Seekur Jr Operations Manual

Important Safety Instructions

- ✓ Read the installation and operations instructions before using the equipment.
- ✓ Avoid using power extension cords.
- ✓ To prevent fire or shock hazard, do not expose the equipment to rain or moisture.
- ✓ Refrain from opening the unit or any of its accessories.
- ✓ Keep wheels away from long hair or fur.
- ✓ Never access the interior of the robot with charger attached or batteries inserted.

Inappropriate Operation

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

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

Table of Contents

Tables

Chapter 1 Introduction

Congratulations on your purchase and welcome to the rapidly growing community of developers and enthusiasts of MobileRobots intelligent mobile platforms.

This ***Seekur Jr Operations Manual*** provides both the general and technical details you need to operate your new intelligent robot base and to begin developing your own robotics software.

ROBOT PACKAGES

Our experienced manufacturing staff put your mobile robot and accessories through a "burn in" period and carefully tested them before shipping the products to you. In addition to the companion resources listed above, we warrant your MobileRobots platform and our manufactured accessories against mechanical, electronic, and labor defects for one year. Third-party accessories are warranted by their manufacturers, typically for 90 days.

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

Basic Components (all shipments)

- ✓ One fully assembled Seekur Jr mobile robot with battery
- ✓ Microcontroller with SeekurOS mobility firmware
- ✓ Heading correction gyroscope
- ✓ Joystick
- ✓ Power/Recharging module
- ✓ ARIA, MobileEyes, MobileSim, Mapper3 Basic software
- ✓ CD-ROM containing licensed copies of MobileRobots software and documentation
- ✓ Set of print manuals
- ✓ Hex wrenches and assorted replacement screws
- ✓ Replacement fuse(s)
- ✓ Registration and Account Sheet

User-Supplied Components / System Requirements

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

Figure 1. MobileRobots' platforms first appeared commercially in 1995 through the company's predecessor, ActivMedia Robotics.

Optional Components and Attachments (partial list)

- ✓ Supplementary and replacement batteries
- ✓ One or two onboard SBC computers and accessories
- ✓ Radio Ethernet-to-serial
- ✓ Laser range finder with Advanced Robotics Navigation and Localization (ARNL) software
- ✓ Global Positioning System
- ✓ mOGS outdoor navigation and localization software
- ✓ Stereo Vision Systems
- ✓ Pan-Tilt-Zoom Surveillance Cameras
- ✓ Advanced Color Tracking System (ACTS)
- ✓ Compass
- ✓ Serial cables for external connections
- ✓ Many more…

ADDITIONAL RESOURCES

New MOBILEROBOTS customers get three additional and valuable resources:

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

Support Website

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

`http://robots.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 robot.

Newsgroups

We maintain several email-based newsgroups through which robot owners share ideas, software, and questions about the robot. 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**
> From: <*your return e-mail address goes here*>
> Subject: <*choose one command:*>
> **help** (returns instructions)
> **lists** (returns list of newsgroups)
> **subscribe**
> **unsubscribe**

Our e-mail list server will respond automatically. After you subscribe, e-mail your comments, suggestions, and questions intended for the worldwide community of Pioneer users:[1]

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

Access to the `pioneer-users` e-mail 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 MOBILEROBOTS platforms.

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

Support

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

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

or by email:

`support@MobileRobots.com`

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

Tell us your robot's SERIAL NUMBER.

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

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

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

Chapter 2 What Is Seekur Jr?

Seekur Jr is the newest member of the MobileRobots family of intelligent mobile platforms. Like the Pioneer AT models, Seekur Jr is a four-wheel, skid-steer differential drive mobile robot, though it is much stronger, carries a much larger payload and is better protected against inclement weather for outdoor, all-terrain mobility.

Figure 2. Seekur Jr with some common accessories

Pioneer Reference Platform

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

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

Besides the open-systems robot-control software onboard the robot microcontroller, every MobileRobots platform also comes with a host of advanced robot-control client software applications and applications-development environments. Software development includes our own foundation Advanced Robotics Interface for Applications (ARIA) and ArNetworking, released under the GNU Public License, and complete with fully documented C++, Java and Python libraries and source code. Several third-party robotics applications development environments also have emerged from the research community, including Ayllu from Brandeis University, Pyro from Bryn Mawr and Swarthmore Colleges, Player/Stage from the University of Southern California and Carmen from Carnegie-Mellon University.

Pioneer Family of Microcontrollers and Firmware

First introduced in 1995, the original Pioneer 1 mobile robot contained a microcontroller based on the Motorola 68HC11 microprocessor and powered by Pioneer Server Operating System (PSOS) firmware. The next generation of Pioneer 2 and PeopleBot (V1) robots used a Siemens C166-based microcontroller with Pioneer 2 Operating System (P2OS) software. AmigoBot introduced an Hitachi H8S-based microcontroller with AmigOS in 2000. From 2002 until Fall of 2004, Pioneer, PeopleBot and PowerBot robots also had an Hitachi H8S-based microcontroller with *Activ*Media Robotics Operating System (AROS) software.

Now most Pioneer platforms use revolutionary high-performance microcontrollers with advanced embedded robot control software based on the new-generation 32-bit Renesas SH2-7144 RISC microprocessor, including the P3-SH microcontroller with ARCOS, µARCS with PatrolBot and the Motivity Core, and AmigoSH for AmigoBot.

The skid-steer Seekur Jr and its big daddy, the four-axis independent drive-and-steer Seekur use a FreeScale MPC565 microcontroller and the latest generation of MobileRobots's mobililty-control firmware, SeekurOS. But you might not even notice the differences. Because we have taken great care to ensure backward compatibility across *Activ*Media Robotics/MobileRobots entire history of robots, client software written to operate an ancient PSOS-based Pioneer AT will work with a brand new Seekur Jr with little or no modification. Client-server communication over a serial communication link remain identical as do support for all robotics commands.

Client Software

All MobileRobots platforms operate as the server in a client-server environment: Their microcontrollers handle the low-level details of intelligent mobile robotics, including maintaining the platform's drive speed and heading, and acquiring sensor readings, such as from the wheel encoders. To complete the client-server architecture, Mobile-Robots platforms require a PC connection: software running on a computer connected with the

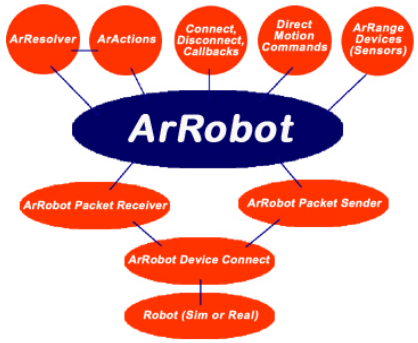


Figure 3. ARIA's architecture

robot's microcontroller via the HOST serial link and which provides the high-level, intelligent robot controls, including obstacle avoidance, path planning, features recognition, localization, navigation and so on.

An important benefit of MobileRobots client-server architecture is that different robot servers can be run using the same high-level client. Several clients also may share responsibility for controlling a single mobile server, which permits experimentation in distributed communication, planning, and control.

ARIA

Advanced Robotics Interface for Applications (ARIA) software, including ArNetworking, come with every MobileRobots platform. ARIA is a C++-based open-source development environment that provides a robust client-side interface to a variety of intelligent robotics systems, including your robot's microcontroller and accessory systems. ArNetworking provides the critical layer for TCP/IP-based communications with your MobileRobots platform over the network (as needed).

ARIA/ArNetworking is the ideal platform for integration of your own robot-control software, since they neatly handle the lowest-level details of client-server interactions, including networking and serial communications, command and server-information packet processing, cycle timing and multithreading, as well as support of a variety of accessories and controls, such as a scanning laser-range finder, heading-correction gyros and many others.

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

Modes of Operation

You may operate your Seekur Jr robot in one of three modes:

- ✓ Server
- ✓ Joydrive
- ✓ Maintenance

Server Mode

Your robot comes to you installed with the latest SeekurOS firmware in its microcontroller. In conjunction with client software like ARIA running on an onboard or other user-supplied computer, SeekurOS lets you take advantage of modern client-server and robot-control technologies to perform advanced mobile-robotics tasks. Most users run their robot in server mode because it gives them quick, easy access to its robotics functionality while working with high-level software on a familiar host computer.

Maintenance Mode

We provide two utilities for you to update and upgrade your robot's firmware. And in the special Maintenance Mode, you also adjust your robot's operating parameters that SeekurOS uses as default values on startup or reset.

We often release upgrades to the maintenance utilities and SeekurOS free for download from our website, so be sure to sign up for the `pioneer-users` email newslist**.** That's where we notify our customers of the upgrades, as well as where we provide access to robot users worldwide.

Joydrive Mode

Finally, we provide onboard software that let you drive the robot from a tethered joystick when not otherwise connected with a controlling client. See Chapter 4 for more details.

Chapter 3 Specifications & Controls

In general, Pioneer robots pack an impressive array of intelligent mobile robot capabilities into small, compact packages that rival bigger and much more expensive machines. Seekur Jr is no exception: though larger than its Pioneer AT cousin, Seekur Jr is a sturdy, all-weather platform that can handle everything from open fields to parking garages. This skid-steer, differential-drive platform operates on moderate terrain with good stability. And SeekurJr offers space, power and networking for two onboard SBCs. This opens the way for onboard vision processing, radio-based communications, laser range-finding, DGPS and many other means to autonomous operation.

