MOBILEROBOTS INC

Pioneer Arm Manual

with NEW!

AKIN Arm Kinematics Software

Important Safety Instructions

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

Inappropriate Operation

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

- ✓ Dropping the equipment
- ✓ Overloading the Arm above its payload capacity
- ✓ Getting the equipment wet
- ✓ Continuing to run the equipment after hair, yarn, string, or any other items have become wound in one or more of its joints
- ✓ All other forms of inappropriate operation or care

Use MobileRobots authorized parts ONLY;
warranty void otherwise.

Table of Contents

CHAPTER 1 INTRODUCTION 1
THE PIONEER PLATFORM 1
THE PIONEER ARM 1
CONTROL HARDWARE AND SOFTWARE 2
ARM PACKAGE 2
Basic Components (all shipments) 2
User-Supplied Components / System Requirements 3
ADDITIONAL RESOURCES 3
Software 3
Newsgroups 3
Support 4
CHAPTER 2 INSTALLATION & OPERATION 5
MOUNTING THE ARM TO YOUR PIONEER ROBOT 5
POWER TO THE ARM 6
Arm Safely at HOME 6
CONFIGURATION 6
ARM AND ARM KINEMATICS DEMONSTRATIONS 7
P2ArmDemo 7
ArmServer and ArmApplet.jar 7
SAFETY WATCHDOGS 8
Power and Connection-Related Automatic Shutdown 8
Timed Shutdown 9
Gripper Release Timer 9
Warm Reset 9
CHAPTER 3 SOFTWARE AND PROGRAMMING 10
Generic versus Server Programming 10
GENERIC CONTROLS 10
Generic Command Syntax 10
Generic Commands 10
Generic Command Communications 11
PIONEER CONTROLLER ARM SERVERS 11
Client Commands 11
INFO and ARMINFOpac 12
POWER 12
STATUS and ARMpac 13
SPEED and POS 13
SPEED Limits 14
HOME and PARK 14
INIT, CHECK, and Warm Reset Protocol 14
ARIA AND ARAKIN ARM SUPPORT 15
ArP2ArmSimple, ArAKinDemo and ArAKinExample 16
CHAPTER 4 ARM FLASH PARAMETERS 17
FIRMWARE TOOLS AND PARAMETERS 17
Installing the FLASH Utilities 17
CONFIGURING PIONEER ARM OPERATING PARAMETERS 17
Step 1. Serial Connection from Computer to Robot 18
Step 2: Enable FLASH (C166 only) 18
Step 3: Put Controller into Maintenance Mode (P2OS and AROS only) 18
Step 4: Run the cf Program 18
Step 5: Changing Arm Configuration Parameters 19
Step 6: Save Your Work 19
EDITING THE ARM PARAMETERS 19
CHAPTER 5 MAINTENANCE & REPAIR 21
FACTORY REPAIRS 21
APPENDIX A 22
WARRANTY & LIABILITIES 23

Chapter 1 Introduction

Figure 1. The Pioneer Arm on a 3-DX

Congratulations on your purchase of MOBILEROBOTS' multi-jointed robotic arm for the Pioneer family of mobile robots.

This manual provides both the general and technical details you need to operate your Pioneer Arm and to begin developing your own control software integrated with other mobile robot activities.

The Pioneer Platform

Pioneers are small, intelligent mobile robots. The basic Pioneer platform contains all of the components of an intelligent mobile robot for sensing and navigation in a real-world environment, including battery power, drive motors and wheels, position encoders, and range-finding ultrasonic sonar transducers—all managed via either a Siemens C166-, an Hitachi H8S-, or a Renesas SH2-based controller board.

Pioneer DX models are two-wheel, differential-drive mobile robots intended for indoor, albeit less-than-ideal surface operation (wheelchair accessible areas, for example). Performance PeopleBot™, too, is a Pioneer differential-drive robot. Its tall body is for human interaction studies and applications. The Pioneer AT is a four-wheel drive, differential skid-steering version intended for outdoor, all-terrain (AT) operation. Technically, the Pioneer platforms are nearly identical and share accessories, including the Pioneer Arm.

All Pioneer robots come with onboard robotics server software. These operating systems have an open API for client control of the robot's systems and accessories. Users access the onboard servers through an RS-232 serial communication port from their client workstation connected wirelessly, or tethered via a piggyback laptop or integrated onboard PC. See your robot's *Operations Manual* for details.

The Pioneer Arm

The Pioneer Arm is a relatively low-cost robot accessory for use in research and in the classroom. Driven by six open-loop servo motors, the five degrees-of-freedom (5-DOF) Pioneer Arm's end-effector is a gripper whose foam-lined fingers allow for firm grasp and manipulation of objects as large as a soda can and as heavy as 150 grams throughout the arm's envelope of operation.

Joints include:

- ✓ rotating base
- ✓ pivoting shoulder
- ✓ pivoting elbow
- ✓ rotating wrist
- ✓ pivoting gripper mount
- ✓ gripper fingers

All servo-driven joints, except for the gripper fingers, pivot or rotate at least 180 degrees. Mounted to the front of the robot's top plate, the Pioneer Arm has a very wide envelope of operation, reaching 50 centimeters from the center of its rotating base to the tip of its closed fingers. The Pioneer Arm's reach, therefore, lets you pick up objects from the floor in front of your Pioneer robot and place them on its back.

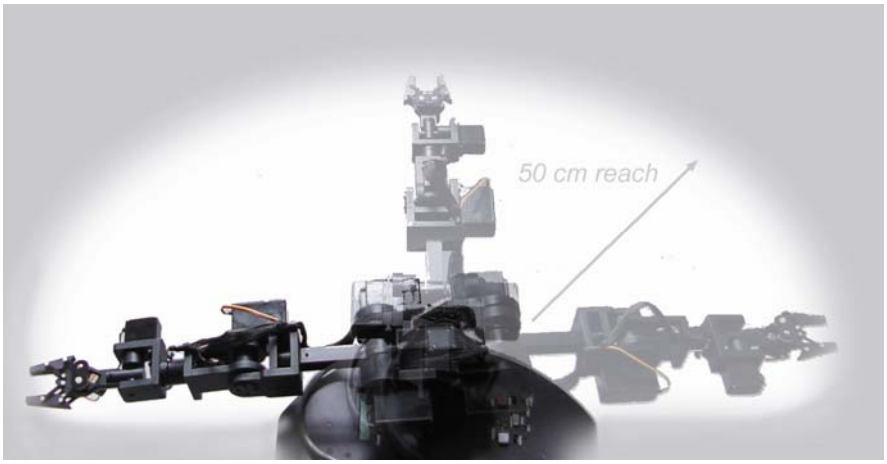


Figure 2. The Pioneer Arm's horizontal reach

Control Hardware and Software

The Pioneer Arm comes with its manufacturer's PIC-based controller and servo-driver hardware and software. These are housed inside the DX on top of the battery box or inside a special accessory box that is mounted at the rear of the AT or Performance PeopleBot robot's top-plate.[1]

The Pioneer Arm's controller is attached to one of your robot controller's AUX serial ports. Its operation is supported by generic ("pass-through") serial commands, as well as dedicated Pioneer Arm servers as part of the controller's embedded system software. Your client software communicates with and controls the Arm through these servers.

