
<tr>
<td>Argument (command dependent)</td>
<td>n</td>
<td>data</td>
<td>Command argument; integer or string</td>
</tr>
</tbody>
</table>
The P2OS command is comprised of a one-byte command number optionally followed by, if required by the command, a one-byte description of the argument type and the argument value.

**Client Command Argument Types**

There are three different types of P2OS client command arguments: positive integers two bytes long, negative integers two bytes long, and NULL-terminated strings consisting of as many as 196 characters.

**Table 6-5. P2OS/PSOS command set**

<table>
<thead>
<tr>
<th>Command</th>
<th>#</th>
<th>Args</th>
<th>Description</th>
<th>PSOS</th>
<th>P2OS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Before Client Connection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYNC0</td>
<td>0</td>
<td>none</td>
<td>Start connection; P2OS echoes synchronization commands back to client.</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>SYNCl</td>
<td>1</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYNC2</td>
<td>2</td>
<td>none</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>After Established Connection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PULSE</td>
<td>0</td>
<td>none</td>
<td>Client pulse resets server watchdog</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>OPEN</td>
<td>1</td>
<td>none</td>
<td>Starts the controller</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>CLOSE</td>
<td>2</td>
<td>none</td>
<td>Close server and client connection</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>POLLING</td>
<td>3</td>
<td>string</td>
<td>Set sonar polling sequence</td>
<td>3.9</td>
<td>1.0</td>
</tr>
<tr>
<td>ENABLE</td>
<td>4</td>
<td>int</td>
<td>Enables/disables the motors</td>
<td>–</td>
<td>1.0</td>
</tr>
<tr>
<td>SETA</td>
<td>5</td>
<td>signed int</td>
<td>Resets translational acceleration parameter, if positive, or deceleration, if negative; in millimeters per second (^2)</td>
<td>–</td>
<td>1.0</td>
</tr>
<tr>
<td>SETV</td>
<td>6</td>
<td>int</td>
<td>Reset maximum translational velocity, in millimeters per second</td>
<td>4.8</td>
<td>1.0</td>
</tr>
<tr>
<td>SETO</td>
<td>7</td>
<td>none</td>
<td>Resets server to 0,0,0 origin</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>SETRV</td>
<td>10</td>
<td>int</td>
<td>Resets maximum rotational velocity in degrees per second</td>
<td>4.8</td>
<td>1.0</td>
</tr>
<tr>
<td>VEL</td>
<td>11</td>
<td>int</td>
<td>Move forward (+) or reverse (-) at millimeters per second</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>HEAD</td>
<td>12</td>
<td>int</td>
<td>Turn to absolute heading; 0-359 degrees</td>
<td>4.2</td>
<td>1.0</td>
</tr>
<tr>
<td>DHEAD</td>
<td>13</td>
<td>int</td>
<td>Turn relative to current heading; + degrees</td>
<td>3.x</td>
<td>1.0</td>
</tr>
<tr>
<td>SAY</td>
<td>15</td>
<td>string</td>
<td>As many as 20 pairs of duration (20 ms increments) /tone (half-cycle) pairs; int is string length</td>
<td>4.2</td>
<td>1.0</td>
</tr>
<tr>
<td>CONFIG</td>
<td>18</td>
<td>int</td>
<td>Request configuration SIP</td>
<td>–</td>
<td>1.4</td>
</tr>
<tr>
<td>ENCODER</td>
<td>19</td>
<td>int</td>
<td>Request continuous stream (&gt;0) or tell to stop sending (0) encoder SIPs</td>
<td>–</td>
<td>1.4</td>
</tr>
<tr>
<td>RVEL</td>
<td>21</td>
<td>signed int</td>
<td>Rotate at + degrees per second</td>
<td>4.2</td>
<td>1.0</td>
</tr>
<tr>
<td>DCHEAD</td>
<td>22</td>
<td>int</td>
<td>Colbert relative heading setpoint; + degrees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETRA</td>
<td>23</td>
<td>int</td>
<td>Sets rotational (+)acceleration or (-)deceleration, in millimeters per second (^2)</td>
<td>–</td>
<td>1.0</td>
</tr>
<tr>
<td>SONAR</td>
<td>28</td>
<td>int</td>
<td>Enable/disable the sonar</td>
<td>–</td>
<td>1.0</td>
</tr>
<tr>
<td>STOP</td>
<td>29</td>
<td>none</td>
<td>Stops robot (motors remain enabled)</td>
<td>–</td>
<td>1.0</td>
</tr>
<tr>
<td>DIGOUT</td>
<td>30</td>
<td>int</td>
<td>Msbits is a byte mask that selects output port(s) for changes; lsbits set (1) or reset (0) the selected port.</td>
<td>4.2</td>
<td>1.2</td>
</tr>
<tr>
<td>VEL2</td>
<td>32</td>
<td>int</td>
<td>Independent wheel velocities; lsb=right wheel; msb=left wheel; PSOS is in +/4mm/sec; P2OS in 2</td>
<td>4.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
**Pioneer Mobile Robots**

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRIPPER</td>
<td>33</td>
<td>int</td>
<td>Pioneer 1 and Pioneer 2 Gripper server command. See the Pioneer Gripper manuals for details.</td>
<td>4.0</td>
</tr>
<tr>
<td>ADSEL</td>
<td>35</td>
<td>int</td>
<td>Select the A/D port number for analog value in SIP. Selected port reported in SIP Timer value.</td>
<td>1.2</td>
</tr>
<tr>
<td>GRIPPERVAL</td>
<td>36</td>
<td>int</td>
<td>Pioneer 2 gripper server value. See P2 Gripper Manual for details.</td>
<td>–</td>
</tr>
<tr>
<td>PTUPOS</td>
<td>41</td>
<td>int</td>
<td>Msb is the port number (1-4) and lsb is the pulse width in 100µsec units PSOS or 10µsec units P2OS</td>
<td>4.5</td>
</tr>
<tr>
<td>TTY2</td>
<td>42</td>
<td>string</td>
<td>Send string argument to serial device connected to AUX port on microcontroller</td>
<td>4.2</td>
</tr>
<tr>
<td>GETAUX</td>
<td>43</td>
<td>int</td>
<td>Request to retrieve 1-200 bytes from the aux serial channel; 0 flushes the aux serial input buffer.</td>
<td>–</td>
</tr>
<tr>
<td>BUMPSTALL</td>
<td>44</td>
<td>int</td>
<td>Stop and register a stall if front (1), rear (2) or either (3) bump-ring contacted. Off (default) is 0.</td>
<td>–</td>
</tr>
<tr>
<td>TCM2</td>
<td>45</td>
<td>int</td>
<td>TCM2 Module commands; see P2 TCM2 Manual for details.</td>
<td>–</td>
</tr>
<tr>
<td>E_STOP</td>
<td>55</td>
<td>none</td>
<td>Emergency stop, overrides deceleration</td>
<td>–</td>
</tr>
<tr>
<td>STEP</td>
<td>64</td>
<td>none</td>
<td>Single-step mode (simulator only)</td>
<td>3.x</td>
</tr>
</tbody>
</table>

The byte order is least-significant byte first. Negative integers are transmitted as their absolute value (unlike information packets, which use sign extension for negative integers; see below). The argument is an integer, a string, or nothing, depending on the command.