With its powerful SeekurOS-based microcontroller server and advanced MobileRobots client software, Seekur Jr is fully capable of mapping its environment—indoor and out-of-doors, finding its way home and performing other sophisticated path-planning tasks.[2]

Physical Characteristics and Components

Weighing in at 70 kilograms (154 pounds with one battery), the basic Seekur Jr's strong aluminum body and solid construction make it virtually indestructible. Its two powerful DC motors and four-wheel drive transmission enables Seekur Jr to carry up to 50 Kg (110 lbs), reach speeds up to 1.2 meters per second [2.7mph] and climb slopes up to 75%. It is ruggedized for outdoor operation on rough terrain (IP rating of 54 at a temperature range of -5 to 35°C. Yet Seekur Jrs is small enough to comfortably fit through standard doorways and into labs.

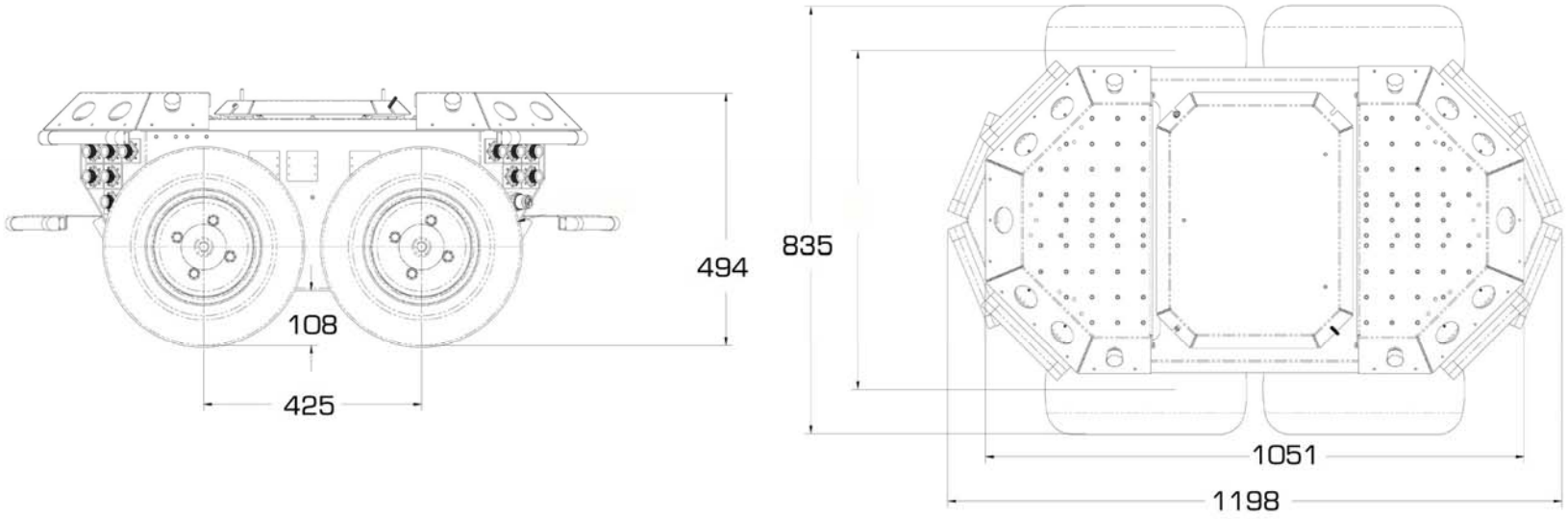


Figure 4. Seekur Jr's physical dimensions

[2] Requires a laser range-finder accessory and special Advanced Robotics Navigation and Localization (ARNL) software.

Seekur Jr uses differential-drive controls to rotate nearly in place and its heavy-duty, wide rubber pneumatic tires provide for all-terrain operation. However, Seekur Jr's tank-like skid-steering can be brutal on high-friction surfaces, such as carpet, so we recommend operation either on unconsolidated or slip-easy surfaces like grass, sand or painted concrete to significantly reduce motor and motion stresses.

CONNECTORS AND INTERFACES

Seekur Jr supports eight standard connectors and up to 12 additional ones depending on accessories. Most are sturdy, weather-proof

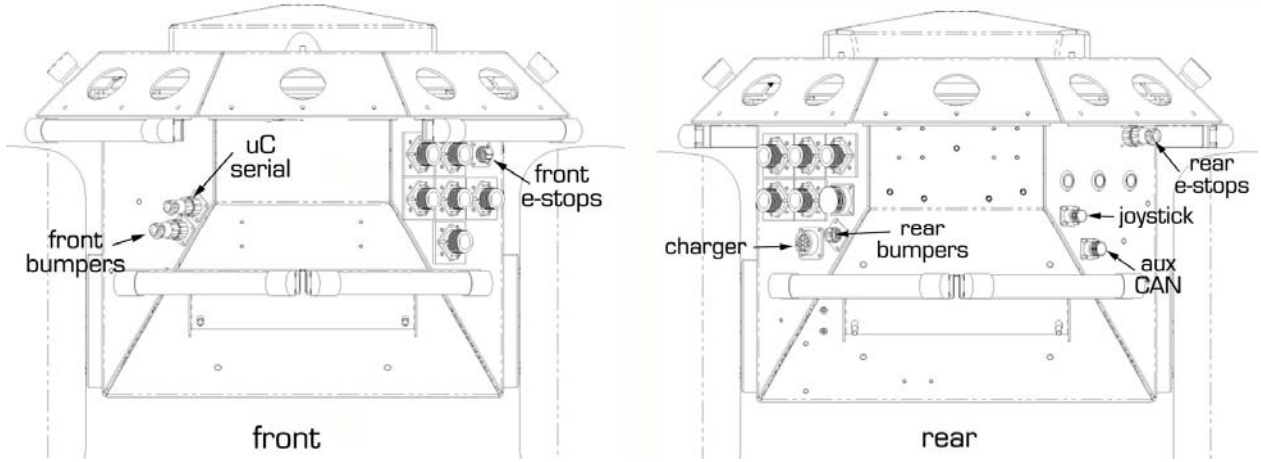


Figure 5. Seekur Jr standard connectors; unlabeled connectors populated with accessories; rear top bumpers removed for clarity

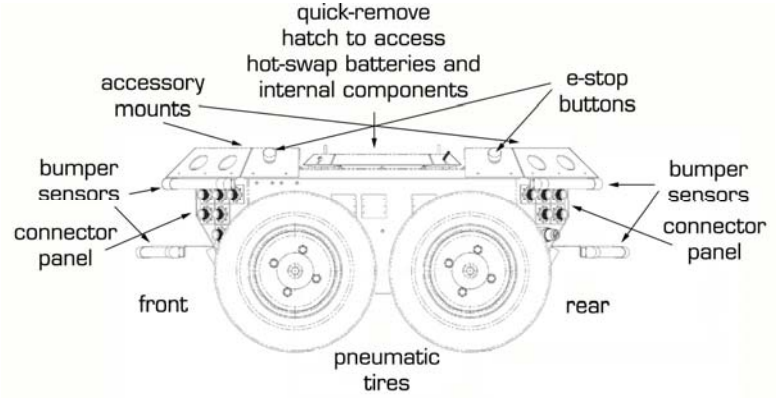


Figure 6. Seekur Jr features

Circular Plastic Connectors (CPCs). Pinouts are provided in the *Appendix A*.

The e-stop buttons and bumpers must be connected in order to operate Seekur Jr. The uC serial port contains the Maintenance Serial interface used for updating SeekurOS (see *Chapter 7* for details). It may also contain the HOST serial port used for client-server communications if there is no onboard SBC.

The joystick and aux CAN ports are CAN-bus interfaces to the microcontroller.

BATTERIES AND CHARGING

Seekur Jr operates with one to three Nickel-Metal Hydride (NiMH) 24 VDC hot-swappable battery packs, located just beneath a quick-remove hatch or from the Power/Recharging Accessory plugged into the Power connector on the left rear panel of Seekur Jr's body.

Installing and Removing the Batteries

One of the first things that you will do when Seekur Jr arrives into your workspace is install its batteries. It is a five-step process to insert or remove one or more battery packs: Remove the top hatch, open the battery pack restraints, slide one and up to three battery packs into position, flip down the restraints to secure the pack(s), and attach a power cable to each battery. See adjacent illustrations for details.

Carefully align the cable and connector keys to avoid damage.

To remove a battery pack, open the top hatch, remove the power cable(s), relieve the restraints, lift out the battery pack(s), and then re-secure the restraints.

Of course the final thing that you should do in either case is re-secure the top hatch with its four latch screws.

Charging and Runtimes

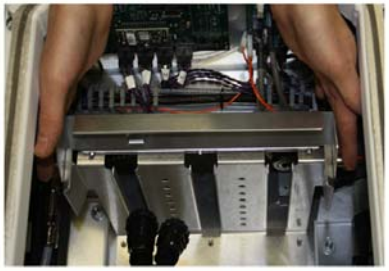
Seekur Jr's onboard systems will operate continuously, but not drive[3] while charging its batteries onboard from an attached Power/Recharging Module. Battery operating times vary depending on mobile activity mostly: Seekur Jr will run for up to three hours of continuous motion starting with three fully charged batteries. One battery provides nearly one hour of continuous run time. The onboard systems will run over four times as long on batteries if the robot is not driven.