Develop integrated robotics client software for the Pioneer platform overall and including the Arm with MOBILEROBOTS' Advanced Robotics Interface for Applications (ARIA). ARIA with ArP2Arm C++-language sources comes with the robot as is available for updates and downloads from the MOBILEROBOTS support website, `http://robots.MobileRobots.com`.

In addition, the Pioneer Arm now comes bundled with ARIA-integrated inverse kinematics software (Arm Kinematics; AKin), as licensed from Fachhochschule Trier University of Applied Sciences. Simply provide the 3-D position for the gripper end-effector and let AKin do the work. ARIA and ArAKin come with several demonstration clients that you may modify and use to create your own mobile-robotic arm software.

Windows® users also get standalone Arm Server software with a Java-based GUI demonstration application (`ArmServer` and `armApplet.jar`) from Fachhochschule Trier.

Arm Package

Your Pioneer Arm is factory installed. Our experienced manufacturing staff put your Pioneer Arm and its support accessories through a "burn in" period and carefully tested them before shipping the products to you. We detach the Arm from the robot and ship them separately for safety.

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

Basic Components (all shipments)

- ✓ This manual
- ✓ Arm calibration sheet
- ✓ One fully assembled Pioneer 5-DOF Arm
- ✓ Arm control electronics, signal and power cabling installed in your robot

[1] Retrofitted DXs get the external control box, too.

- ✓ Mounting hardware and cabling
- ✓ CD-ROM containing controller software for Arm support, Arm demonstration and development software, and documentation

User-Supplied Components / System Requirements

- ✓ Tools for attaching the arm to the robot[2]
- ✓ Client-side PC and robot software

Additional Resources

All new MOBILEROBOTS customers get three additional and valuable resources:

- ✓ A private account on our Internet server for software, updates, and manuals
- ✓ Access to private robotics newsgroups
- ✓ Direct access to the MOBILEROBOTS technical support team

Software

Figure 3. Pioneer Arm's vertical reach

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

`http://robots.MobileRobots.com`

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

Newsgroups

We maintain several email-based newsgroups through which robot owners share ideas, software, and questions about the robot and accessories, such as your new Arm. Visit the support `http://robots.MobileRobots.com` website for more details.

To sign up for pioneer-users, for example, send an e-mail message to the `-requests` automated newsgroup server:

To: **pioneer-users-requests@MobileRobots.com**

[2] The required tools, particularly the hex wrenches, come with the robot.

From: *<your return e-mail address goes here>*
Subject: *<choose one command:>*
help (returns instructions)
lists (returns list of newsgroups)
subscribe
unsubscribe

Our SmartList-based listserver will respond automatically.

After you subscribe, send your email comments, suggestions, and questions intended for the worldwide community of Pioneer users to the users group, NOT to the –requests address. For example:[3]

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

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

Tell us your robot's SERIAL NUMBER.

Support

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

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

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 MOBILEROBOTS technical support team. There a staff member will help you or point you to a place where you can find help.

[3] Notice that the –**requests** part of the email address is left out when sending messages to the newsgroup?

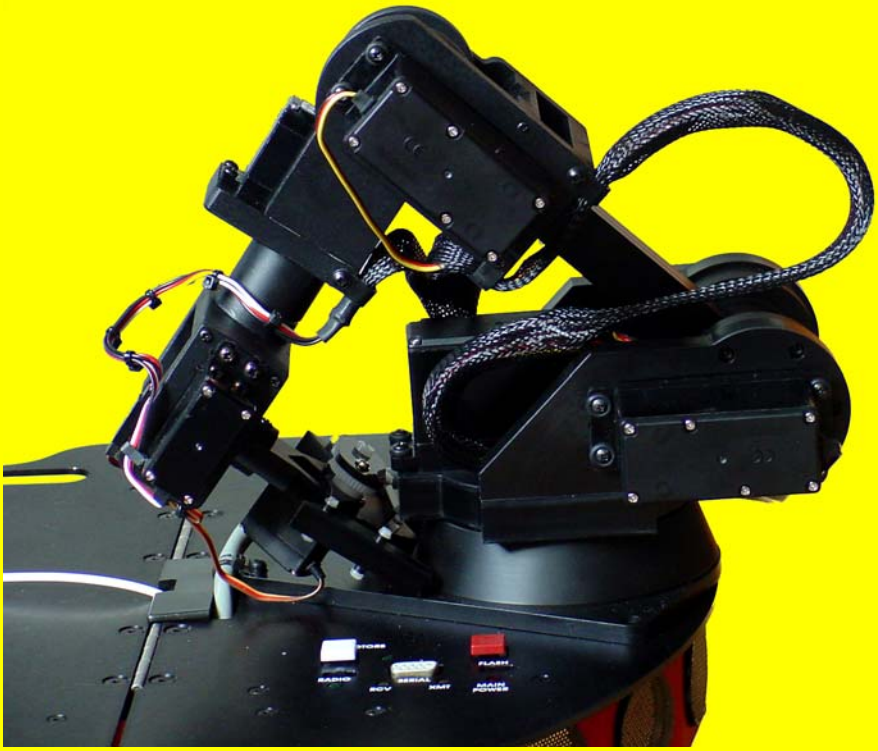
Chapter 2 Installation & Operation

The Pioneer Arm's control electronics and server software are factory installed and configured, but the arm is detached and shipped separately for safety.

Mounting the Arm to Your Pioneer Robot

After carefully unpacking and inspecting the Pioneer Arm, robot, and accessories, manually fold the Pioneer Arm into its HOME position. Then secure the Arm to the prepared top plate at the front of your robot:

CAUTION



Always align your Pioneer Arm into it's HOME position before use.

1. Put the Pioneer Arm into its HOME position: The tip of the Arm's V-shaped mounting base should point towards the rear of the robot, and the arm should be folded towards the back of its base.

2. Use the hex wrench that came with your robot and one of the mounting screws to secure the arm's base tip to the robot's top plate just in front of the access port. Freely rotate the arm around its base to access the mounting holes.

3. Align the two other mounting holes on each side and at the front of the Pioneer Arm's base plate with their respective holes on the robot's top plate and secure with the provided mounting screws.

4. Mate with the signal and power connector attached to the arm with the one that comes up through the robot's maintenance port on the top plate. Insert wired harnesses back into the body of the robot.

Power to the Arm

Power for the Pioneer Arm's controller comes from your robot's batteries, as switched by `Main Power`. Power to the Pioneer Arm servo joints is normally OFF and only switched ON from software by toggling a digital output port in the arm controller.