**Saphira Client Command Support**

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

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

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

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

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

**Programming P2OS**

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

**Synchronization—SYNC**

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

When first started, the Pioneer 2 robot is in a “wait” state; P2OS listens for communication packets over its designated port. To establish a connection, the client application must send a series of three synchronization packets through the host communication port—SYNC0, SYNC1, and SYNC2, in succession, and retrieve the server responses.
P2OS responds to each client command, forming a succession of identical synchronization packets. The client should listen for the returned packets and only issue the next synchronization packet after it has received the echo.

**Autoconfiguration**

P2OS automatically sends configuration information back to the client in the last sync packet (SYNC2). After the SYNC byte, there are three NULL-terminated strings that comprise the robot’s name, class, and subclass. You may uniquely name your Pioneer 2 with a special configuration tool we provide with the robot. The class and subclass also are parameters stored in the robot's flash PROM, but are constants not changed by the user. (See Chapter 7, *Updating and Reconfiguring P2OS* for details.)

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

**Opening the Servers—OPEN**

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

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

**Keeping the Beat—PULSE**

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

It’s good practice to send a PULSE to Pioneer just after opening the servers. And if your client application will be otherwise distracted for some time, periodically issue the PULSE client command to let the server know you are indeed alive and well. If the robot shuts down due to lack of communications traffic, it will revive upon receipt of a client command and automatically accelerate to the last-specified speed at the current heading.

**Closing the Connection—CLOSE**

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

**Movement Commands**

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

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

When P2OS receives a P2OS server-translation or -rotation command, it accelerates the Pioneer robot at the SETA or SETRA rate you program, or the rates preset in the P2OS configuration parameters. Rotational headings are achieved by a trapezoidal velocity function (Figure 6-2). This function is recomputed each time a new heading command is received, making on-the-fly orientation changes possible.

Note that, with P2OS version 1.8 and later, you may override deceleration with the emergency stop (E_STOP) command number 55. Accordingly, with E_STOP, 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.

![Figure 6-2. Pioneer's trapezoidal turning velocity profile](image)

**Position Integration**

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

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

<table>
<thead>
<tr>
<th>Table 6-6. P2OS motion commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rotation</strong></td>
</tr>
<tr>
<td>HEAD</td>
</tr>
<tr>
<td>DHEAD, DCHEAD</td>
</tr>
<tr>
<td>SETRA</td>
</tr>
<tr>
<td>SETRV</td>
</tr>
<tr>
<td><strong>Translation</strong></td>
</tr>
<tr>
<td>VEL</td>
</tr>
<tr>
<td>SETA</td>
</tr>
<tr>
<td>SETV</td>
</tr>
</tbody>
</table>

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

The orientation commands HEAD and DHEAD turn the robot with respect to its internal dead-reckoned angle (Figure 6-3). On start-up, the robot is at the origin (0,0), pointing toward the positive x-axis at 0 degrees. Absolute angles vary between 0 and 360 degrees. As the robot moves, it will update this internal position based on dead-reckoning. The x,y position is always positive, and rolls over at about 3,000 millimeters. So, if the robot is at position (400,2900) and moves +400 millimeters along the y-axis and -600 millimeters along the x-axis, its new position will be (2800,300).
You may reset the internal coordinates to 0,0,0 with the SETO P2OS command.

![Internal coordinate system for P2OS](image)

**Sonars**

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

For example:

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

Use the POLLING command to change the polling sequence of the sonar. Its argument is a null-terminated string of up to 15 sonar numbers. Front sonars are numbered in octal notation \001 through \010, and the rear sonar are octal numbers \011 through \100. You may repeat a sonar number and have it ping more than once per sequence.

For example:

```c
sfRobotComStr(sfCOMPOLLING,"\001\002\001\002\000"); /* ping front left only */
```

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

**Input / Output (I/O)**

Your Pioneer 2 comes with a number of I/O ports that you may use for some of Pioneer’s accessories and for your own custom attachments. See Appendix A, *C166 Ports and Connections* for port locations on the Pioneer microcontroller. The various port states and reading appear in the SIP and may be manipulated with P2OS client commands.

**Digin, Timer, and ADSEL**

The states of the eight digital input ports are mapped as a single byte, whose value is continuously updated and reported in each standard SIP (Table 6-3). For example, examine Saphira's sfRobot structure value digin:

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

When not connected, the digital input ports’ values will vary and change without warning.

The digital input ports 4-7 (ID4-7) also may be used as A/D input ports. Accordingly, their digin states will depend on whether or not the applied voltage exceeds the standard digital CMOS threshold voltage.

Use the client command ADSEL to select the A/D port that is to appear in the P2OS SIP analog value. The default port is #5, the dedicated A/D port on the system. The P2OS SIP reports the currently selected analog input port number.
For example, the first Saphira statement queries for the current analog port and its A/D value. The second Saphira command changes the selected port which value gets reported through the last Saphira command in the example:

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

```c
sfSMessage("Port% reads %i", sfRobot.timer, sfRobot.analog);
Port 2 reads 224
```

**DIGOUT and PSUPOS**

The eight digital output ports on the Pioneer 2 User I/O connector are both reported in the P2OS SIP (Digout value) and controllable with the P2OS commands DIGOUT and PTUPOS. Electrically, the ports express a digital high (1) state as ~5 VDC (Vcc) and low (0) at ~0 VDC (GND).

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

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

240

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

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

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

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

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

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

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

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

**BUMPSTALL**

Because the robots are programmed to climb, it can take up to a second or more to detect object collisions with motor stalls. With P2OS version 1.5, we introduced a responsive way to stop the Pioneer 2 DX or AT within a few milliseconds upon collision with an obstacle: BUMPSTALL. With the robot equipped with forward and/or rear bumpers, you can now program P2OS to immediately stop the robot and notify the client of a stall if any one of the forward or rear bump sensors get triggered.

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

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

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

BUMPSTALL requires supporting bumper ring accessory.
**Extended Packets**

P2OS 1.4 introduces several new Server Information Packets to better support the Pioneer 2 robotics community. On request from the client by a related P2OS command (see Table 6-5), the P2OS server will package and send one or a continuous stream of packet types to the client over the host serial communication line.

**Packet Processing**

It is up to the client to handle all packet types. Extended packets get intermingled with the standard SIP that P2OS sends every 100 or 50 milliseconds. Please consult the respective client application programming manuals for details.

The P2OS extended packets are packaged similarly to the standard Server Information Packet, including header (0xFA, 0xFB) and the checksum. The difference, besides the number and included data, of course, is a different packet type (Status byte). Accordingly, the client processor should simply examine the packet type byte and process each packet accordingly.

A sample packet processor is included with the Saphira distribution—packet.c in the apps/ directory.

**Configuration Packets**

Send the CONFIG command number 18 with any integer argument to request P2OS send back a special server information packet containing the robot's operational parameters. The CONFIG SIP packet type is 32 (0x20). Table 7.2 in the next chapter gives details about the configuration data.

**Encoder Packets**

By issuing the ENCODER command number 19 with a non-zero integer argument, you initiate a continuous stream of ENCODERpac (type 144) SIPs. One ENCODERpac is sent every 100 or 50 milliseconds, depending on the standard packet cycle rate (sInfoCycle; see next Chapter). Discontinue the packets by sending the ENCODER command number 19 with the argument = 0.