For nearly continuous battery operation, use hot-swapping with additional battery packs: Either temporarily connect the Power/Recharging Module to provide systems power while removing exhausted battery packs, or keep at least one battery in place while replacing the other(s).

Recharge the batteries either onboard or offboard the robot. The Power/Recharging Module provides sufficient charge power for up to

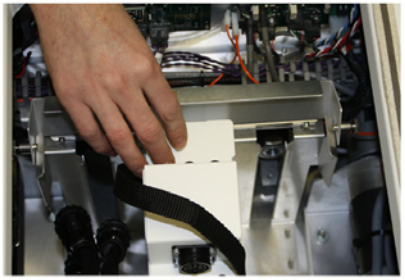
[3] Of course not and we make sure that doesn't happen by disabling power to the motors when the Power/Recharging Module is connected. The MOTORS button blue LED indicator double-blinks in that state (see *Power and Motors* below).

STEP 1: Loosen the four latch screws at the corners and lift off Seekur Jr's top hatch.



STEP 2: Inside, pull the spring-loaded pin latches on each side of the battery mount and open the two restraints on their hinges.

STEP 3: Slide one to three battery packs into position along the guides with their power connector facing the power wires.



STEP 4: Flip the two battery restraints down and latch them to secure the packs.

STEP 5: Attach a power cord to each pack (no order – any cord to any battery) and twist the connector to secure.

three batteries at a time while also maintaining sufficient power for continuous operation of the onboard systems.

It takes three or more hours to recharge one or more of Seekur Jr's NiMH battery pack(s). Accordingly, recharge to runtime ratio is approximately 3:1 for a single battery and 1:1 for three batteries.

Charge Indicators

SeekurOS keeps tabs on the state of charge for each battery and notifies a connected client like the ARIA demo of the charge state, including when the batteries get below a LowSOC operating parameter value (currently 10%). If the batteries go below a ShutdownSOC of 5%, SeekurOS notifies the onboard SBCs and within one minute automatically shuts down all of Seekur Jr systems.

When starting up Seekur Jr with its batteries below ShutdownSOC, systems power is removed after about ten seconds. Currently, there is no other visual or audio warning of a low-battery condition.

Power and Motors

Three weather-proof user-operated buttons on the right rear body of Seekur Jr control main power and power to the motors. Four Emergency Stop (e-stop) buttons at the top corners of Seekur Jr also control power to the robot (see E-Stop below).

Main Power

Press the left-most POWER ON button (green when lit) to supply power to the onboard systems. Press the center POWER OFF button (red when lit) to immediately remove power from all the onboard systems.

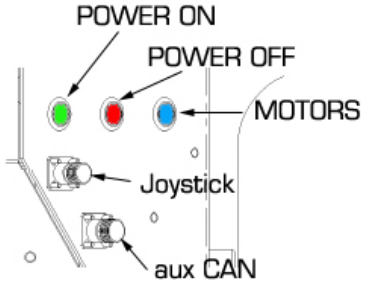


Figure 7. Seekur Jr power and motor buttons (right rear panel)

If the robot does not respond, make sure all the e-stop buttons are connected and relieved and that the robot has batteries or is connected to an external power supply. If power shuts down ten seconds or so after startup, battery charge probably is below ShutdownSOC of 5% and need recharging.

E-Stops

Press any one of the four red e-stop buttons mounted on top near the four corners of Seekur Jr to immediately remove main power from the robot. Cables to the buttons run beneath the accessory mount panels front and rear and connect to the interior power circuitry via weather-proof Circular Plastic Connectors (CPCs) on the front and rear body panels. Seekur Jr will not operate without proper e-stop connections.

To relieve an e-stop, twist the cap clockwise until it pops up. You'll then have to restart the robots power to continue operation.

MOTORS

The third user-operated button next to the main-power buttons is the MOTORS button (blue when lit). It must be pressed at least once after starting power to the robot in order to engage hardware-mediated power to the motors. The button does not engage the motor drivers, however. That sequence is a bit more complicated and depends on the readiness of the microcontroller (µC), motor-control initialization and client or joystick connections.

As shown in the truth table nearby, motors status depends on whether and when the MOTORS button gets pressed and whether or not SeekurOS has completed its startup and has initialized its motor-control systems. Startup takes about ten seconds. Motor initialization takes about 45 seconds to complete and happens only after SeekerOS power startup and after the MOTORS button gets pressed. Accordingly, most users will press the MOTORS button immediately after pressing the main POWER ON button.

Table 1. MOTOR power sequence and status indication

µC ready?	MOTORS pressed?	MOTORS LED
N	N	OFF
N	Y	single blink
Y	N	double blink
Y	Y	ON
Y	again	double blink*

* Motors disabled, too, when the Charging/Power Module is attached.

Once the motors have power and the control systems are fully initialized, you may press the MOTORS button again to manually disable power to the motors and thereby immobilize Seekur Jr without removing main power. Press the MOTORS button again to immediately

restore power to the motors—the motor-control systems don't need to re-initialize.

To actually drive the robot, the motors need to be engaged through software, either from the joystick or through a motors-enable software command from a connected client, such as the ARIA demo.

BUMPERS

Bumpers fore and aft provide contact sensing for when other sensing has failed to detect an obstacle.

Electronically and programmatically, the bumpers trigger digital events which are reflected in the `STALL` values of the standard server-information packet that SeekurOS automatically sends to a connected client. The bumper segments are numbered clockwise from one to four.

Your robot may not move if you unplug one or both bumpers.

SeekurOS itself monitors and responds whenever one or more bumper segments get triggered while the robot is moving in the same direction (front forward or rear reverse). In that case, the robot stops moving at its top deceleration rate and then disables the motors, just as if you had pressed the MOTORS button

SAFETY SEEKUROS WATCHDOGS

SeekurOS contains a communications `WatchDog` that will halt the robot's motion if communications between a PC client and the robot server are disrupted for a set time interval (two seconds, by default). The robot will automatically resume activity, including motion, as soon as communications are restored.

SeekurOS also contains a stall monitor. If the drive exerts a PWM drive signal that equals or exceeds a configurable level (`StallVal`) and the wheels fail to turn, motor power is cut off for a configurable amount of time (`StallWait`). SeekurOS also notifies the client which motor is stalled. When the `StallWait` time elapses, motor power automatically switches back on and motion continues under client control.

Chapter 4 Accessories

MobileRobots' platforms, including Seekur Jr, have many accessory options. For convenience, we include a description of the more commonly integrated accessories in this document. Please also refer to the detailed documents that come with the accessory.

Joystick and Joydrive Modes

Like its many MobileRobots cousins, Seekur Jr has a joystick interface with its microcontroller and SeekurOS contains a `JoyDrive` server for manual operation. Start driving your robot with its joystick any time it is not connected with a client. Simply plug it into the joystick port and press the GO button to engage the motors.

Well, maybe: If you had pressed the MOTORS button and waited until the motors are initialized according to the discussion in the previous chapter, then yes, the joystick's green LED indicator will light when you press the GO button and you will be able to manually drive the robot. The LED won't light if the motors are not yet ready.

Figure 8. Seekur Jr joystick

To drive your robot with the joystick while it is connected with an ARIA client (overrides client-based drive commands for manual operation), you must have the client software send the SeekurOS command #47 with an integer argument of one to enable the SeekurOS joystick servers. Have your client send the SeekurOS command #47 with an integer argument of zero to disable the joystick drive override.

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

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

Client Connections and Accessories

All MobileRobots platforms are servers in a client-server architecture. You supply the client computer to run your intelligent mobile-robot applications. The client can be either a piggy-back laptop or embedded SBC, or an off-board PC connected through radio modems or wireless serial Ethernet.

In all cases, the client computer connects with the microcontroller through a dedicated `HOST` serial port in order for the robot and your software to work. The connection may be direct internally or, in the absence of an onboard SBC, via the uC serial port on the front right panel. We provide the necessary cables.

The wireless Ethernet-to-serial accessory allows you to emulate that serial cable over the network. It works by automatically translating network-based Ethernet packet communications into streaming serial for the robot microcontroller and back again. Accordingly, you run the client software on a remote LAN-based PC.

However, a major disadvantage of the wireless Ethernet-to-serial device is that it requires a consistent wireless connection with the robot. Disruption of the radio signal—a common occurrence in even the most modern installations—leads to poor robot performance and very short ranges of operation.

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

Integrated SBCs

Mounted inside just behind the nose and at the rear of Seekur Jr can be two integrated Single Board Computers (SBCs). Each is an EBX form-factor board that comes with two 10/100Base-T Ethernet ports. With special cabling and connectors accessible inside the robot, the SBCs also support four serial ports, monitor, keyboard and mouse, audio line input and output, two USB, floppy, as well as two shock-mounted IDE hard-disk drives, 32 digital I/O and eight analog ports. For additional functionality, such as video framegrabbing, firewire, CAN

or PCMCIA bus, the onboard SBCs accept PC104 and PC104-plus (PCI bus-enabled) interface cards that stack on the motherboard.