For power conservation and safety reasons, the arm-servo power relay normally is deactivated. Accordingly, all the joints are limp and easily rotated and swiveled manually. When engaged, the servo motors will resist your efforts to move them.

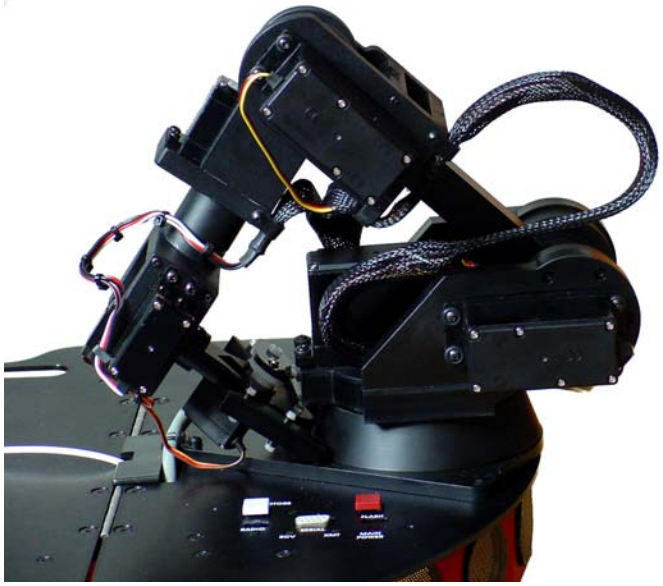


Figure 4. Your Pioneer 2 Arm safely at `HOME`

When the Pioneer Arm servos get power, all joints simultaneously and as quickly as possible jump to their positions last programmed into the arm's controller. If power is removed from the servos, all the joints fall into their mechanically limpest position.

Since the Pioneer Arm's joint positions cannot be known, they are assumed by their generic controller to be in a default center position. When you apply power to the servos, the arm thereby snaps fully extended into a "salute" nowhere near any reasonable limp configuration. This snap-to-salute draws excessive power and can swing the joints into nearby objects. [4]

Arm Safely at HOME

On start up or reset, your robot controller's Arm servers establish communication with the Arm's controller and reset the joints to be in a low-energy, folded `HOME` position. When limp in that same position, no joint is far from its `HOME` position when servo power gets applied. This way, the snap-to-position jolts characteristic of servo motors when first powered up are minimized.

Accordingly, when operating the Pioneer Arm, be sure to manually place it in its `HOME` resting position ***before*** applying power to its servos, and be careful to have it resume its `HOME` position before removing power. Failure to do so may cause damage to the Pioneer Arm or to nearby people, animals, lamps, vases, undergraduates…

For additional safety, the Pioneer Arm normally should ride on the moving robot in its `HOME` resting position with servo power OFF, unless involved in manipulation activities. It only makes good sense to protect the arm from damage and to conserve power; the servos draw precious battery power while holding a position as well as while moving, and the Pioneer Arm can go dangerously limp if you carelessly withdraw power from it.

Configuration

We normally install and configure the Pioneer Arm at the factory. During that process, we enable the Pioneer Arm servers in your robot's controller. And we calibrate each arm joint to arm-unique maximum, minimum, center, and `HOME` positions. All these details are saved in your robot controller's FLASH parameters and recorded on a document that comes with your robot and arm. If you move the Pioneer Arm to another robot or if you change controllers, you'll need to reset that robot's FLASH configuration.

Please consult a subsequent chapter in this manual concerning editing the FLASH parameters, as needed. Note here that with C166-based controller with P2OS, you need to set in FLASH the

[4] Some may argue that this presents a fine opportunity to perfect your robot's "dope slap" maneuver made famous by Click and Clack, the Tappet Brothers, on their infamous National Public Radio show, Car Talk©.

ArmNumJoints parameter value to 6 and the auxbaud value to 0 (9600 baud). With AROS, set the ArmPort parameter to 1 if the Pioneer Arm controller is connected via to the controller's AUX1 serial port, or 2 if connected to AUX2.

Similarly, with the latest ARCOS-enabled Pioneer robots, set the ArmPort parameter to 1 if the Pioneer Arm controller is connected via to the controller's AUX1 serial port, 2 if connected to AUX2, or 3 if it's attached to serial AUX3. In all cases, you need not set the related port's baud rate since the Arm servers automatically set the communication rate to 9600 baud.

Also note that an LED attached to the Pioneer Arm controller only flashes on and off when the arm is properly attached, the Arm servers are enabled, and there are proper communications between the Pioneer controller and the arm controller.

Arm and Arm Kinematics Demonstrations

Several Pioneer Arm demonstration programs and source files come on the CD-ROM that accompanies the system. Use the P2ArmDemo executable with the keyboard arrows or a joystick to select and move individual joints. Alternatively, steer the Arm kinematically with ArmServer and its ArmApplet.jar demonstration GUI (Windows only). Finally, see how to develop robot-integrated Arm software with the ArP2ArmSimple and ArAKinDemo sources in ARIA.

Start up your robot. Its Pioneer Arm automatically gets power, too, but the joint servos should be OFF and the arm limp in its HOME position. If not, manually move the joints until the arm is in that HOME position.

P2ArmDemo

On startup, the P2ArmDemo program automatically makes a client connection with your robot's controller through the COM1 (/dev/ttyS0 with Linux) serial port.[5] It then graphically displays the Arm on your PC screen along with a set of control buttons and operating values.

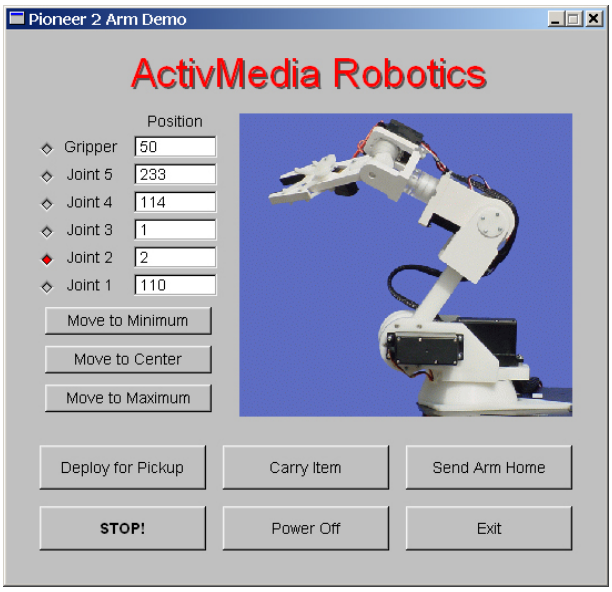


Figure 5. The P2ArmDemo lets you operate the Pioneer Arm quickly and easily.