The P2OS extended Encoder packet contents are described in Table 6-7.

**Table 6-7. P2OS Encoder Server Information Packet Contents**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>integer</td>
<td>Exactly 0xFA, 0xFB</td>
</tr>
<tr>
<td>Byte Count</td>
<td>byte</td>
<td>Number of data bytes + 2</td>
</tr>
<tr>
<td>Left Encoder</td>
<td>integer</td>
<td>Least significant portion and the most significant portion comprise the current 4-byte raw encoder count from the left drive wheel</td>
</tr>
<tr>
<td>Right Encoder</td>
<td>integer</td>
<td>Least significant portion and the most significant portion comprise the current 4-byte raw encoder count from the left drive wheel</td>
</tr>
<tr>
<td>Checksum</td>
<td>integer</td>
<td>Checksum for packet integrity</td>
</tr>
</tbody>
</table>

**AUX Serial Packets**

The new P2OS 1.4 implements two-way communications through the AUX serial port on the Pioneer 2 microcontroller. With earlier versions of P2OS, you are able to send a string of data (a command or otherwise) from the HOST port-connected client to an AUX port-connected serial device (TTY2 command number 42), but you could not get data back from that device.

With P2OS 1.4 and later, use the P2OS command number 43 (GETAUX) to request feedback from the AUX port. It tells P2OS to retrieve a given number of bytes (command argument value) from the serial device attached to the AUX port, and to package and send those incoming data in a special SERAUXpac (type 176; 0xB0) SIP back to the client via the common HOST serial port.
/* Send command and retrieve response via P2OS AUX serial port */
#define SERAUXpac 0xB0
int serbytes, inbufptr;
char inbuf[200];
...

/* Replace Saphira's default packet processor with our own that can detect and parse SERAUXpac packets */
void
myStartupFn(void)
{
    sfInitProcess(myPackets, "GetAuxPkt");
sfRemoveTask("packets"); /* default packet processing */
    ...
}

/* SERAUXpac packet processor */
void
myPackets(void)
{
    char packet_type;
    if (sfIsConnected)
    {
        while (sfHaveClientPacket())
        {
            packet_type = sfReadClientByte();
            if (packet_type == SERAUXpac)
            {
                /* get aux serbytes into inbuf */
                while (inbufptr++ < serbytes) inbuf[inbufptr] = sfReadClientByte();
            }
            else sfProcessClientPacket(packet_type);
        }
    }
}

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

P2OS maintains a circular buffer for incoming serial data from the AUX port and returns successive portions depending on the number of bytes you request via the GETAUX command, up to 200 total at a time. P2OS waits to collect that number of incoming AUX-port serial bytes before sending the packet to the client. Use the GETAUX command with a zero argument to flush the P2OS circular buffer and reset its pointers.

For example, the following Saphira client code fragment sends a fictitious command to a serial device attached to the Pioneer 2 microcontroller's AUX port and retrieves a 10-byte response from that device (note that ellipses mark the many portions of a complete client that have been deleted for brevity):
**Standalone Mode**

As mentioned in previous chapters, the Pioneer 2 microcontroller’s flash-PROM is reprogrammable. And there is 32K RAM onboard waiting for something to do. Standalone mode also lets you perform experiments in low-level sensing and control, such as feedback control of the motors.

*Activ* MEDIA ROBOTICS is developing Pioneer 2 standalone mode support kits and materials. Stay tuned to the pioneer-users newsgroups for availability.
Updating & Reconfiguring P2OS

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

Where to Get P2OS Software

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

http://robots.activmedia.com/p2

Installing the P2OS Utilities

The P2OS utilities come with your robot or can be downloaded from the support website as a separate package. Install the version that matches your client computer’s environment. For example, use p2osV_r.tgz for RedHat Linux or p2osV_r.EXE with Microsoft Windows 32-bit systems (“V” is the version number and “r” is the revision number; p2os1_c, for example. We distribute these as compressed archives containing all the programs and accessory files you need to perform the PSOS upgrade and to set your Pioneer’s configuration parameters.

The p2osV_r.EXE distribution is a self-extracting, self-installing package: Simply follow the on-screen directions. For the Linux/UNIX versions, uncompress and untar them using the appropriate system utilities. For example, with the Linux version, the proper command is:

% tar –zxvf p2os1_c.tgz

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

Updating P2OS

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

Step 1. Serial Connection from Computer to Pioneer

Connect Pioneer 2 to your host computer through their respective serial ports. We recommend a direct tether from the computer to the 9-pin serial port on top of the Console, because it is the most reliable path for communication. In fact, if you have a Fast-Track Vision System installed, you must use a direct tether to that top Console serial port; neither a tether to the serial port on the back panel nor the radios will work.

If you have, but do not use, the radio modems for serial communications, make sure to switch their power off; if you do not, they will interfere with the direct connection.

Step 2: Enable FLASH

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

Step 3: Put Pioneer into Boot Mode

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

The robot should not reset; if it does, you probably didn’t hold the MOTORS button down long enough. Try it again. And you’ll notice that the ”heartbeat” asterisk stops blinking in the LCD.
Step 4: Run `p2osdl`

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

```
% p2osdl p2os1_c.hex <comm-port>
```

The pathname for the `p2os.hex` image file is required and may be a direct or relative path. The file `p2os1_c.hex`, for example, is the current P2OS version 1.c image in the `p2os/` folder.

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

```
% p2osdl p2os1_c.hex /dev/ttyS3
```

For 32-bit Windows systems, you must pass the required `p2osV_r.hex` file to `p2osdl.exe`, so you cannot simply double-click on the program icon from the Windows desktop. Instead, use your mouse to successfully launch the program by “dragging and dropping” the `.hex` icon (`p2os1_c.hex`, for example) onto the `p2osdl.exe` icon. But this works only if you also plan to use the default COM1 serial port. The universal solution is to execute `p2osdl.exe` and pass it the proper arguments from the MS-DOS Prompt program, which normally resides in the Programs section of the Start menu.

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

Download Troubleshooting

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

Configuring P2OS Operating Parameters

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

Limited reconfigurations.

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

Steps 1–3: Preparing Pioneer 2 for Configuration

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

Step 4: Run `p2oscf`

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

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

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

Step 5: Changing the Configuration Parameters

On startup (after power cycle or RESET), P2OS reads a set of operating parameters from its flash PROM and uses these values if and until you override them with explicit P2OS commands. For instance, a default maximum velocity is stored in flash PROM (TransVelMax) which value is used by P2OS when receiving a
Colbert move command or other “goto position” client command (robot accelerates to that TransVelMax and runs at that velocity until it nears and then reaches its destination).