The rear SBC is the primary and is wired to communicate with the robot's microcontroller HOST serial port through a dedicated serial port `COM1` under Windows or `/dev/ttyS0` on Linux systems. Note also that some signals on the microcontroller's HOST serial port as connected with the onboard PC or other accessory can be used for automated PC shutdown or other utilities: Pin 4 (DSR) is RS-232 high when the microcontroller operates normally; otherwise it is low when reset or in maintenance mode. Similarly, pin 9 (RI) normally is low and goes RS-232 high when the robot's batteries drop below a microcontroller FLASH `ShutdownSOC` parameter charge level.

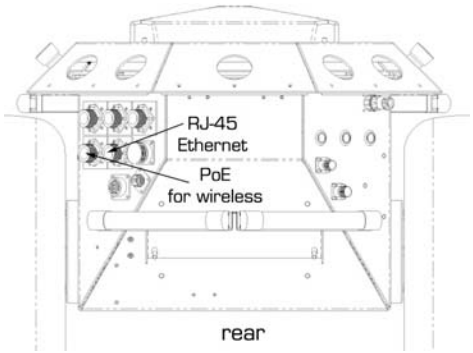


Figure 9. RJ-45 Ethernet port to the systems inside and Power-over-Ethernet port for the wireless accessory (top bumpers not shown for clarity)

Operating the Onboard SBCs

This is a brief overview of operating the onboard SBCs. Please consult the *Computer Systems Documentation* and the OS manufacturer's documentation for more detail.

MOBILEROBOTS software runs on either Windows or Debian Linux. Accordingly, we prefer and support those operating systems on the onboard SBC(s). When we perform the installation and configuration, we put the robotics and accessory software typically in `/usr/local` on Linux systems or in `C:\Program Files\MobileRobots` under Windows. Of course, we install the appropriate drivers for the various accessory devices.

The onboard SBCs power ON when Seekur Jr's main power is ON. The systems come with two default users: one with common systems and file read/write permissions (`'guest'`) and one with full-access to the PC software and OS—`root` (Linux) or `'Administrator'` (Windows), with no passwords. When connected directly, we recommend you log in with full-access capabilities so that you can do systems set up and maintenance, such as change passwords, add users and set up the network. Do note that with Linux systems, you cannot log in remotely over the network as `root`; you must log in as a common user and use the `'su -'` command thereafter to attain superuser (`'root'` login) status.

Once logged into a Windows system, it's simply a matter of clicking the mouse to select programs and applications. With Linux, use the `'startx'` command to enable the X-Windows desktop and GUI environment, only as needed, for instance to perform some of the *Quick Start* activities.

PC Networking

An RJ-45 connector on the rear left panel of Seekur Jr provides wired 10/100Base-T Ethernet networking to an onboard Ethernet switch and, hence, access to all the onboard, networked systems, such as the primary and secondary SBCs, wireless Ethernet and the laser range finders.

The installed SBCs are preset and tested at fixed IP addresses with Class-C network configuration. The primary SBC uses addresses `10.0.125.32` and `.33`; the second SBC is addressed `10.0.125.34` and `.35`. Before reconfiguring these network settings, please consult with your network systems administrator for networking details.

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

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

Please note that, with the primary SBC and wireless Ethernet, as opposed to the wireless Ethernet-to-serial device, you may ***not*** connect with the robot's microcontroller directly over the network: That is, you cannot run a client application like the ARIA `demo` on a remote PC and

choose to directly connect with the robot server by selecting the robot SBC's IP address. Rather, either run the client application on the primary SBC and export the display and controls over the network to the remote PC (preferred), or use the ARIA-based `IPTHRU` programs (see program sources in `Aria/examples`) to negotiate the IP-to-serial conversions needed by the client-server connection.

UPS and Genpowerd

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

SeekurOS raises the HOST serial port's DSR pin 6 to RS-232 high and puts the RI pin 9 to low when the microcontroller is operating normally and your robot's battery power is above `ShutdownSOC` (default is 5% SOC).[4] The RI pin goes high when power drops consistently below `ShutdownSOC`.

Genpowerd running on the onboard Linux system[5] detects the change of state and initiates OS shutdown after a short wait, during which the shutdown may be canceled by raising the battery charge level, such as by attaching the external Power/Recharging Accessory supply or by adding or hot-swapping a fresh battery pack. *Genpowerd* monitors the HOST serial RI port on `/dev/ttyS0`.

[4] RI and DSR on the HOST serial port are RS-232 low during startup or reset.
[5] Windows no longer supports UPS.EXE.

Chapter 5 Quick Start

PREPARATIVE HARDWARE ASSEMBLY

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

Install Batteries

Out of the box, your Seekur Jr comes with its batteries fully charged, although shipped separately. Slide at least one and up to three batteries into the robot's battery box through the top hatch. See *Chapter 3* for details.

Client-Server Communications

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

- ✓ A tether cable from the robot's HOST serial connector to a computer, typically a piggyback laptop
- ✓ Wireless serial Ethernet
- ✓ Radio Modem
- ✓ An integrated onboard SBC wired internally for direct onboard control

DEMO CLIENT

MOBILEROBOTS' robot software development environment, ARIA, comes with many demonstration software. ARIA's best is `demo` since it also serves as a way to test all your robot's features and accessories.

Your MOBILEROBOTS platform also may come with LRF- and/or GPS-based advanced robotics localization and navigation hardware and software, including two GUI networked-client application, MobileEyes and Mapper3. MOBILEROBOTS customers also may obtain these and related software and updates from our support website:

`http://robots.MobileRobots.com`

Please consult the separate *mOGS* or *ARNL Installation and Operations Manual* for details.

STARTING UP ARIA DEMO

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

First, please note well that you cannot connect with and control your robot through its microcontroller directly from a remote client over the network without special hardware (Ethernet-to-serial device) or, alternatively, special software that runs on the onboard SBC and converts IP packets into serial data.[6] Otherwise, you must run the client software on the robot's SBC or on a computer that is connected to the robot's microcontroller `HOST` serial port. You may, of course, export the controls and display of your onboard SBC over the network from X-windows or with special Windows software, such as VNCserver or XWin.

If you are using a wireless Ethernet-to-serial device to communicate with the robot's microcontroller from a networked PC, you might test your connection—either peer-to-peer or managed through an access point on your LAN—with the common `ping` program.

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

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

```
% ./demo
```

Demo Startup Options

By default, the ARIA `demo` program connects with the robot through the serial port `COM1` under Windows or `/dev/ttyS0` under Linux. And, by default, `demo` connects with an attached LRF accessory through serial port `COM3` or `/dev/ttyS2`. To change those connection options, either modify the ARIA source code (`examples/demo.cpp` and

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

related files in `src/`) and recompile the application, or use a startup argument on the command line (Table 2).

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

```
C:\Program Files\MobileRobots\ARIA\bin\demo.exe
                  -remoteHost 192.168.1.32
```

Table 2. ARIA demo's robot connection start-up options

-remoteHost <Host Name or IP> (abbreviated -rh)	Connect with robot through a remote host over the network instead of a serial port; requires serial-to-Ethernet device or IPTHRU software.
-robotPort <Serial Port> (abbreviated -rp)	Connect with robot through specified serial port name; COM3, for example. COM1 or /dev/ttyS0 is the default.
-robotBaud <baudrate> (abbreviated -rb)	Connect with robot using the specified baudrate; 19200 or 38400, for example. Default is 9600.
-remoteRobotTcpPort <Number> (abbreviated -rrtp)	Remote TCP host-to-robot connection port number; default is 8101.

A Successful Connection

ARIA prints out lots of diagnostic text as it negotiates a connection with the robot. If successful, the client requests various SeekurOS servers to start their activities, including position integration and so on. Note that the ARIA demo automatically engages your robot's motors though the special client command #4 with argument one. Normally, the motors are disengaged when first connecting.

Operating the ARIA Demonstration Client

When connected with the ARIA demo client, your robot becomes responsive and intelligent. For example, it moves cautiously if it has an LRF accessory, so that although it may drive toward an obstacle, your robot will not crash because the ARIA demo includes obstacle-avoidance behaviors which enable the robot to detect and actively avoid collisions.

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

Table 3. Keyboard teleoperation

KEY	ACTION
↑	forward
↓	reverse
←	turn left
→	turn right
space	all stop

While driving from the keyboard, hold down the respective keys to simultaneously drive the robot forward or backward and turn right or left. For instance, hold down the up-arrow key to have the robot accelerate forward to its cruising speed of around 400 millimeters per second (defined in the source code). Release the arrow key to have the robot slow down and stop. Press and hold the right- or left-arrow key to have the robot rotate or turn in an arc if you also hold down the up- or down-arrow key.

The other modes of ARIA demo operation give you access to your robot's various sensors and accessories. Accordingly, use the ARIA demo not only as a demonstration tool, but as a diagnostic one, as well, if you suspect a sensor or effector has failed or is working poorly.

Access each ARIA demo mode by pressing its related hot-key: '`t`', for instance, to select `teleoperation`. Each mode includes onscreen

instructions and may have sub-menus for operating of the respective device.