Red dots indicate the current control joint. Change joints by clicking the joystick fire button, with the up or down arrow key, or by clicking the mouse. Move the selected joint's position with the right or left arrow key or with the joystick (active when fire button pressed). Or click a control button with the mouse to automatically center or move a joint to its center, home, maximum, or minimum positions. Press the STOP! button or the space bar at any time to stop the Arm from moving.

Deploy for Pickup sends the Arm around and to the floor, ready to pick up some item you may place in its gripper. Carry Item sends the Arm up and around to place the item it has in its gripper on the back of the robot.

The Arm servos automatically power up when you select to operate a joint. The Power Off button automatically homes the Arm to its power off position.

ArmServer and ArmApplet.jar

[5] See your *Pioneer 2 or Pioneer 3 Operations Manual* for details on how to attach your PC to the robot and make a client-server connection. If you get a start up error, please make sure the Pioneer Arm is properly attached and configured.

Forward and inverse kinematics libraries now come with the Pioneer Arm, as licensed from Fachhochschule Trier University of Applied Sciences. `ArmServer.exe` is a Windows-only application that lets you integrate the AKIN arm-kinematics software with your own, arm-specific applications. `ArmApplet.jar` is a demonstration GUI for `ArmServer.exe`.

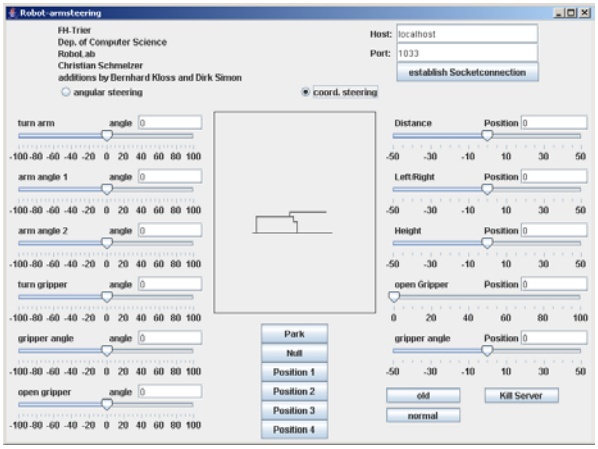


Figure 6. Connect ArmApplet.jar GUI to ArmServer to steer the arm with AKIN-based kinematics.

Like `P2ArmDemo`, `ArmServer.exe` makes a serial connection with your robot controller, which in turn makes a connection with the Pioneer Arm. If you connect a PC with the robot controller through COM1, simply double-click `ArmServer.exe` to launch it. Otherwise, use a command window and append the '`-rp <serial port>`' startup argument to the command line, where <serial port> is the alternative communication port; `COM3`, for example.

Now start `ArmApplet.jar`, either on the same or on a PC elsewhere on the network. Enter the network address of `ArmServer`'s Host: PC and click `establish SocketConnection`.[6]

Operate the arm by angles or by explicit x,y,z coordinates from the respective sliders. The arm animation in the center of `ArmApplet.jar` reflects the motions. Click and hold the mouse for a few seconds on any of the `Park`, `Null`, or `Position 1` through `4` buttons to capture the current arm coordinates or angles. Click the button again later to have the arm achieve that configured position.

Safety Watchdogs

It is important to well manage the Pioneer Arm to minimize its power consumption and to protect it and its surroundings from inadvertent damage. The robot controller's Arm servers support important features that monitor Arm activity and automatically, if enabled, act to protect your Pioneer Arm. These watchdogs include a client connection-related automatic shutdown server, a release timer for the Arm gripper, and a warm-reset server.

Power and Connection-Related Automatic Shutdown

Because it is attached to your Pioneer robot's microcontroller, your client software may command and control the Pioneer Arm only while it has an established client-server connection. Moreover, you must explicitly software-command the Arm to power its servos: Its default state is in the `HOME` position with servo power OFF.

With servo power ON and in a position other than `HOME`, the Pioneer Arm automatically returns to its `HOME` position when your client software tells the arm servers to remove power from the servos.[7] Otherwise, the arm might flop into an awkward and potentially damaging position when you remove power to its positioning servos.

Similarly, the Arm servers automatically send the Pioneer Arm to its `HOME` position and remove power from its servos immediately following severance of a client-server connection. See the `CLOSE` server descriptions in your *Operations Manual* for more details about client-server connections.

The automated shutdown service may be disabled with a special client command and FLASH parameter setting. See following chapters for details.

[6] The default localhost name typically works when running ArmServer and armApplet.jar on the same PC. Otherwise, enter another hostname or IP address for ArmServer's Host: PC on the nework.

[7] You may override some ARCOS/AROS/P2OS Arm servers with direct, generic Arm commands.

Timed Shutdown

Another Pioneer Arm watchdog tests the amount of time that may lapse while the robot is connected with a client PC and the Arm is in service but has not moved. If that time is excessive, the AutoParkTimer server automatically sends the Pioneer Arm to its HOME position and removes power to its servos. This guards against forgetful neglect.

AutoParkTimer is set from a parameter stored in your robot controller's FLASH and altered through the FLASH configuration utility. That timer default also can be disabled, lengthened, or shortened with an arm-related client command. It gets reset every time your client software sends an arm-control command through the arm's servers, too.

Gripper Release Timer

The Pioneer Arm's gripper servo can be subjected to overheating and stress while grasping an object and holding it for excessive periods of time. Accordingly, like the automatic shutdown timer, the arm servers support a special watchdog that automatically opens the gripper to its default HOME position after it has been closed for a FLASH-parameter set (GripperParkTimer) period of time. A special client command lets you reset the gripper timer, thereby overriding its immediate effects, as needed[8], although we recommend that you don't grasp objects for longer than five minutes at a time, with healthy rest periods in between.

CAREFUL!

Unless carefully managed by your software, the Pioneer Arm gripper timer may act to drop an object prematurely.

Like AutoParkTimer, the gripper's watchdog timer GripperParkTimer is a parameter stored in your robot controller's FLASH and modified with the FLASH configuration utility. The timer defaults to five minutes and can be disabled. The timer gets reset every time your software opens the gripper to or near its HOME position.

Warm Reset

Another Pioneer Arm-server watchdog keeps track of the connection between your robot's controller and the arm's controller. This is most important for when the connection between the arm's controller and the arm servers is not enabled while establishing a client-server connection, or is lost during a client-server session. In these rare circumstances, the servers assume that the arm is disabled and not in its HOME position, and thereby goes into an arm-disabled state that requires a special protocol for recovery. The protocol for recovery from a communications failure is either a cold-reset of the robot's controller or through the Warm Reset protocol of commands from a connected client. Of course, take the latter approach if your client software already is connected and controlling the robot. In all cases, be sure to put the Pioneer Arm into its HOME position ***before*** resetting the system.

[8] Holding some nearly closed position, but not grasping an object, for example.