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

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

**Editing P2OS Parameters**

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

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

```
> watchdog 4000
```

or:

```
> watchdog 0xfa0
```

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

**Table 7-2. p2oscf control commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>keyword &lt;value&gt;</code></td>
<td>Alone, keyword displays current, edited value. Add value argument to change current value.</td>
</tr>
<tr>
<td><code>c</code> or <code>constants</code></td>
<td>Display P2OS constant values. User cannot change.</td>
</tr>
<tr>
<td><code>v</code> or <code>variables</code></td>
<td>Display current, edited P2OS operational values; may be different than values currently stored in flash ROM.</td>
</tr>
<tr>
<td><code>r</code> or <code>restore &lt;pathname&gt;</code></td>
<td>Restores edited (p2oscf) variables to values currently stored in flash ROM or from file, if argument included.</td>
</tr>
<tr>
<td><code>save &lt;pathname&gt;</code></td>
<td>Saves current edited values to flash ROM and exits program OR saves current edited values to disk for later reference and continues in editor.</td>
</tr>
<tr>
<td><code>q</code> or <code>quit</code></td>
<td>Exits p2oscf without saving any changes to flash.</td>
</tr>
<tr>
<td><code>?</code> or <code>help</code></td>
<td>Displays p2oscf commands and descriptions.</td>
</tr>
</tbody>
</table>
Table 7-2. Pioneer 2 DX configuration parameters  

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>TYPE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONSTANTS</strong></td>
<td></td>
<td></td>
<td><strong>Cannot be changed using p2oscf</strong></td>
</tr>
<tr>
<td>Type</td>
<td>string</td>
<td>Pioneer</td>
<td>String identifies the robot as a Pioneer type and is included in the SYNC2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>connection return packet along with Subtype and Name.</td>
</tr>
<tr>
<td>Subtype</td>
<td>string</td>
<td>P2DX</td>
<td>Identifies the Pioneer model; P2AT or P2CE for other models.</td>
</tr>
<tr>
<td>Serial</td>
<td>string</td>
<td>factory set</td>
<td>Serial number for the Pioneer 2.</td>
</tr>
<tr>
<td>FourMotors</td>
<td>byte</td>
<td>0</td>
<td>Is ‘1’ for the four-motor Pioneer 2-AT</td>
</tr>
<tr>
<td>RotVelTop</td>
<td>integer</td>
<td>360</td>
<td>Maximum allowable rotational velocity in degrees per second</td>
</tr>
<tr>
<td>TransVelTop</td>
<td>integer</td>
<td>2200</td>
<td>Maximum allowable speed in millimeters per second</td>
</tr>
<tr>
<td>RotAccTop</td>
<td>integer</td>
<td>600</td>
<td>Maximum allowable rotational (de)acceleration in degrees per second</td>
</tr>
<tr>
<td>TransAccTop</td>
<td>integer</td>
<td>4000</td>
<td>Maximum allowable translational (de)acceleration; millimeters per second</td>
</tr>
<tr>
<td>PwmMax</td>
<td>integer</td>
<td>500</td>
<td>Maximum motor pulse period (500=fully on).</td>
</tr>
<tr>
<td>Compass</td>
<td>Integer</td>
<td>0</td>
<td>Compass type: 0=none; 1=V2XG; 2=TCM2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>VARIABLES</strong></th>
<th>Parameters that you may change with p2oscf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>string not_set Unique name you may give</td>
</tr>
<tr>
<td></td>
<td>your Pioneer 2. Besides its ownership</td>
</tr>
<tr>
<td></td>
<td>value, this parameter gets passed to a</td>
</tr>
<tr>
<td></td>
<td>connecting client as the first argument in</td>
</tr>
<tr>
<td></td>
<td>the SYNC2 packet, therefore useful for</td>
</tr>
<tr>
<td></td>
<td>differentiating among multiple Pioneers.</td>
</tr>
<tr>
<td></td>
<td>Maximum of 20 characters; no intervening</td>
</tr>
<tr>
<td></td>
<td>spaces.</td>
</tr>
<tr>
<td>SInfoCycle</td>
<td>byte 0 Server information packet</td>
</tr>
<tr>
<td></td>
<td>communication cycle time: 0=classic 100</td>
</tr>
<tr>
<td></td>
<td>milliseconds, as for PSOS; 1=new 50ms cycle</td>
</tr>
<tr>
<td></td>
<td>time</td>
</tr>
<tr>
<td>HostBaud</td>
<td>byte 0 Baud rate for host (client) serial</td>
</tr>
<tr>
<td></td>
<td>port connection. 0=9600, 1=19200, 2=38400</td>
</tr>
<tr>
<td></td>
<td>bps.</td>
</tr>
<tr>
<td>AuxBaud</td>
<td>byte 2 Baud rate for second (AUX) serial</td>
</tr>
<tr>
<td></td>
<td>port; values as for HostBaud</td>
</tr>
<tr>
<td>HasGripper</td>
<td>integer 0 ‘1’ if P2 Gripper installed into</td>
</tr>
<tr>
<td></td>
<td>User I/O; ‘2’ if installed in General I/O;</td>
</tr>
<tr>
<td></td>
<td>otherwise 0</td>
</tr>
<tr>
<td>FrontSonar</td>
<td>integer 1 ‘1’ if front sonar array</td>
</tr>
<tr>
<td></td>
<td>installed; otherwise 0</td>
</tr>
<tr>
<td>RearSonar</td>
<td>byte 0 ‘1’ if rear sonar array installed;</td>
</tr>
<tr>
<td></td>
<td>otherwise 0</td>
</tr>
<tr>
<td>LowBattery</td>
<td>integer 110 In 1/10 volts; microcontroller</td>
</tr>
<tr>
<td></td>
<td>alarm activated when battery charge falls</td>
</tr>
<tr>
<td></td>
<td>below this value.</td>
</tr>
<tr>
<td>RevCount</td>
<td>integer 20000 The number of encoder ticks</td>
</tr>
<tr>
<td></td>
<td>for a 360 degree revolution of the robot.</td>
</tr>
<tr>
<td></td>
<td>Reset this parameter to a number that best</td>
</tr>
<tr>
<td></td>
<td>reflects the characteristics of your robot</td>
</tr>
<tr>
<td></td>
<td>in a particular environment. See revcountcal</td>
</tr>
<tr>
<td>WatchDog</td>
<td>integer 2000 Milliseconds time before robot</td>
</tr>
<tr>
<td></td>
<td>automatically stops if it has not received</td>
</tr>
<tr>
<td></td>
<td>a command from a client. Restarts on</td>
</tr>
<tr>
<td></td>
<td>restoration of connection with client.</td>
</tr>
<tr>
<td>P2Mpacs</td>
<td>byte 0 ‘1’ enables new, extended P2OS server</td>
</tr>
<tr>
<td></td>
<td>information packet.</td>
</tr>
<tr>
<td>StallVal</td>
<td>integer 200 Maximum PWM before stall; either</td>
</tr>
<tr>
<td></td>
<td>or both motors. If &gt; PwmMax, never stalls.</td>
</tr>
<tr>
<td>StallCount</td>
<td>integer 100 Milliseconds after a stall for</td>
</tr>
<tr>
<td></td>
<td>recovery. Motors not engaged during this</td>
</tr>
<tr>
<td></td>
<td>time.</td>
</tr>
<tr>
<td>CompX</td>
<td>integer 0 Compass calibration X-offset (See</td>
</tr>
<tr>
<td></td>
<td>compasscal utility)</td>
</tr>
<tr>
<td>CompY</td>
<td>integer 0 Compass calibration Y-offset (See</td>
</tr>
<tr>
<td></td>
<td>compasscal utility)</td>
</tr>
<tr>
<td>RotVelMax</td>
<td>integer 200 Maximum velocity for completion</td>
</tr>
<tr>
<td></td>
<td>of a Colbert or similar rotation.</td>
</tr>
<tr>
<td>TransVelMax</td>
<td>integer 300 Maximum velocity for completion</td>
</tr>
<tr>
<td></td>
<td>of a Colbert or similar translation.</td>
</tr>
<tr>
<td>RotAcc</td>
<td>integer 50 Rotational acceleration in degrees</td>
</tr>
<tr>
<td></td>
<td>per second</td>
</tr>
<tr>
<td>RotDecel</td>
<td>integer 50 Rotational deceleration in degrees</td>
</tr>
<tr>
<td></td>
<td>per second</td>
</tr>
<tr>
<td>RotKp</td>
<td>integer 30 Proportional PID parameter for</td>
</tr>
<tr>
<td></td>
<td>responsiveness of the drive system. Lower</td>
</tr>
<tr>
<td></td>
<td>values make a slower, less-responsive</td>
</tr>
<tr>
<td></td>
<td>system; higher values make the robot</td>
</tr>
<tr>
<td></td>
<td>“zippier”, but can lead to overshoot and</td>
</tr>
<tr>
<td></td>
<td>oscillation.</td>
</tr>
<tr>
<td>RotKv</td>
<td>integer 60 Differential PID dampens</td>
</tr>
<tr>
<td></td>
<td>oscillation and overshoot. Increasing</td>
</tr>
<tr>
<td></td>
<td>values gives better control oscillation</td>
</tr>
<tr>
<td></td>
<td>and overshoot, but they also make the robot’s</td>
</tr>
<tr>
<td></td>
<td>movements more sluggish.</td>
</tr>
<tr>
<td>RotKl</td>
<td>integer 0 Integral PID adjusts residual</td>
</tr>
<tr>
<td></td>
<td>error in turning and velocity. Higher</td>
</tr>
<tr>
<td></td>
<td>values make the robot correct</td>
</tr>
<tr>
<td></td>
<td>increasingly smaller errors between its</td>
</tr>
<tr>
<td></td>
<td>desired and actual angular position and</td>
</tr>
<tr>
<td></td>
<td>speed.</td>
</tr>
<tr>
<td>TransAcc</td>
<td>integer 300 Translational acceleration in</td>
</tr>
<tr>
<td></td>
<td>degrees per second</td>
</tr>
<tr>
<td>TransDecel</td>
<td>integer 300 Translational deceleration in</td>
</tr>
<tr>
<td></td>
<td>degrees per second</td>
</tr>
<tr>
<td>TransKp</td>
<td>integer 30 see RotKp</td>
</tr>
</tbody>
</table>
### Saving and Restoring