Table 4. ARIA demo operation modes

MODE	HOT KEY	DESCRIPTION
laser	l	Displays the closest and furthest readings from the laser range finder
io	i	Displays the digital and analog-to-digital I/O ports
position	p	Displays the coordinates of the robot's position relative to its starting location
bumps	b	Displays bumpers status
camera	c	Controls and exercises the pan-tilt-zoom robotic camera
wander	w	Sends the robot to move around at its own whim while avoiding obstacles
teleop	t	Drive and steer the robot via the keyboard or a joystick; avoids collisions
unguarded	u	Same as teleop, except no collision avoidance
direct	d	Direct command mode

DISCONNECTING

When you finish, press the `Esc` key to disconnect the ARIA client from your robot server and exit the ARIA demonstration program. Your robot should disengage its drive motors and stop moving.

NETWORKING WITH MOBILEEYES

To use the MobileEyes GUI client for much more advanced, network-based robotics control, you need to do things a bit differently. You need the Ethernet-to-serial device or an SBC on the robot. But instead of ARIA `demo`, use ArNetworking's `serverDemo`, `also found in ARIA/bin`, to mediate communications between MobileEyes and your robot over the network.

Start serverDemo

The ArNetworking client, `serverDemo`, works to mediate communications between your MOBILEROBOTS platform over the network and remote appplications, such as the GUI MobileEyes. To examine its inner workings, look at `serverDemo.cpp` (and others) sources in the `ARNetworking/examples` subdirectory.

Start `serverDemo` just like `demo`, with the `-remoteHost` argument if you aren't running the software onboard. ServerDemo also accepts the special command-line argument –connectLaser if you have an LRF attached to an onboard `serverDemo`-based SBC. ServerDemo can be run anywhere on the network from which your MobileEyes and your robot have access, such as on the same SBC as you run MobileEyes.

Connect MobileEyes with serverDemo, not the robot's microcontroller.

Start MobileEyes and Connect with serverDemo

From Windows, simply double-click the MobileEyes.exe program normally located in the `C:\Program Files\MobileRobots\MobileEyes\bin` directory. With Linux, you need to have started X and, from a terminal session, navigate to the `MobileEyes/bin` directory in `/usr/local` and execute it:

```
./MobileEyes &
```

In the MobileEyes startup dialog, enter the hostname or IP address of the SBC on which you are running `serverDemo`.

Operating MobileEyes

MobileEyes is not that interesting when run with just `serverDemo`. Simply click the teleop icon on the menu bar to engage manual drive and operate the robot as you did with the ARIA demo. Your robot with MobileEyes really comes to life when used with mOGS or ARNL for map-based path-planning and navigation. Nonetheless, this demonstration presents the networking, control and programming concepts inherent in ArNetworking.

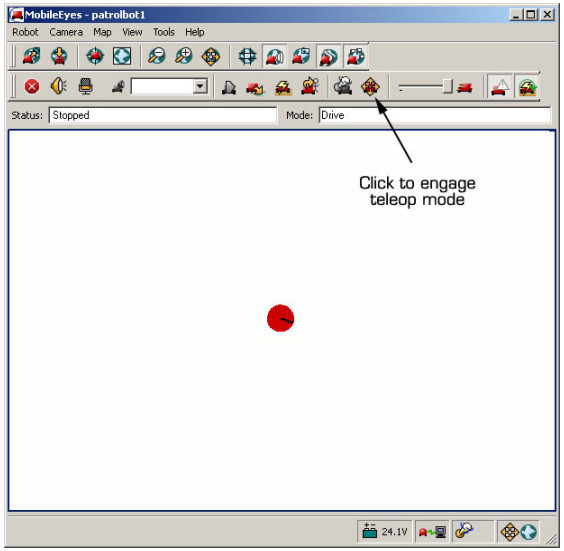


Figure 10. Very simple `serverDemo`-mediated connection between your MOBILEROBOTS platform and MobileEyes. MobileEyes is a terrific GUI-based robotics command and control client, but works best when running with mOGS or ARNL in a mapped space.

QUICKSTART TROUBLESHOOTING

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

Parameter Files

The most common mistake is not having your robot's "parameter" file located in the `Aria/params` directory. For Seekur Jr, the ARIA-parameter filename is `seekurjr.p`. This and all other MobileRobots ARIA-based parameter files either come with the latest version of ARIA with ArNetworking or can be retrieved from the `http://robots.MobileRobots.com` support website. Note that ARIA itself contains parameter defaults in case the parameter file is missing. And you can make up your own robot parameter files, too.

Proper Connections

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

Make sure your robot's batteries are fully charged. The robot servers shut down and won't allow a connection at under `ShutdownSOC` (default is 5%).

If you are using the onboard SBC or Ethernet-to-serial radios, the serial connection is internal and established at the factory; you should not have problems with those cables.

ATTENTION!

The demo/serverDemo-to-robot connection is *SERIAL only*. Accordingly, run them on the onboard or piggyback computer, over radio modems or over the network with a wireless Ethernet-to-serial device.

With other serial connections, make sure to use the proper cable: a "pass-through" one, minimally connecting pins 2, 3, and 5 of your computer's serial port to their respective contacts of the robot's serial port on the User Control Panel (assuming no onboard SBC).

If you access the wrong serial port, the demonstration program will display an error message. If the robot server isn't listening or if the serial link is severed somewhere between the client and server (cable loose or the radio is off, for instance), the client will attempt "Syncing 0" several times and fail. In that case, `RESET` the robot and check your serial connections.

If for some reason communications get severed between the client and SeekurOS server, but both the client and server remain active, you may revive the connection with little effort. If you are using wireless communications, first check and see if the robot is out of range.

Communications also will fail if the client and/or server is somehow disabled during a session. For instance, if you inadvertently switch off the robot's `Main Power` or press an e-stop button, you must restart the robot and the connection. If the demo or other client application is still active, simply press `esc` and restart.

Motor Power

Still can't get Seekur Jr to move? Check the MOTORS status: the blue MOTORS button should be ON solid. If the motors have power, but have not been enabled by the software (client command #4), the LED double-blinks. If you haven't manually pressed the MOTORS button after startup, the LED blinks.

Chapter 6 SeekurOS

All MOBILEROBOTS platforms use a client-server mobile robot control architecture. In the model, the robot's microcontroller works to manage all the low-level details of the mobile robot's systems. These include operating the motors, collecting and reporting odometry and so on—all on command from and reporting to a separate client application, such as the ARIA demo.

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

Experienced MOBILEROBOTS users can be assured that SeekurOS is upwardly compatible with all MOBILEROBOTS platforms, implementing the same commands and information packets that first appeared in the Pioneer 1-based PSOS, in the original Pioneer 2-based P2OS, and more recent ARCOS-based Pioneer 3s, as well as PeopleBot and PowerBot.

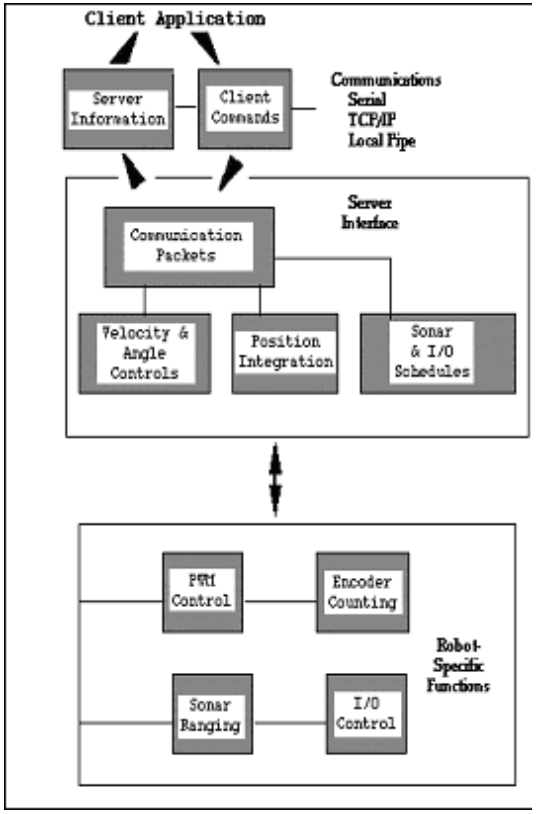


Figure 11. MOBILEROBOTS client-server control architecture

CLIENT-SERVER COMMUNICATION PACKET PROTOCOLS

MOBILEROBOTS platforms communicate with a client using special client-server communication packet protocols, one for command packets from client to server and another for Server Information Packets (SIPs) from the server to client. Both protocols are byte streams consisting of five main elements: a two-byte header, a one-byte count of the number of subsequent packet bytes, the client command or SIP packet type, command data types and argument values or SIP data bytes and, finally, a two-byte checksum. Packets are limited to a maximum of 207 bytes each.

The two-byte header which signals the start of a packet is the same for both client command packets and SIPs: 0xFA (250) followed by 0xFB (251). The subsequent count byte is the number of all *subsequent* bytes in the packet including the checksum, but not including the byte count value itself or the header bytes.

Data types are simple and depend on the element (see descriptions below): client commands, SIP types, and so on, are single 8-bit bytes, for example. Command arguments and SIP values may be 2-byte integers, ordered as least-significant byte first. Some data are strings of up to a maximum 200 bytes, prefaced by a length byte. Unlike common data integers, the two-byte checksum appears with its *most-significant* byte first.