Chapter 3 Software and Programming

The Pioneer Arm's controller is attached to and controlled through an AUX serial port on your robot's controller. The robot controller, on the other hand, is connected with your robotics programs through its HOST serial port. This means that your command and control client programs communicate with your Pioneer Arm through the Pioneer controller, not directly with the Pioneer Arm's controller.

Generic versus Server Programming

We connect the Pioneer Arm through the robot's controller for several reasons: It means that you need only a single communication channel for all client-server communications with your robot and its accessories, thereby obviating the need for additional wireless modems or an expensive onboard PC with multiple serial ports. Perhaps more importantly, we've included special servers in your robot controller's embedded software that provide control features and watchdogs not available with the Pioneer Arm's generic control software and not easily performed from a standard PC.

Even so, Pioneer's controller servers also support pass-through servers that transfer serial communications directly to and from the HOST and AUX serial ports, so that you may control your Pioneer Arm with its generic command set, if you prefer.

Generic Controls

The Pioneer Arm's generic controller responds to a simple set of commands that enable/disable power to the servos and position the servo-driven joints. It also lets you query the controller software version number and accessory digital I/O.

However, the Pioneer Arm's generic controller has no provision for controlling joint speeds. The servo motors adjust to their new position as fast as they can, leading to some pretty wild swings unless you are careful. Nor, with the generic commands, can you query for the current position of a joint.

Your robot's Pioneer Arm support servers, on the other hand, give you control over joint speeds and maintain current position information, as well as a variety of other advanced and convenient features. Which is why we recommend that you program to the Pioneer Arm servers and not to the arm's generic command set.

Generic Command Syntax

The Pioneer Arm's generic control protocols consist of a header byte (255; 0xFF) followed by a command byte and, sometimes, one command parameter.

The most significant bit (b_7) of the command byte determines the mode—set the bit for query mode; clear it for servo-positioning or digital output port commands. The next three most significant bits of the command byte ($b_{4\text{-}6}$) select the board number; always 0 for the Pioneer Arm. The remaining bits of the command byte depend on the command mode.

Generic Commands

The Pioneer Arm's generic commands let you enable/disable power to the servos and query for the controller's software version number and analog/digital port states. The native Arm controller supports three digital control bits and two analog input channels. Digital channel 0 is connected to and controls power to the Pioneer Arm's servo motors. Channel 1 is connected to the status LED inside the Arm controller box.

You may use the Pioneer Arm controller's generic servo-positioning command to move a joint. The command consists of three bytes: The header (255; 0xFF), a servo-selector byte, and the position value byte. The servo-selector byte's most significant bit (b7) always is OFF (0) to distinguish servo-positioning commands from query commands. Its remaining bits ($b_{0\text{-}6}$) select the servo number, for a total of up to eight joints. Only joints one—the base swivel—through six—the gripper—are active with the Pioneer Arm.

The servo-positioning byte allows up to 255 different servo positions, although in practice the range of positions for any individual joint and depending on the individual arm varies to roughly 200 positions

around values 25-235. Our robot servers let you set realistic position boundaries (MIN and MAX) and CENTER offsets, so that your client-side code is portable from arm to arm. See following sections for details.

Table 1. Pioneer Arm generic commands

CONTROL MODE COMMAND BYTES	DESCRIPTION	BOARD RESPONSE
255, 12, 1	Servo Power ON	None
255, 12, 0	Servo Power OFF	None
255, 128	Request version number	ASCII-encoded version number, followed by CR, LF[9]
255, 133	Read digital port states	0-7 corresponding to ports 0, 1 and 2, followed by CR, LF
255, s, p	Set servo joint s=0-7 to position p=0-255	None

For example, using the generic command to turn the Pioneer Arm fully clockwise around its base (servo #1), you may send the generic command byte sequence:

255, 1, 255

Similarly, use a three byte command to power the Pioneer Arm servos: The header (255; 0xFF), command byte 12 (0x0C), followed by a byte value of one to enable, or zero to disable servo power.

Generic Command Communications

The Pioneer Arm's controller is attached to the AUX serial port on your C166, to either the AUX1 or AUX2 serial port on your H8S-, or to either the AUX1, AUX2, or AUX3 serial port on your SH2-based Pioneer controller.[10] Use the client TTY2 command #42 to send generic commands from your client software to the Pioneer Arm controller through AUX1; command #66 (TTY3) for AUX2; or command #60 (TTY4) for AUX3. Include the Pioneer Arm's generic command as the integer or string argument.

To retrieve responses from the Pioneer Arm controller using generic commands, use the respective GETAUX commands (#43, #67, #61 for AUX1-3) and corresponding SERAUXpac (types 176, 184, and 200, respectively) server information packet.

Command and packet details are in your robot's *Operations Manual*. MobileRobots' ARIA client-development environment supports these functions, too.

Pioneer Controller Arm Servers

Your robot's controller has software with advanced control functions for safe and convenient operation of the Pioneer Arm. These include commands to enable and disable power to the arm's servos, to move the arm to a new position—joint-by-joint and each at a controllable speed—and to query for current arm positions and status. The servers also include safety watchdogs and which act to minimize power consumption as well as help protect your Pioneer Arm and surroundings from damage.

Client Commands

P2OS, AROS, and ARCOS all support identical servers for control of the Pioneer Arm. Commands originate in your client software and thereby require that you establish a client-server connection between your PC and the robot's controller. After connecting, your client software must tell the servers to power the Pioneer Arm's servos before you can send position commands. Note that the arm always starts a client-server connection in its HOME position and automatically returns to that position when servo power is set to OFF, including at the termination (CLOSE) of a client-server connection session.

[9] Carriage return (13; 0x0D) and line-feed (10; 0x0A) characters.
[10] Depends on physical attachment and ArmPort setting in FLASH. See following chapters for details.

Table 2. Pioneer Arm client-command set

CLIENT COMMAND	COMMAND NUMBER	VALUE(S)	DESCRIPTION
INFO	70	none	Request an ARMINFOpac server-information packet
STATUS	71	0, 1 or 2	Request one or a continuous stream of ARMpac information packets; 0 stops continuous
INIT	72	none	Warm-reset the Pioneer Arm
CHECK	73	none	Has Arm servers check to make sure it's still responding to commands; reflects in status byte of ARMpac.
POWER	74	0 or 1	Switches power off (0) or on (1) to all the Arm servos
HOME	75	1-6 or >6	Sends all (>6) or specified joint (1-6) to the HOME position
PARK	76	none	Sends all joints to their HOME positions and shuts power off.
POS	77	two bytes: 1-6, 0-255	Sends joint (1st byte) to the position specified in the 2nd byte, subject to end-limits specified in FLASH parameters
SPEED	78	two bytes: 1-6, 1-127	Delay in milliseconds (2nd byte) between incremental steps to control the speed and motion of the joint (1st byte)
STOP	79	1-6 or 255	Stop the specified (1-6) or all (255) moving joints
AUTOPARK	80	0-65535	Disable the autopark watchdog (0) or reset it to some time in seconds other than default AutoParkTimer in FLASH
GRIPPARK	81	0-65535	Disable the gripper watchdog (0) or reset it to some time in seconds other than the default GripperParkTimer

Once connected, you may query for status, current servo positions, and other salient information as included in special arm server-information packets, as well as modify joint speeds. But the Pioneer Arm will not move nor will the servers update joint positions unless your robot's controller verifiably has communication with the arm's controller and the arm's servo power is enabled.