The `p2oscf` program lets you save and restore whole configuration sets from stored files. This lets you easily configure Pioneer 2 for the various different environments, as well as maintain a record of your original and test parameters.

To save your current configuration to a disk file, get connected with the Pioneer robot through `p2oscf` as described earlier. This loads the current operating parameters into the configuration editor. Then simply provide argument to the configuration command consisting of the pathname for the configuration file. For example, to save your current configuration:

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

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

Use the `p2oscf restore` command to not only retrieve the current operating parameters from the robot, but to load a saved parameters files from disk. The `p2os` distributions come with common configuration files for the various Pioneer 2 models, include `p2dx` for the Pioneer 2 DX and `p2at` for the Pioneer 2 AT.

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

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

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

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

### Calibration Tools - revcountcal and compasscal

`P2OS` 1.4 introduces two new `P2OS` calibration tools: `revcountcal` and `compasscal`. Both tools are for helping your robot and its Vector 2X compass accessory adapt to your unique operating environment.

#### Compasscal

*Compasscal for Vector 2X compass only. Not for TCM2 Module.*

The revcount calibration tool helps you redefine the differential encoder-count configuration parameter that describes a full 360-degree turn of your Pioneer 2 robot (See `revcount` in Table 7-2). The default revcount works fine for smooth, hard surfaces. But the value can change dramatically for other operating environments, particularly for the Pioneer 2 AT.

The compass tool lets owners calibrate their Pioneer 2 integrated electronic compass for various environments and positions on the robot.
Saphira required.
Compasscal and Revcountcal only work with licensed and properly installed Saphira.

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

A Saphira license comes with your Pioneer 2 Mobile Robot, and you may retrieve new licensed versions from the http://robots.activmedia.com support website. See Chapter 3, "Quick Start", for additional detail.

Briefly, the programs operate in two stages. First, they connect with your Pioneer 2 Mobile Robot via a Saphira-based client and retrieve its current operating parameters via the ENCODERpac and CONFIGpac extended SIPs that are newly available in P2OS 1.4 (see previous chapter). Following program prompts, you change the orientation of the device—rotate the compass and/or the robot—and the program calculates a new revcount value or X and Y offsets for the electronic compass. The process may be repeated indefinitely, each time calculating fresh configuration values, but not permanently changing them on the robot.

In a second stage, the calibration utilities let you save their newly calculated parameters to your Pioneer 2's FLASH ROM, so that the new value gets used by the robot after each startup. You may choose not to change the FLASH ROM values. Rather, you may record the new configuration values and set/reset them using the p2oscf tool described earlier in this Chapter.

To use the tools, first install either compasscal(.exe) or revcountcal(.exe) into the bin directory of your Saphira distribution. Connect your Pioneer 2's HOST serial port to the computer, by tether if you plan to save the calibration parameter to FLASH ROM. Turn on the robot's Main Power.

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

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

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

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

Your Pioneer 2 is built to last a lifetime and requires little maintenance.

Drive Lubrication

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

Pioneer Batteries

Lead-acid batteries like those in Pioneer 2 last longest when kept fully charged. In fact, severe discharge is harmful to the battery, so be careful not to operate the robot if the battery voltage falls below 11 VDC.

Changing Batteries (DX and AT)

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

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

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

Hot-Swapping the Batteries (DX and AT)

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

Charging the Battery

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

It takes fewer than 12 hours—often just a few hours, depending on the level of discharge—to fully charge a Pioneer battery using the accompanying charger (roughly, three hours per volt). Although you may operate the robot while recharging, it restricts the robot’s mobility. Just don’t leave the charger connected for much longer than 24 hours.

Alternative Battery Chargers

The center post of the charger socket on the Pioneer 2 is the positive (+) side of the battery; the case is the negative (-) side. If you choose to use an alternative battery charger for Pioneer, be sure to connect positive to positive and negative to negative from charger to Pioneer.

An alternative AC to DC converter/battery charger for Pioneer should sustain at least 0.75A at 13.75 to 14 VDC per battery. It also should be voltage- and current-limited so that it cannot overcharge the batteries.
Getting Inside

We normally discourage you from opening up your robot. However, on occasion, you may need to get inside your Pioneer 2 Mobile Robot. For instance, you need to get inside to access the User Power connections on the Motor-Power board to attach your custom electronics. Or you may need to get at your onboard computer.

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

We describe here how to remove your robot’s Nose to get at the onboard computer. And we describe what screws to remove to access the contents of the Body of your Pioneer 2 DX, CE, or AT robot.

Onboard Computer

The Pioneer 2 onboard computer sits just behind the robot’s Nose. Its hard-drive, fan, and system speaker are attached to the Nose. Accordingly, remove the robot’s Nose to access the computer to change or add a part or check connections.

For all models, the Nose is attached to the robot with two screws: one directly underneath the front sonar bay and one on the bottom of the robot just behind the seam between the robot’s Body and the Nose. Remove both screws with the hex wrench supplied with your robot. If you have either the P2 Gripper or front Bump Ring accessories, remove it first.