Packet Checksum

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

NOTE: The checksum integer is placed at the end of the packet, with its bytes in the reverse order of that used for data; that is, b_0 is the high byte and b_1 is the low byte.

```
AREXPORT ArTypes::Byte2 ArRobotPacket::calcCheckSum(void)
{
  int i;
  unsigned char n;
  int c = 0;

  i = 3;
  n = myBuf[2] - 2;
  while (n > 1) {
    c += ((unsigned char)myBuf[i]<<8) |
         (unsigned char)myBuf[i+1];
    c = c & 0xffff;
    n -= 2;
    i += 2;
  }
  if (n > 0)
    c = c ^ (int)((unsigned char) myBuf[i]);
  return c;
}
(from MobileRobots ARIA ArRobotPacket.cpp)
```

Packet Errors

SeekurOS ignores a client command packet whose byte count exceeds 204 (total packet size of 207 bytes) or has an erroneous checksum. The client should similarly ignore erroneous SIPs.

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

Nonetheless, the client-server interface provides a simple means for dealing with ignored command packets: Most of the client commands alter state variables in the server. By examining those values in respective SIPs, client software may detect ignored commands and re-issue them until achieving the correct state.

The Client-Server Connection

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

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

Specifically, and as examples of the client command protocol described below, the sequence of synchronization bytes is:

```
SYNC0:  250, 251, 3, 0, 0, 0
SYNC1:  250, 251, 3, 1, 0, 1
SYNC2:  250, 251, 3, 2, 0, 2
```

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

Table 5. Client command packet protocol

Component	Bytes	Value	Description
header	2	0xFA, 0xFB	Packet header; same for client and server
byte count	1	N	Number of command/argument bytes plus checksum, but not including Byte Count or the header bytes. Maximum of 249.
command number	1	0 - 255	Client command number; see Table 7.
argument type	1	0x3B or 0x1B or 0x2B	Required data type of command argument, in order: positive integer, negative or absolute integer, or string
argument	n	data	Command argument; always 2-byte integer or string containing length prefix
checksum	2	computed	Packet integrity checksum

Autoconfiguration (SYNC2)

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

The type string typically is Pioneer. The subtype depends on your robot model; SeekurJr in this case. Clients may use these identifying strings to self-configure their own operating parameters. ARIA, for instance, loads and uses the robot's related parameter file found in the special Aria/params directory ("seekurjr.p").

Opening the Servers—OPEN

Once you've established a connection with SeekurOS, your client should send the OPEN command #1 (250, 251, 3, 1, 0, 1) to the server, which causes the microcontroller to perform a few housekeeping functions, start its various servers and begin transmitting server information to the client.

Note that when at first connected, Seeur Jr's motors are disabled regardless of their state when last connected. To enable the motors after starting a connection, you must have your client send an ENABLE client command #4 with an integer argument of 1 (see *Client Commands* below). This also assumes that you also had manually engaged power to the motors by pressing the blue MOTORS button on start up.

Press the blue MOTORS button on the User Control Panel to enable Seekur Jr's motors power.

Server Information Packets

Once OPENed, SeekurOS automatically and repeatedly sends a standard packet of information over the HOST serial port back to the connected client exactly every SIPcycle = 100 milliseconds. The standard SeekurOS SIP informs the client about a number of operating states and readings, using the order and data types described in the nearby Table. SeekurOS also supports several additional SIP types. See following sections for details.

Keeping the Beat—PULSE

A SeekurOS safety watchdog expects that, once connected, your Seekur Jr receives at least one command packet from the client every two seconds. Otherwise, it assumes the client-server connection is broken and stops the robot.

Some clients—ARIA-based ones, for instance—use the good practice of sending a PULSE client command #0 (250, 251, 3, 0, 0, 0) just after OPEN. And if your client application will be otherwise distracted for some time, periodically issue the PULSE command to let your robot server know that your client is indeed alive and well. It has no other effect.

If the robot shuts down due to lack of communication with the client, it will revive upon receipt of a client command and automatically accelerate to the last-specified speed and heading setpoints.

Closing the Connection—CLOSE

To close the client-server connection, which automatically disables the motors and other functions, simply issue the client CLOSE command #2 with no argument (250, 251, 3, 2, 0, 2).

Most of SeekurOS's operating parameters return to their FLASH-based default values upon disconnection with the client.

Client Commands

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

The number of client commands you may send per second depends on the HOST serial baud rate, average number of data bytes per command, synchronicity of the communication link and so on. SeekurOS's command processor runs on a less-than-one-millisecond interrupt cycle, but the server response speed depends on the command. Typically, limit client commands to a maximum of one every 3-5 milliseconds or be prepared to recover from lost commands.

Table 6. SeekurOS standard SIP contents

LABEL	DATA	DESCRIPTION
HEADER	2 bytes	Exactly in order 0xFA (250), 0xFB (251)
BYTE COUNT	byte	Number of data bytes + 2 (checksum), not including header or byte-count bytes
TYPE	0x3s	Motors status; s = 2 when motors stopped or 3 when robot moving.
XPOS	int	Wheel-encoder and gyro-corrected integrated coordinates in millimeters (DistConvFactor‡ = 1.0)
YPOS	int	
THPOS	int	Orientation in angular units (AngleConvFactor‡ = 0.001534 radians per angular unit)
L VEL	int	Wheel velocities in millimeters per second (VelConvFactor‡ = 1.0)
R VEL	int	
BATTERY	byte	Battery charge in 0.2V increments (101 = 20.2 volts, for example)
STALL AND BUMPERS	uint╪	Motor stall and bumper indicators; b_0 is the left-wheel stall indicator, set to 1 if stalled; b_{1-4} correspond to the front bumper segments; b_8 is the right-wheel stall; and b_{9-12} correspond with the rear bumper segments.
NU	int	Not used (always 0)
FLAGS	uint	States 1 when active: b_0 is motors status; b_5 is E-STOP; b_9 is the joystick GO button; b_{10} is battery charger power-good.
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
NU	byte	Not used (always 0)
BATTERYX10	int	Actual battery voltage in 0.1 V (especially useful for battery voltages > 25.5)
CHARGESTATE	byte	Batteries recharging state byte: -1=unknown; 0=not charging; 1=regeneration; 2=fast; 3=floating; 4=balance
ROTVEL	int	Current rotational velocity in degrees X 10 per sec
FAULTFLAGS	int	1 when active: b_0 when LRF inadvertently shut down; b_3 when PC shutting down due to low-charge on battery; b_{15} when joydrive is UNSAFE
LATVEL	int	Lateral velocity (mm/sec)
NU	int	Not used (always 0)
CHECKSUM	int	Packet-integrity checksum

‡ Client-side data-conversion factor. Consult the ARIA parameter file for your robot.

╪ Explicitly, an unsigned integer; all others sign-extended

Table 7. SeekurOS client-side command set

COMMAND	#	ARGS	DESCRIPTION
	Before Client Connection		
SYNC0	0	none	Start connection. Send in sequence. SeekurOS echoes
SYNC1	1	none	synchronization commands back to client, and
SYNC2	2	none	robot-specific auto-synchronization after SYNC2.
	After Client Connection		
PULSE	0	none	Reset server watchdog.
OPEN	1	none	Start up servers.
CLOSE	2	none	Close servers and client connection.
ENABLE	4	int	1=enable; 0=disable the motors.
SETA	5	int	Set translation acceleration, if positive, or deceleration, if negative; in mm/sec2.
SETV	6	int	Set maximum/move translation velocity; mm/sec.
SETO	7	none	Reset local position to 0,0,0 origin.
ROTATE	9	int	Rotate (+) counter- or (-) clockwise degrees/sec at SETRV limited speed.
SETRV	10	int	Sets maximum/turn rotation velocity; degrees/sec.
VEL	11	int	Translate at mm/sec forward (+) or backward (-) (SETV limited speed).
HEAD	12	int	Turn at SETRV speed to absolute heading; ±degrees (+ = counterclockwise).
DHEAD	13	int	Turn at SETRV speed relative to current heading; (+) counter- or (–) clockwise degrees.
JOYREQUEST	17	int	Request one or continuous stream (>1) or stop (0) joystick SIPs
CONFIG	18	none	Request a configuration SIP.
ROTVELMAXDIR	20	int	Positive sign (+) sets CCW max rotational velocity (degrees per second); - sets CW max rot vel; 0 resets both to default rotVelMax; 1 or -1 means cannot move in that direction.
RVEL	21	int	Rotate at (+) CCW or (–) CW degrees/sec (SETRV limit).
DCHEAD	22	int	Adjust heading relative to last setpoint; ± degrees (+ = ccw)
SETRA	23	int	Change rotation de(-) or (+)acceleration, in degrees/sec2
IMU	26	int	Request one (1) or continuous stream (>1) or stop (0) IMUpac SIPs
STOP	29	none	Stop the robot; motors remain enabled
BUMPSTALL	44	int	0=disable; 1=front only; 2=rear only; 3=front and rear bumpstalls engaged
JOYDRIVE	47	int	1=allow joystick drive from port while connected with a client; 0 (default) disallows.
E_STOP	55	none	Emergency stop; very abrupt by overriding deceleration.
LRFPOWER	96	int	Start up (argument 1) or shut down (0) LRF power
LATVEL	110	int	Translate millimeters/sec to the left or, if negative value, to the right (Seekur only)
LATACCEL	113	int	Lateral acceleration (millimeters per second per second) or, if negative value, deceleration (Seekur only).
SEEKURPOWER	116	2-byte	First byte is power port #; second byte is state: 0=OFF, 1=ON
SEEKUROFF	119	none	Turn off main power from software.