There is a special warm-reset protocol your client must perform in the case that the Pioneer Arm controller is not in contact with the robot's controller when your client software requests to enable servo power, or in the case that communication between them is lost during an active session.

Your client software must package arm commands into a special communication packet, and retrieve and decode the Pioneer Arm-related server-information packet (SIP), as described in your robot's *Operations Manual* and supported in part by various MobileRobots development software.

INFO and ARMINFOpac

The robot's arm server software maintain detailed information about each servo-driven joint, as well as connection and power status. Your client software may request that detailed information with the `INFO` client command #70, which prompts your robot controller to respond with an `ARMINFOpac` SIP. See your *Operations Manual* for details about SIPs and guidelines on how to write client-side packet decoders. Arm-related SIP decoders come with the provided ARIA software.

POWER

After establishing a client-server connection, you must send the client command #74 with argument value of one to have power enabled to the Pioneer Arm's servos in order to move its joints to positions other than `HOME`.

If you subsequently tell the servers to disable power to your arm's servos (command #`74,0`), or if your client disconnects when the servos are enabled, the arm automatically returns to its `HOME` position and disables servo power.

Table 3. ARMINFOpac server information packet contents

BYTE	VALUE(S)	DESCRIPTION
-2	0xFAFB	Header signals start of packet
0	n+53	Number of data bytes, including checksum
1	161	ARMINFOpac server information packet ID
2 thru n	string	NULL-terminated string containing the Pioneer Arm's version number response, or "No arm" if not connected.
n+1	0-7	Number of Pioneer Arm joints; default is six; 0 if no Arm.
n+2	0-127	Servo #1's speed setting.
n+3	0-255	Servo #1's home position setting.
n+4	0-127	Servo #1's minimum position value.
n+5	~127	Servo #1's center position value.
n+6	127-255	Servo #1's maximum position value.
n+7	1-255	Servo #1's ticks per 90 degree rotation value.
n+8 to n+50	…	Servo #'s 2-6 speed, home, minimum, center, maximum, and ticks/90deg settings.
n+51, 52	xx	Data checksum

STATUS and ARMpac

The ARMpac SIP provides current Pioneer Arm servo position information as well as arm status. Get one or a continuous stream (one per standard SIP cycle) of ARMpac SIPs with the STATUS command.

Table 4. ARMpac contents

BYTE	VALUE(S)	DESCRIPTION
-2	0xFAFB	Header
0	11	Data byte count
1	160	ARMpac server information packet ID
2	bit-endcoded status	Bit-encoded servo power (b0) and connection (b1) status; respective bit is set (1) if servo power enabled or a connection established between Pioneer Arm controller and the robot's microcontroller.
3	bit-encoded status	Bit-encoded motion status for each servo-driven joint; bit set (1) if respective joint in motion. Note that only bits 1-6 are active, corresponding to the current Pioneer Arm's six servo motors and joints.
4-9	0-255 ea	Servos 1-6 current positions.
10, 11	xx	Data checksum

If, in the case that the robot's controller cannot communicate with the arm's controller, or if it loses communication while servo power is enabled, the arm servers automatically revert to the HOME positions with servo power OFF and will not allow restoration of servo power until reset. See INIT and Warm Reset Protocols below.

SPEED and POS

The Pioneer Arm servers move a joint from one position to another in individual steps, rather than effect an uncontrolled jump from one position to another as happens with the generic arm commands. And the arm servers control each joint's rotational speed by waiting a particular period of time

between each step for much smoother operation.

Accordingly, to move joint #1 from position 100 to position 200, for example, the robot's controller software sends a sequence of 100 individual position commands to the Pioneer Arm controller, each incremented by one position value and separated by a SPEED number of milliseconds until the joint reaches its destination.

You may set individual joint speeds on-the-fly with the SPEED command #78 or set their start-up defaults through FLASH parameters (see next chapter); 1-127 milliseconds delay each. Speeds may be set at any time, even when the arm is not powered, and persist through servo-power and client connection/disconnection cycles. Their current values are reflected in the ARMINFOpac.

When powered, control the Pioneer Arm's servo positions with the POS command #77. Servos vary. Each joint for your particular arm is characterized by maximum and minimum position values which limit its range despite the value you may command, as well as a center position. Limiting values for each and every joint are reflected in the ARMINFOpac, and may be updated in FLASH (see next chapter). Individual servo positions are reflected in the ARMpac server information packet (see above).

SPEED Limits

Joint speeds have limits. Generic arm-position commands take two bytes each. Serial communication with the Pioneer Arm controller is 9600 baud, which is about 1 byte per millisecond. Hence, an individual joint may move at the maximal speed at about one new position every two milliseconds, even though the server attempts to update each joint every millisecond.

The delays are cumulative, too. Move two joints at a time, and the maximal update rate, due to serial communication delays, is every four milliseconds. And so on.

Fortunately, the worst case—all six joints moving at once and requiring simultaneous position updates—is rare, for which a complete update cycle takes 2 x 6 = 12 milliseconds. As it happens, the larger, heavier joints move best at slower speeds, and speed-controlling delay times of 15 milliseconds and above are smoothest and plenty fast for most manipulation applications.

HOME and PARK

As described earlier, HOME is a safe and stable position for when the Pioneer Arm servo motors come to rest. When switching servo power, the HOME position minimizes the drop as the arm goes limp when powered OFF, as well as the snap-to-position jumps characteristic of servo motors as each joint seeks its position when powered ON. Indeed, your robot controller's arm servers ***always*** presume that the Pioneer Arm is stored in its HOME position except when its servo motors power is ON.

Use the HOME command #75 to send one (servo number 1-6 as command parameter) or all (command parameter >6) of your Pioneer Arm joints to their HOME, default positions.

Use the PARK command #76 to send all the joints to their HOME positions and shut power OFF to the arm. The HOME and PARK commands are ignored if servo power is OFF.

INIT, CHECK, and Warm Reset Protocol

Upon Main Power start up or controller reset, the Pioneer Arm servers automatically initialize the arm controller, setting the arm joints to their default HOME positions and their speeds to the defaults stored in FLASH. It does not apply power to the arm's servos.

Instead, initialization includes *removing* power from the arm's servo motors. Please be careful to recognize the implications: Upon reset, your robot controller's arm servers cannot know the positions of arm joints, and therefore presume, upon initialization, that they are at or near their HOME positions. When reset, the controller's arm servers cannot and will not drive the Pioneer Arm servos to their HOME positions, as they may do under other circumstances.