BE CAREFUL: Delicate wiring inside.

Four pins along its back edges guide the Nose onto the front Body of the robot. So, to remove it once the Nose screws are out, pry the Nose out and away from the Body. Careful: The 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.

Top Plate

You may access the contents of your Robot, such as the aforementioned User Power connector on the Motor-Power board, by removing select screws that fasten the Top Plate to the Body. See Figures 8-1 and 8-2 for screw locations on the DX/CE and AT models, respectively. In all cases, remove the screws completely, then carefully lift the Top Plate up off the Body just enough to access internal cables. Detach only those cables, such as the rear sonar cable, that interfere with rotating the Top Plate so you can access the portion you need.

For the Pioneer 2 DX and CE models, use the hex wrenches that came with your robot to remove the six 3mm flat-head screws that attach the top plate of the Pioneer 2 to the sides of its Body. They are the three rearmost screws on each side. Do not remove the screws that attach the microcontroller board and the front sonar ring to the top plate (Figure 2-5). Also remove the eight 4mm screws from the two wheel struts and remove the struts from the wheel axles.

Figure 8-1. Pioneer 2 DX/CE Top Plate retaining screws

For the Pioneer 2 AT model, remove the sixteen 3mm flat-head screws that attach the Top Plate to the Body. All of the screws are to the rear of the robot, along the sides and around the Accessory door. Do not remove the screws at the front of the robot or around the Console.
Figure 8-2. Pioneer 2 AT Top Plate retaining screws

*Factory Repairs*

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

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

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

We will try to resolve the problem through communication. If the robot must be returned to the factory for repair, obtain a shipping and repair authorization code and shipping details from us first.

We are not responsible for shipping damage or loss.
Appendix A
C166 Controller Ports & Connections

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

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

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

Figure A-1. Common IDC connector pinouts

Figure A-2. Pioneer 2 microcontroller ports and connectors

**Console Serial Port**

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

Table A-1 lists the cable connections for various host computers to this external serial port. Most direct serial connections use a simple, straight-through cable, such as the one for a common PC. The communication lines are shared with those on the internal P2OS serial port (see below).
The Pioneer 2 operates at the following configuration: 9,600, 19,200 or 38,400 bits per second data rate; eight data bits; one stop bit; no parity or hardware handshaking (DTR).

**Internal Serial Connectors**

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

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

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Connection</th>
<th>Pin #</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gnd</td>
<td>2</td>
<td>P3_12</td>
</tr>
<tr>
<td>3</td>
<td>TxD1</td>
<td>4</td>
<td>12 VDC (switched)</td>
</tr>
<tr>
<td>5</td>
<td>RxD1</td>
<td>6</td>
<td>Gnd</td>
</tr>
<tr>
<td>7</td>
<td>P3_15</td>
<td>8</td>
<td>P3_14</td>
</tr>
<tr>
<td>9</td>
<td>Gnd</td>
<td>10</td>
<td>5 VDC (switched)</td>
</tr>
</tbody>
</table>

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

**User I/O Expansion Port**

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

- 8 general-purpose digital input and output ports (ID/OD0-7)
- 5 analog-to-digital input ports (AD1-5)
- 4 Pulse-width-modulation ports (PWM1-4)
- 1 signal ground (Gnd)
- 1 Vcc (+5 VDC)
- 1 Vpp (+12 VDC)

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

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

In addition, four of the A/D ports and the PWM ports are shared with digital input and output lines, respectively.

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

The General I/O Bus

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

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

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Label</th>
<th>Pin #</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P3_0</td>
<td>2</td>
<td>D0</td>
</tr>
<tr>
<td>3</td>
<td>P3_1</td>
<td>4</td>
<td>D1</td>
</tr>
<tr>
<td>5</td>
<td>P3_2</td>
<td>6</td>
<td>D2</td>
</tr>
<tr>
<td>7</td>
<td>P3_3</td>
<td>8</td>
<td>D3</td>
</tr>
<tr>
<td>9</td>
<td>P3_4</td>
<td>10</td>
<td>D4</td>
</tr>
<tr>
<td>11</td>
<td>P3_5</td>
<td>12</td>
<td>D5</td>
</tr>
<tr>
<td>13</td>
<td>P3_6</td>
<td>14</td>
<td>D6</td>
</tr>
<tr>
<td>15</td>
<td>P3_7</td>
<td>16</td>
<td>D7</td>
</tr>
<tr>
<td>17</td>
<td>WR</td>
<td>18</td>
<td>CS2</td>
</tr>
<tr>
<td>19</td>
<td>Vpp</td>
<td>20</td>
<td>CS3 (Bumpers)</td>
</tr>
<tr>
<td>21</td>
<td>P2_12</td>
<td>22</td>
<td>CS4</td>
</tr>
<tr>
<td>23</td>
<td>P2_13</td>
<td>24</td>
<td>CS5</td>
</tr>
<tr>
<td>25</td>
<td>P2_14</td>
<td>26</td>
<td>A0</td>
</tr>
<tr>
<td>27</td>
<td>P2_15</td>
<td>28</td>
<td>A1</td>
</tr>
<tr>
<td>29</td>
<td>P5_9</td>
<td>30</td>
<td>Vcc</td>
</tr>
<tr>
<td>31</td>
<td>Gnd</td>
<td>32</td>
<td>RD#</td>
</tr>
<tr>
<td>33</td>
<td>Gnd</td>
<td>34</td>
<td>Vcc</td>
</tr>
</tbody>
</table>
Appendix B
Motor-Power Board Connectors

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

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

Figure B-1. Pioneer 2 Motor-Power Board

User Power Connections

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

Table B-1. User Power (J1)

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Connection</th>
<th>Pin #</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vcc</td>
<td>7</td>
<td>Vcc</td>
</tr>
<tr>
<td>2</td>
<td>Gnd</td>
<td>8</td>
<td>Gnd</td>
</tr>
<tr>
<td>3</td>
<td>Vpp</td>
<td>9</td>
<td>Vpp</td>
</tr>
<tr>
<td>4</td>
<td>Vcc</td>
<td>10</td>
<td>Vcc</td>
</tr>
<tr>
<td>5</td>
<td>Gnd</td>
<td>11</td>
<td>Gnd</td>
</tr>
<tr>
<td>6</td>
<td>Vpp</td>
<td>12</td>
<td>Vpp</td>
</tr>
</tbody>
</table>
**Onboard Computer Option**

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

**Power Switch (J7) and Delayed Shutdown Logic**

A simple SPDT switch controls logical power to the computer-power section of the Motor-Power Board. When on, the switch provides Vpp to the input of a voltage comparator which output controls the computer's DC-DC voltage converter. Designed to provide for "crashless" shutdown of the computer in the event of a power brownout or for unattended shutdowns, if Vpp drops below 10 VDC, the comparator automatically initiates a two-minute delay (LMC555 timer-mediated) before computer power (not system-wide) shutdown.