CNTRWHEELS	120	none	Recenter wheels (Seekur only)
AUXPCPOWER	125	int	Power control for the second PC; 0=OFF or 1=ON
PTZPOWER	127	int	Power control for the camera accessory; 0=OFF, 1=ON
LRF2POWER	129	int	Cycle power to the second LRF; 0=OFF, 1=ON.
BATTEST	250	int	Artificially set the state-of-charge battery; argument in percent remaining; 0 to revert to real
DEBUG	251	int	Request debug packets; 1=send each SIPcycle; 0=stop sending
RESET	253	none	Force a power on-like reset of the microcontroller.
MAINTENANCE	255	none	Engage microcontroller maintenance (ARSHstub) mode.

ROBOTS IN MOTION

The SeekurOS motor-control servers accept either translation or rotation movements independently and, upon receipt, immediately switches to the new setpoint. Note that once connected, Seekur's motors are *disabled*, regardless of their state when last connected. Accordingly, you must enable the motors with the ENABLE client command #4 with the argument value of one.[7] Monitor the status of the motors with bit 0 of the Flags integer in the standard SIP.

When SeekurOS receives a motion command, it accelerates or decelerates the robot at the translation SETA (command #5), rotation SETRA (command #23) or lateral LATACCEL (command #113) rates until the platform either achieves its SETV (command #6) maximum translation, LATVEL (command #110) lateral or SETRV (command #10) maximum rotation speeds or nears its goal. Accordingly, rotation headings and translation setpoints are achieved by a trapezoidal velocity function, which SeekurOS recomputes each time it receives a new motion command.

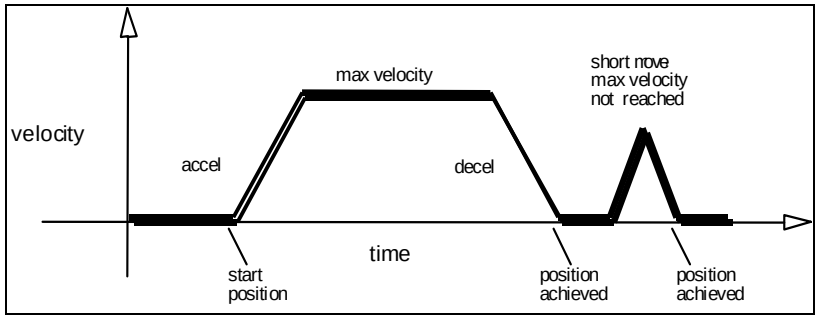


Figure 12. Trapezoidal velocity profile

SeekurOS automatically limits VEL-, LATVEL- and RVEL-specified velocities to previously imposed, client-modifiable SETVEL and SETRV maximums, and ultimately by absolute, platform-dependent, embedded constants (TOP values). Similarly, the distinct acceleration and deceleration parameters for both translation and rotation are limited by embedded constants. SeekurOS initializes these values from embedded constants upon microcontroller startup and reset, and when first connecting with a client. The speed limits, either defaults or when changed by related client commands take effect on subsequent commands, not for current translation or rotation. The maximums revert to their defaults when disconnected from a client.

Table 8. Client motion-related commands

Rotation	
HEAD (#12)	Turn to absolute heading at SETRV max rotational velocity
DHEAD (#13), DCHEAD (#22)	Turn to heading relative to control point at SETRV max velocity
ROTATE (#9)	Rotate at SETRV velocity
RVEL (#21)	Rotate at + (counter) or – (clockwise) deg/sec up to SETRV max
Translation	
VEL (#11)	Translate forward/reverse at SETV prescribed velocity
LATVEL (#110)	Move sideways right (+) or left (-) (Seekur only)

Send an orientation command HEAD (#12), DHEAD (#13), or DCHEAD (#22) with argument value in degrees to turn the robot with respect to its internal dead-reckoned angle to an absolute heading (±0-180 degrees), relative to its immediate heading, or relative to its current heading setpoint (achieved or last commanded heading), respectively. In general, positive relative heading command arguments turn the robot in a counterclockwise direction. However, the robot always turns in the direction that will achieve its heading most efficiently. Accordingly, relative-heading arguments greater than 180 degrees automatically get reduced to 180 or less degrees with a concomitant change in direction of rotation.

The E_STOP command #55 overrides normal deceleration and abruptly stops the robot in the shortest distance and time possible.

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

Position Integration

MOBILEROBOTS platforms track their position and orientation based on dead-reckoning from wheel motion derived from encoder readings and from the integrated gyroscope. The SeekurOS-based robot maintains its internal coordinate position in platform-dependent units, but reports the values in platform-independent millimeters and angular units (2pi/4096 radians) in the standard SIP (Xpos, Ypos, and Thpos).

Be aware that 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 a few meters and one or two revolutions, depending on the surface.

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.

On start-up, the robot is at the origin (0, 0, 0), pointing along the positive X-axis at 0 degrees. Absolute angles vary between 0 and ±4096 angular units (+180 to -179 degrees).

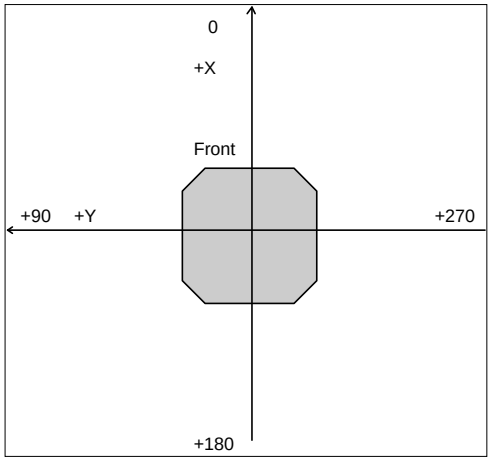


Figure 13. Internal coordinate system

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

STALLS AND EMERGENCIES

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

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

ACCESSORY COMMANDS AND PACKETS

Other server information packets (SIPs) come with SeekurOS to better support the MOBILEROBOTS community. On request from the client by a related SeekurOS command, the SeekurOS server packages and sends one or a continuous stream of information packets to the client over the HOST serial communication line. Extended packets get sent immediately before (such as `GYROpac` and `JOYSTICKpac`) or after (such as `CONFIGpac`) the standard SIP that SeekurOS sends to your client every `SipCycle` (100) milliseconds.[8]

The standard SIP takes priority so you may have to adjust the `HOST` serial baud rate to accommodate all data packets in the allotted cycle time, or some packets may never get sent.

Packet Processing

Identical with the standard SIP, all SeekurOS server information packets get encapsulated with the header (0xFA, 0xFB; 250, 251), byte count, packet type byte and trailing checksum. It is up to the client to parse the packets, sorted by type for content. Please consult the respective client application programming manuals for details.

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

Joystick

Use the SeekurOS client JOYREQUEST command number 17 with an argument value of zero, one or two to request information about the joystick, if it is enabled (see next Chapter). The argument value one requests a single packet (type = 248; 0xF8) to be sent by the next client-server communications cycle. The request argument value of two tells SeekurOS to send JOYSTICKpac packets continuously, at approximately one per cycle depending on serial port speed and other pending SIPs. Use the JOYREQUEST argument value zero to stop continuous JOYSTICKpac packets.

Table 10. JOYSTICKpac contents

LABEL	BYTES	VALUE	DESCRIPTION
header	2	0xFA, 0xFB	Common header
Byte count	1	11	Varies
type	1	0xF8	Packet type
button0	1	0 or 1	1=button pressed
button1	1	0 or 1	1=button pressed
X-axis	2	varies 0-1023	Rotation analog
Y-axis	2	varies 0-1023	Translation analog
throttle	2	varies 0-1023	Throttle setting
checksum	2	varies	Computed checksum

CONFIGpac and CONFIG Command

Send the CONFIG command #18 without an argument to have SeekurOS send back a CONFIGpac SIP packet type 32 (0x20) containing the robot's operational parameters. Use the CONFIGpac to examine many of your robot's default FLASH_based settings and their working values, where appropriate, as changed by other client commands, such as SETV and ROTKV.