CAREFUL!

Put the Pioneer Arm into its HOME position
BEFORE
resetting the microcontroller.

Your Pioneer Arm may flop dangerously if it is not in its HOME position, and consequently flail erratically to that position when its servos get powered. So be very careful to have the Pioneer Arm in its HOME position before you turn on your robot's Main Power or reset the robot's microcontroller!

The arm servers also monitor the serial connection with the arm's controller. If the robot's controller does not detect the arm, or if it loses contact with the arm's controller when the arm's servos are enabled, your robot's controller automatically shuts down command and control, and awaits a warm-reset command from the client to re-establish command and control of the Pioneer Arm through the arm servers.

In the meantime, you should manually restore the Pioneer Arm to its HOME position.

For your client to recover command and control of the Pioneer Arm while remaining connected after the communication failure, you must follow the warm-reset protocol:

1. Request status (STAT) or CHECK and wait for restoration of communication (bit 1 of the status byte)
2. Issue the INIT command
3. Use STAT and INFO SIP requests to reinitialize any client-side variables that my be tracking joint positions
4. Issue the POWER command to power the Arm servos
5. Resume operation

Note that while the joint positions are reset to HOME; the latest joint speed settings persist.

ARIA and ArAKin Arm Support

The classes and methods for ARIA-based client operation of the Pioneer Arm and the companion ArAKin and AKIN forward and inverse kinematics libraries are documented extensively in their respective ARIA directories. ARIA is the preferred foundation software for client-server

communications and control of MOBILEROBOTS platforms and accessories. It's free, published under the GNU General Public License, and comes not only with source code, but with many examples of robot and accessory control client software.

ArAKin libraries depend on the licensed AKin forward/inverse kinematics libraries. New Pioneer Arm owners automatically receive a license. Legacy owners may purchase a license from sales@mobilerobots.com.

ArP2ArmSimple, ArAKinDemo and ArAKinExample

All current ARIA distributions contain sources for individual joint- and general Pioneer Arm maintenance client activities, as well as the ArAKin kinematics libraries. Install the software into your ARIA directory, typically in `C:\Program Files\MobileRobots\Aria` or `/usr/local/Aria`. Be sure to use matching ARIA and ArAKin versions.

In the generic `Aria/examples` subdirectory, find and create the `ArP2ArmSimple` program which shows you how to connect and operate the individual arm joints, as well as monitor arm states and positions through ARIA. In the nearby `Aria/ArAKin/examples` subdirectory, build and execute the ARIA-integrated, AKin-based kinematics demonstrations, `ArAKinDemo` and `ArAKinExample`.[11]

These simple demonstration programs just exercise the Pioneer Arm, but modify their source code at will to incorporate your own actions with other Pioneer robotics functions.

[11] Windows executables get put into the Aria\bin directory.

Chapter 4 Arm FLASH Parameters

Your robot controller's operating software—P2OS, AROS, or ARCOS—use a variety of robot- and accessory-specific settings that are stored in its FLASH memory. These include important settings for your Pioneer Arm.

We describe the Pioneer Arm-related FLASH parameters in detail here. Please consult your robot's *Operations Manual* for more details about all the other FLASH configuration parameters and supporting utilities.

Firmware Tools and Parameters

Your robot controller maintains six operating values for each of the six joints of your Pioneer Arm. These include `HOME`, `CENTER`, `MIN` and `MAX` positions, default joint `SPEEDs`, and `TICKS-PER-90DEGREES`. `HOME`, of course, is the default, resting position for each joint mentioned often in previous chapters.

The `CENTER` position values are unique to your Pioneer Arm, and they are calibrated at the factory. They describe the position value where a joint is half-way between its minimum and maximum mechanical rotation. These values are not used explicitly by the arm servers, but are available to your client software through the `ARMpac` server information packet. You may reset these `CENTER` values, but we caution against it.

Likewise, although the Pioneer Arm's controller accepts position values 0-255, we set limits on the range of position values for each joint at the extremes (`MIN` and `MAX`); the servos' behaviors beyond these extremes are unpredictable.

The `SPEED` parameters are default joint speeds in 1-millisecond increments. Unlike the other FLASH-based position values, joint speeds may be modified by your client software through the `SPEED` client command #78. See the previous chapter for details.

The `TICKS-PER-90DEGREES` define the number of unique positions—ticks—the individual servo takes for a 90-degree swing in the joint. Use this value to convert absolute joint positions into degrees.

Installing the FLASH Utilities

Your MobileRobots platform comes with the software on CD-ROM and installed on your robot. Thereafter, stay tuned to the `pioneer-users` newsgroup or periodically visit our support website to obtain the latest versions and related documentation:

```
http://robots.MobileRobots.com
```

Use the version that matches your host computer's environment and robot operating system. For example, use ARCOSV_r.tgz for Linux or ARCOSV_r.EXE with Microsoft Windows© systems and SH2-based Pioneers ("V" is the version number and "r" is the revision number; ARCOS1_3, for example).

The OS distributions are self-extracting, self-installing packages: Simply follow the on-screen directions. For the Linux/UNIX versions, uncompress and untar them using the appropriate system utilities. For example, with the Linux version:

```
% tar -zxvf ARCOS1_1.tgz
```

The command creates an `ARCOS/` directory in the current path and stores the software there.

With Windows, the ARCOS directory default path is `C:\Program Files\MobileRobots\ARCOS`.

Configuring Pioneer Arm Operating Parameters

The ARCOScf(.exe), AROScf(.exe), or p2oscf(.exe) program is the tool you use to view and change your SH2-, H8S-, or C166-based, respectively, Pioneer controller's identity and operating parameters.

Step 1. Serial Connection from Computer to Robot

Connect your robot's HOST controller to your host PC through their respective serial ports. If you have an onboard PC, use it to perform the configuration updates. Otherwise, switch its power OFF and use a direct ("straight-through") serial tether from an off-board PC to the 9-pin SERIAL connector on your robot's User Control Panel.

If you have an onboard radio modem, you may have to remove its serial connector from the C166 controller's HOST serial port by reaching in with your fingers through the rectangular access port on the robot's deck. Sorry about the inconvenience—the modem interferes with the direct connection.

To update P2OS, you may need to remove your onboard radio modem's serial connector from the HOST serial port inside the robot on the C166 microcontroller.

Step 2: Enable FLASH (C166 only)

Locate the FLASH switch on the C166-based Pioneer 2 deck. It's recessed and may be covered by an accessory. Use a flat-bladed screwdriver or other thin instrument to move the slide switch forward toward the front of your robot to enable FLASH writes.