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

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

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

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vcc</td>
</tr>
<tr>
<td>2</td>
<td>Gnd</td>
</tr>
<tr>
<td>3</td>
<td>Vpp</td>
</tr>
</tbody>
</table>

**Power-State Logic**

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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCD</td>
<td>20K pull-up pins 4,6</td>
<td>nc</td>
<td>nc</td>
<td>pin 6</td>
<td>nc</td>
<td>pin 4</td>
<td>nc</td>
<td>Gnd</td>
</tr>
</tbody>
</table>

**Computer Power**

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

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vpp</td>
</tr>
<tr>
<td>2</td>
<td>Gnd</td>
</tr>
<tr>
<td>3</td>
<td>Gnd</td>
</tr>
<tr>
<td>4</td>
<td>5VDC @ 7A</td>
</tr>
</tbody>
</table>
Appendix C
Saphira Parameter Files

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

**P2DX.p**

```plaintext

;;;;
;;;; Saphira parameters describe the Pioneer 2 DX
;;;;
Class Pioneer
Subclass p2dx
;;;;
AngleConvFactor 0.001534 ; radians per encoder count diff (2PI/1024)
DistConvFactor 0.826 ; 5in*PI / 5000 (mm/count)
VelConvFactor 1.0 ; mm/sec per encoder count per 1/50 sec
RobotRadius 250.0 ; radius in mm
RobotDiagonal 120.0 ; half-height to diagonal of octagon
Holonomic 1 ; turns in own radius

;;;; These are for sixteen sonars: six each front and rear, two each side
;;;;
;;;; Sonar parameters
;;;; SonarNum N is number of sonar
;;;; SonarUnit I X Y TH is unit I (0 to N-1) description
;;;; X, Y are position of sonar in mm, TH is bearing in degrees
;;;;
RangeConvFactor 0.268 ; sonar range mm per 800 nsec tick
;;;;
SonarNum 16
;;;; # x y th
;;;;;;;;
SonarUnit 0 115 130 90
SonarUnit 1 155 115 50
SonarUnit 2 190 80 30
SonarUnit 3 210 25 10
SonarUnit 4 210 -25 -10
SonarUnit 5 190 -80 -30
SonarUnit 6 155 -115 -50
SonarUnit 7 115 -130 -90
SonarUnit 8 -115 130 -90
SonarUnit 9 -155 115 -130
SonarUnit 10 -190 80 -150
SonarUnit 11 -210 25 -170
SonarUnit 12 -210 -25 -170
SonarUnit 13 -190 -80 150
SonarUnit 14 -155 -115 130
SonarUnit 15 -115 -130 90

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


P2AT.p

;;; Parameters for the Pioneer 2 AT Mobile Robot
;;;

AngleConvFactor  0.001534 ; radians per angular unit (2PI/4096)
DistConvFactor    1.303   ; mm returned by P2
VelConvFactor      1.0     ; mm/sec returned by P2
RobotRadius       500.0    ; radius in mm
RobotDiagonal     120.0    ; half-height to diagonal of octagon
Holonomic          1        ; turns in own radius
MaxRVelocity      300.0    ; degrees per second
MaxVelocity       1200.0   ; mm per second
RangeConvFactor    0.268   ; sonar range returned in mm

;;; Robot class, subclass
;;;
Class   Pioneer
Subclass p2at
SonarNum 16 ; 16 total sonars

;;; These are for the eight front sonars: six front, two sides
;;;

SonarUnit  #   x   y   th
-----------
0         0  145  130  90
1         1  185  115  50
2         2  220  80   30
3         3  240  25   10
4         4  240 -25  -10
5         5  220 -80  -30
6         6  185 -115  -50
7         7  145 -130  -90

;;; These are for the eight rear sonars: six back, two sides
;;;

SonarUnit  #   x   y   th
-----------
8         8 -145 -130  -90
9         9 -185 -115 -130
10        10 -220 -80  -150
11        11 -240 -25  -170
12        12 -240  25  170
13        13 -220  80  150
14        14 -185 115  130
15        15 -145 130  90

;;; Number of readings to keep in circular buffers
FrontBuffer  20
SideBuffer   40
Appendix D
Joystick Connector

Use a 20-wire IDC-terminated ribbon cable to attach the 15-pin joystick connector to the User I/O port on the Pioneer 2 microcontroller. Both require P2OS version 1.c or later.

Figure D-1. Standard joyport for manual drive of a Pioneer 2 Mobile Robot

Figure D-2. Joyport with signal switch for sharing User I/O ports with other accessories
# Appendix E

## Specifications

### Physical Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Pioneer 2 DX/CE</th>
<th>Pioneer 2-AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>44cm</td>
<td>50cm</td>
</tr>
<tr>
<td>Width</td>
<td>33cm</td>
<td>49cm</td>
</tr>
<tr>
<td>Height (body)</td>
<td>22cm</td>
<td>24cm</td>
</tr>
<tr>
<td>Body clearance</td>
<td>5.1cm</td>
<td>5.5cm</td>
</tr>
<tr>
<td>Weight</td>
<td>9kg</td>
<td>14kg</td>
</tr>
<tr>
<td>Payload</td>
<td>20kg</td>
<td>40kg</td>
</tr>
</tbody>
</table>

### Construction

<table>
<thead>
<tr>
<th></th>
<th>Pioneer 2 DX/CE</th>
<th>Pioneer 2-AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>1.6mm CNC fabricated aluminum</td>
<td>Same</td>
</tr>
<tr>
<td>Console &amp; main deck</td>
<td>2.4mm CNC fabricated aluminum</td>
<td>Same</td>
</tr>
<tr>
<td>Assembly</td>
<td>Allen hex screws (metric)</td>
<td>Same</td>
</tr>
</tbody>
</table>

### Power

<table>
<thead>
<tr>
<th></th>
<th>Pioneer 2 DX/CE</th>
<th>Pioneer 2-AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>3x12V lead-acid (DX only)</td>
<td>Same</td>
</tr>
<tr>
<td>Charge</td>
<td>252 watt-hr; hot-swappable (DX)</td>
<td>Same</td>
</tr>
<tr>
<td>Run time</td>
<td>8–10 hrs (3 hrs CE)</td>
<td>4-6 hrs</td>
</tr>
<tr>
<td>Recharge time std charger</td>
<td>12 hrs (8 hrs CE)</td>
<td>Same</td>
</tr>
<tr>
<td>Recharge time hs charger</td>
<td>2.4 hrs (1 hrs CE)</td>
<td>Same</td>
</tr>
</tbody>
</table>

### Mobility

<table>
<thead>
<tr>
<th></th>
<th>Pioneer 2 DX/CE</th>
<th>Pioneer 2-AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive wheels</td>
<td>2 solid rubber, with caster</td>
<td>4 pneumatic</td>
</tr>
<tr>
<td>Wheel diameter</td>
<td>16.5cm</td>
<td>22cm</td>
</tr>
<tr>
<td>Wheel width</td>
<td>3.7cm</td>
<td>7.5cm</td>
</tr>
<tr>
<td>Steering</td>
<td>Differential</td>
<td>Skid</td>
</tr>
<tr>
<td>Gear ratio</td>
<td>19.7:1</td>
<td>66:1</td>
</tr>
<tr>
<td>Swing radius</td>
<td>32cm</td>
<td>40cm</td>
</tr>
<tr>
<td>Turn radius</td>
<td>0cm</td>
<td>0cm</td>
</tr>
<tr>
<td>Translate speed max</td>
<td>1.6 m/sec</td>
<td>0.7 m/sec</td>
</tr>
<tr>
<td>Rotational speed max</td>
<td>300 degrees/sec</td>
<td>150 degrees/sec</td>
</tr>
</tbody>
</table>
Appendix E: Specifications

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Traversable step max</th>
<th>2cm</th>
<th>8.9cm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Traversable gap max</td>
<td>8.9cm</td>
<td>12.7cm</td>
</tr>
<tr>
<td></td>
<td>Traversable slope max</td>