Step 3: Put Controller into Maintenance Mode (ARCOS, P2OS, and AROS)[12]

Start up or reset your robot. After it has finished initializing, place it into FLASH-maintenance mode:

1. Press and hold the white MOTORS button.
2. Press and release the red RESET button.
3. Continue holding the MOTORS button for three or more seconds (P2OS).
4. Release the MOTORS button.

The robot should not reset. If it does, you probably didn't hold the MOTORS button down long enough. Try again. When successfully in FLASH-maintenance mode, notice that the "heartbeat" asterisk stops blinking in the C166 Console LCD. With H8S- and SH2-based robots, the STATUS LED blinks rapidly and the BATT LED lights bright red.

Table 5. FLASH configuration startup options

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

Step 4: Run the cf Program

Find the "cf" configuration program in your aptly named distribution directory (ARCOScf in the ARCOS/ directory, for example). The program accepts optional arguments; the argument-less default startup is to connect with your robot's controller through your PC's serial port /dev/ttyS0 with Linux/UNIX systems or COM1 with Windows. With the default, you may double-click to launch the respective ".exe" directly from your desktop, rather than executing the program from the MS-DOS Prompt window or the

[12] This also works with SH2-(ARCOS-)based robots, but simply connecting ARCOScf will cause the microcontroller to engage Maintenance Mode.

Start:Run dialog. Notice that ARCOScf automatically engages the SH2 microcontroller's Maintenance Mode.

Step 5: Changing Arm Configuration Parameters

Normally when it starts, your robot-FLASH configuration program automatically retrieves the current identifying and operating parameters from your robot's controller. Use the 'a' or arm keyword to see a list of the working values. Thereafter, type the listed keyword alone to review its individual value, or type the keyword and give a new value to change the FLASH parameter.

While working, you change a temporary copy of the FLASH parameters in your PC's memory. Your changes are not committed to your robot's FLASH memory until you choose to explicitly "save" them.

Step 6: Save Your Work

Use the save command to save configuration changes to your robot's FLASH. If you don't save your changes to FLASH, they won't take effect.

We also strongly recommend that you save your work to disk, as well, for later retrieval should your controller get damaged or it's FLASH inadvertently erased. To save your current configuration to a disk file, simply specify a path and filename with the save command. For example:

```
> save myP3DX
```

copies the changes you've made on the PC to a disk file. Be careful: this does not download the changes to your robot's FLASH. Use save alone to do that.

Use the restore command to not only retrieve the current operating parameters from the robot, but to load a saved parameters files from disk.

You may edit the file-restored parameters just as you edit those retrieved from the robot. And you may save those edited parameters over the same file or a different one, using the save command. Note that file-restored configuration parameters are not necessarily the same as those stored in the robot's FLASH. You must save them there separately. Default parameter files come with each distribution, but it is tedious to reconstruct an individual robot's unique configuration, particularly the unique Arm values.

Editing the Arm Parameters

To view the list of current FLASH values for your Pioneer Arm, type **'a'** or "arm", followed by a return (Enter). Type '**?**' or **'help'** to see a list of p2oscf commands.

Table 6. Configuration commands

COMMAND	DESCRIPTION
keyword <value>	Alone, keyword displays current, edited value. Add value argument to change current value.
c or constants	Display constant values (P2OS only)
v or variables	Display current, edited operational values; may be different than values currently stored in FLASH.
a or arm	Display current, edited FLASH-embedded values for control of the Pioneer Arm accessory.
r or restore <pathname>	Restores edited variables to values currently stored in FLASH or from file, if pathname argument included.
save <pathname>	Saves current edited values to FLASH and exits program or saves current edited values to disk for later reference and continues in editor.
q or quit	Exits without saving any changes to flash.
? or help	Displays commands and descriptions.

Table 7. Pioneer Arm FLASH parameters

KEYWORD	DEFAULT VALUE	DESCRIPTION
ArmPort	0	AROS/ARCOS only; 1= arm controller connected to AUX1; 2=AUX2 serial port; 3=AUX3; 0=no Arm
NumJoints	6	Number of servo joints; with P2OS, 0 if there is no Pioneer Arm.
AutoParkTimer	600	Pioneer Arm will autopark if it does not receive a command in time seconds; 0 disables
GripperParkTimer	300	Gripper will autohome if it is closed, presumably gripping something, for time seconds; 0 disables

Table 8. Example Pioneer Arm FLASH parameters for individual servo joints

KEYWORD* JOINT #	JOINT 1	JOINT 2	JOINT 3	JOINT 4	JOINT 5	JOINT 6
Home#	125	30	20	130	20	100
Speed#	20	20	20	20	20	20
Min#	1	2	1	1	1	28
Center#	fc¥	fc¥	fc¥	fc¥	fc¥	fc¥
Max#	223	230	215	224	233	250
TickPer90Deg#	fc¥	fc¥	fc¥	fc¥	fc¥	fc¥

*Replace keyword # suffix with joint number 1-6; ServoMin4, for example.
¥ Cailbrated at the factory.

With the Pioneer Arm parameters, several relate to the individual servos, so that the number at the end of the keyword refers to the joint. Values are a positive decimal or hexadecimal ("0xnn") numbers from 0 to 255 (0xFF), except speeds, which are constrained from 1 to 127.

For example, to change the home position for joint #3:

```
> Home3 25
```

or

```
> Home3 0x19
```

Chapter 5 Maintenance & Repair

There are no customer-serviceable parts on your Pioneer Arm.

Use MOBILEROBOTS authorized parts *ONLY*;
warranty void otherwise.

Factory Repairs

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

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

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

We will try to resolve the problem through communication. If the Arm 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

SPECIFICATIONS

CONSTRUCTION: anodized, CNC fabricated, and painted aluminum, and plastic, with foam-covered gripper fingers
MOTION: 5 DOF arm and 1 DOF gripper
POWER: +5 and +12 VDC supplied by Pioneer robot
ARM RANGE: 50 cm fully extended
GRIPPER RANGE: Gripper fingers part to 5 cm
PAYLOAD: 150 gm (5 oz.) lift capability
SPEED: 1 second from fully extended to fully relaxed position

Warranty & Liabilities

Your MobileRobots platform is fully warranted against defective parts or assembly for one year after it is shipped to you from the factory. Accessories are warranted for 90 days. Use only MobileRobots authorized parts or warranty void. This warranty also 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 MobileRobots products shall bear no liabilities for operation and use of the robot or any accompanying software except that covered by the warranty and period. The developers, marketers or manufacturers shall not be held responsible for any injury to persons or property involving MobileRobots products in any way. They shall bear no responsibilities or liabilities for any operation or application of the robot, or for support of any of those activities. And under no circumstances will the developers, marketers or manufacturers of MobileRobots products take responsibility for support of any special or custom modification to MobileRobots platforms or their software.

19 Columbia Drive
Amherst, NH 03031 USA
+1 (603) 881-7960
+1 (603) 881-3818 fax
http://www.MobileRobots.com