<td>25% grade</td>
<td>40% grade</td>
</tr>
<tr>
<td></td>
<td>Traversable terrains</td>
<td>All wheelchair accessible</td>
<td>Unconsolidated surfaces; no carpet!</td>
</tr>
</tbody>
</table>

| Sensors                               | Front Ultrasonic     | 8   | Same |
|                                       | 1 each side          |     | Same |
|                                       | 6 forward @ 20° intervals | Same | |
| Rear Ultrasonic                       | 8                    |     | Same |
|                                       | 1 each side          |     | Same |
|                                       | 6 rearward @ 20° intervals | Same | |
| Position encoders                     | 76 ticks per mm      | 47 per mm | |
|                                       | 9,850 ticks per revolution | 8,520 per rev. | |

| Electronics (basic onboard microcontroller) | Processor       | Siemens 8C166 (20 MHz) | Same |
|                                             | Position inputs  | 4                     | Same |
|                                             | Sonar inputs     | 2x8 (multiplexed)     | Same |
|                                             | Digital I/O      | 16 logic ports        | Same |
|                                             | A/D              | 5 @ 0-5 VDC           | Same |
|                                             | Digital timer output | 8 @ 1µsec resolution | Same |
|                                             | Digital timer inputs | 8 @ 1µsec resolution; | Same |
|                                             | Comm port        | 2 RS-232 serial       | Same |
|                                             | FLASH PROM       | 32 KB; P2OS-encoded software | Same |
|                                             | RAM              | 32 KB; user programmable | Same |
|                                             | Power switches   | 1 main; 2 auxiliary    | Same |

<p>| Controls and Ports                      | Main Power        | Robot power On/Off    | Same |
|                                       | Charge            | System power/battery recharge | Same |
|                                       | LCD display       | Systems status and messages | Same |
|                                       | RESET pushbutton  | Warm reboot/download   | Same |
|                                       | MOTORS pushbutton | Motors/joydrive/download/self-tests | Same |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADIO</td>
<td>Radio modem or Ethernet</td>
<td>Same</td>
</tr>
<tr>
<td>Speaker</td>
<td>Piezo</td>
<td>Same</td>
</tr>
<tr>
<td>Serial comm port</td>
<td>9-pin RS-232</td>
<td>Same</td>
</tr>
<tr>
<td>Joystick Drive</td>
<td>Optional</td>
<td>Standard</td>
</tr>
</tbody>
</table>
Index

A
A/D, 34
ADSEL, 34
Argument types, 30
Assembly, 16
Autoconfiguration, 32
AUX, 36
Ayllu, 5

B
Batteries, 11, 45
Changing, 45
Charge Cube, 13
Charging, 45
Hot-swap, 45
Recharge port, 12
Body, 9
Boot mode, 39
BUMPSTALL, 35

C
Calibration tools
compass, 43
revcount, 43
Checksum, 27
class, 32
Client
Commands, See Client commands
Saphira, 16
Client commands
Argument types, 30
Communication rate, 29
CONFIG, 36
ENCODER, 36
General, 29
GETAUX, 36
Saphira.h, 31
Client-Server, 26
CLOSE, 32
Colbert, 5
Cold Start, 17
Communication packets, 27. See Packets
Communications, 17
Communication rate, 28
compass, 25, 43
compasscal, 43
Components, 8
Basic, 1
Batteries, 11
Body, 9
Console, 9
Deck, 9
Electronics, 11
Microcontroller, 12
Nose, 10
Optional, 1
Pioneer 1, 5
Pioneer 2, 4
Pioneer AT, 5
Power, 12
Sonars, 10
User supplied, 2
Computer, 10
Onboard, 53
Shutdown delay, 53
Computer, 1
CONFIG, 36
CONFIGpac, 36
Configuration, 15
Parameters, 42
Configuration packets, 36
Configuration parameters, 41, 43
Configuration Parameters
Pioneer 1, 42
Console, 9
Contrast, 13
Controller
Specifications, 48
Controls, 8
Sonar Gain, 11
Controls & Ports
Contrast, 13
FLASH, 14
LCD, 13
Main Power, 12
MOTORS, 14
RADIO POWER, 14
Recharge, 12
RESET, 14
SERIAL PORT, 14
STATUS LED, 13
Critical parameters, 41
D
Data types, 27
DCHEAD, 32
Deck, 9
DHEAD, 32
Digin, 34
Digital Input, 34
Digital Output, 35
DIGOUT, 35
Disassembly, 45
Nose, 46
Onboard computer, 46
Top Plate, 46
Drive Lubrication, 45

E
E_STOP, 33
Electronics, 11
Email	pioneer-support, 3
pioneer-users, 2
saphira-users, 2
ENABLE, 32
ENCODER, 36
ENCODERpac, 36
Encoders, 6, 10
Errors, 27
Extended packets, 36

F
Flash, 39
FLASH, 14
Flash ROM, 6
Fuse, 12

G
General I/O, 22, 50
GETAUX, 36

I
I/O, 34
Information packets, 28
Installation
Top Plate, 46

J
Joystick, 22
Joystick, 56
Joystick-drive, 22

K
Kurt Konolige, Dr., 4

L
Liquid-Crystal Display, 13

M
Maintenance, 45
Microcontroller, 6, 12
Modes
Connected, 14
Download, 14
Joystick-drive, 14
Range, 11
Specifications, 8
Stalls, 15
stallval, 15
stallwait, 15
Standalone Mode, 38
Start up, 31
STATUS, 13
subclass, 32
Support
  pioneer-support, 3
SYNC, 31
SYNCO, 31
SYNCl, 31
SYNC2, 31

T
  TCM2, 25
  Timer, 34
  Translation, 32
  Troubleshooting, 20

U
  User I/O, 22
  Digin, 23, 24
  User I/O Ports, 49
    Analog, 24, 50
    Digital, 50
    DIGOUT, 24

PWM, 25
  User power, 52

V
  Vector
    2X, 43
    TCM2, 43
    VEL, 32
    VEL2, 32

W
  Wait State, 17
  watchdog, 15
Warranty & Liabilities

Your Pioneer is fully warrantied against defective parts or assembly for one year after it is shipped to you from the factory. Accessories are warrantied for 90 days. This warranty explicitly does not include damage from shipping or from abuse or inappropriate operation, such as if the robot is allowed to tumble or fall off a ledge, or if it is overloaded with heavy objects.

The developers, marketers, and manufacturers of Pioneer shall bear no liabilities for operation and use of the robot or any accompanying software except that covered by the warranty and period. The developers, marketers, or manufacturers shall not be held responsible for any injury to persons or property involving Pioneer Mobile Robots in any way. They shall bear no responsibilities or liabilities for any operation or application of the robot, or for support of any of those activities. And under no circumstances will the developers, marketers, or manufacturers of Pioneer take responsibility for support of any special or custom modification to Pioneer or its software.
44 Concord Street
Peterborough, NH 03458
(603) 924-9100
(603) 924-2184 fax
http://www.activrobots.com