Table 11. CONFIGpac contents

LABEL	BYTES	VALUE	DESCRIPTION
Header	2	0xFA, FB	Common packet header = 0xFAFB
Byte count	1		Number of following data bytes
Packet type	1		ENCODERpac = 0x20
Robot type	str		Typically "Pioneer"
Subtype	str		Identifies the MobileRobots model; e.g. "p3dx-sh",
Sernum	str		Serial number for the robot.
NU	1	0	Antiquated 4Mots
Rotveltop	2		Maximum rotation velocity; deg/sec
Transveltop	2		Maximum translation speed; mm/sec
Rotacctop	2		Maximum rotation (de)acceleration; deg/sec^2
Transacctop	2		Maximum translation (de)acceleration; mm/sec^2
PWMmax	2	500	Maximum motor PWM (limit is 500).
Name	str		Unique name given to your robot.
SIPcycle	1	100	Server information packet cycle time; ms.
Hostbaud	1	0	Baud rate for client-server HOST serial: 0=9.6k, 1=19.2k, 2=38.4k, 3=56.8k, 4=115.2k.
Auxbaud	1	0	Baud rate for AUX1 serial port; see HostBaud.
NU	2	0	Pioneer Gripper
NU	2	0	Pioneer front sonar
NU	1	0	Pioneer rear sonar
Lowbattery	2	230	In 1/10 volts; alarm activated when battery charge falls below this value.
NU	2	0	Pioneer revcount
Watchdog	2	2000	Ms time before robot automatically stops if it has not received a command from the client. Restarts on restoration of connection.
NU	1		Antiquated p2mpacs
Stallval	2		Maximum PWM before stall. If > PWMMAX, never.
Stallcount	3		Ms time after a stall for recovery. Motors lax during this time.
Joyvel	2		Joystick translation velocity setting, mm/sec
Joyrvel	2		Joystick rotation velocity setting in deg/sec
Rotvelmax	2		Current max rotation speed; deg/sec.
Transvelmax	2		Current max translation speed; mm/sec.
Rotacc	2		Current rotation acceleration; deg/ sec^2
Rotdecel	2		Current rotation deceleration; deg/ sec^2
Rotkp	2		Current Proportional PID for rotation
Rotkv	2		Current Derivative PID for rotation

Rotki	2		Current Integral PID for rotation
Transacc	2		Current translation acceleration; mm/ sec^2
Transdecel	2		Current translation deceleration; mm/ sec^2
Transkp	2		Current Proportional PID for translation.
Transkv	2		Current Derivative PID for translation.
Transki	2		Current Integral PID for translation.
Frontbumps	1	4	Number of front bumper segments.
Rearbumps	1	4	Number of rear bumper segments.
Charger	0		0 if no automated charging
NU	1	0	Pioneer sonarcycle
Autobaud	1	0	1 if the client can change HOST baud rates; 2 if auto-baud implemented.
HasGyro	1	3	Gyro type: 0=none; 3=SAG; 4=IMU
NU	2	0	Pioneer driftfactor
NU	1	0	Pioneer Aux2baud rate
NU	1	0	Pioneer Aux3baud rate
NU	2	0	Pioneer ticksmm
Shutdownvolts	2	215	DC volts X10 at or below which the onboard PC will shut down if Pb-acid batteries
Versionmajor Versionminor	str str	1.1	Null-terminated strings for SeekurOS version numbers
GyroCW	2		Gyro calibration factor clockwise
GyroCCW	2		Gyro calibration factor counterclockwise
KinematicsDelay	1	25	Kinematics latency in milliseconds
LatVelTop	2		Seekur only; absolute max lateral velocity
LatAccTop	2		Seekur only; absolute max lateral acceleration
LatVelMax	2		Seekur only; current lateral max velocity
LatAcc	2		Seekur only; current lateral max acceleration
LatDec	2		Seekur only; current lateral max deacceleration
NU	2	0	PowerBot only charge threshold
PowerCommands	2	0x2A2	Bit pattern related to which power ports are assigned
BatteryType	1	2	Battery type 0,1=Pb-acid; 2=NiMH
LowSOC	2	10	Remaining state-of-charge percent to prompt for a recharge
ShutdownSOC	2	5	Remaining state-of-charge at or below which to auto-shutdown the robot

GYROSCOPE AND INERTIAL MEASUREMENT UNIT

Seekur Jr comes with either a single-axis gyroscope (SAG; `CONFIGpac HasGyro` = 3) or an inertial measurement unit (IMU; `HasGyro` = 4) attached to the microcontroller. Each device axis has an associated temperature sensor. The SAG or the IMU's Z-axis gyro is used by SeekurOS to compute the platform's heading and corrects the Th term in the standard SIPs odometry values.

The SAG and IMU's `gyroRanges` are adjustable, but currently set to 160 or 150 degrees per second, respectively, at the factory.

Clients may retrieve SAG or IMU readings from the `IMUpac` SIP. Use client command #26 with argument one to retrieve a single or argument two to have a stream of IMUpacs (type=154) SIPs sent after the standard SIP.

For synchronization, a `TimeTo` value measures the time in milliseconds from when the standard SIP started transmitting to when the latest average had been taken by SeekurOS.

Please consult Analog Devices' specification sheets for the SAG (ADIS16255) and the IMU (ADIS16355). SeekurOS retrieves pertinent values over an internal SPI bus.

Table 12. IMUpac contents

Label	Bytes	Value	Description
header	2	0xFA, 0xFB	Common header
Byte count	1	varies	Varies
type	1	0x9A	Packet type
TimeTo	1	varies	Milliseconds from std SIP transmit to last reading
nAverages	1	varies	Varies; typically 4 per 100ms SIPcycle
nGyroAxes	1	1 or 3	1=SAG; 3=IMU
Repeat for nAverages:			
gyroRange	1	1, 2 or 3	1=80/75, 2=160/150; 3=320/300 SAG/IMU deg/sec
gyroAvgX, gyroAvgY, gyroAvgZ	2 ea	varies	Each varies; gyroAvgZ only if SAG
nAccelAxes	0	0 or 3	0=SAG; 3=IMU
Repeat for nAverages:			
avgAccelX, avgAccelY, avgAccelZ	2 ea	varies	Accelerometer averages
nTempAxes	1	1 or 3	1=SAG; 3=IMU
Repeat for nAverages:			
TempX, TempY, TempZ	2 ea	varies	Temperature readings
Checksum	2	varies	

Chapter 7 Updating & Reconfiguring SeekurOS

The SeekurOS firmware and its set of operating parameters get stored in your Seekur's microcontroller FLASH memory. With special download and configuration software tools, you may change the parameters and update SeekurOS. No hardware modification is required.

WHERE TO GET SEEKUROS SOFTWARE

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

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

SEEKUROS DOWNLOADER

SeekurOS is distributed as an s19-encoded file and comes with a set of associated download files and programs, Windows only. The distribution archive is a self-extracting file, SeekurOSV_v, where V and v are the major and minor version numbers. Extracting puts the contents in C:\Program Files\MobileRobots\SeekurOS.

The distribution includes the SeekurOS parameter management program, too, and places its contents in the C:\Program Files\MobileRobots\SeekurOSParamsManager directory. See below for details.

Use the supplied
Maintenance Serial Cable
for SeekurOS updates and the supplied
HOST Serial Cable
for changing parameter values.

To update the SeekurOS image on your robot's microcontroller, first make sure that there are no other connections, such as from an ARIA client. With the supplied serial cable, attach your Windows PC or laptop serial port to your robot's external Maintenance Serial.

Note that if you don't have an onboard SBC, a HOST serial port shares the Maintenance serial port's connector on the rear left panel of Seekur Jr, but their supplied serial cables are different.

Simply run the download.bat program from the SeekurOS directory.

SEEKUROS PARAMETERS MANAGER

The SeekurOS operating parameters manager, `Param-ManagerStatic`, runs either on Linux or Windows and like other ARIA clients communicates with the microcontroller through the HOST serial port. If you have an onboard SBC, run `ParamManagerStatic` from it. Otherwise, connect through the external HOST serial port located on the Maintenance serial connector using the supplied HOST Serial Cable.

With Linux, the program and supporting files come installed on your onboard primary SBC in `/usr/local/SeekurOSParamsManager` Updates and the distribution on CD come as simple, compressed tar archives, so locate the file in /usr/local and, for example, use

```
% tar –zxvf SeekurOSParamsManager.tgz
```

to create the directory and its contents.

With Windows, the parameters manager comes with SeekurOS and the directory normally is located in C:\Program Files\MobileRobots\SeekurOSParamsManager.

Starting Up

ParamManagerStatic is a text-based console application which like `demo` is built with ARIA. To start it from a console terminal under Linux, navigate to the SeekurOSParamsManager directory and invoke the program:

```
% cd /usr/local/SeekurParamManager
% ./paramsManagerStatic <options>
```

With Windows PCs, you may double-click the `ParamsManager.exe` icon to automatically open a console window and start the program without any options.

To start up with command-line options, `run` the program from the `Start menu`, or run `Command` from the `Start` menu, then navigate to

the SeekurOS directory and start `seekurParamManager.exe` with options.

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

```
C:\> cd Program
Files\MobileRobots\SeekurOSParamsManager
C:…> seekurParamManager <options>
```

Normally (without any command-line arguments), `paramsManagerStatic` starts up expecting to connect with SeekurOS through the `COM1` or `/dev/ttyS0` serial port. To start up the program and make a connection with a serial port other than the default `COM1` or `ttyS0`, for example:

```
C:…> seekurparammanagerstatic –rp COM3
```

When successfully connected, the program automatically retrieves your robot microcontroller's FLASH-stored operating parameters and enters interactive mode.

Reviewing/Changing Parameter Values

Some character commands or keywords affect the operation of ParamsManagerStatic and the status of an associated parameters file, if any. For instance, to review the entire list of current values, type 'v' or 'view' followed by a return (`Enter`).

Each parameter has an associated ID number. Type the ID number followed by a new value separated by a space to change that value. To first review the value of a parameter before changing it, type its ID number alone at the command prompt. At the end of the listing, you will be prompted to provide a new value. Type in a new value or press Return (Enter) alone to keep the current value.

Save Your Work!

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

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

Table 13. SeekurParamManagerStatic keys and keywords

COMMAND	DESCRIPTION
<id#>	Type a parameter's ID number alone to display its current value and have the opportunity to change it.
v or view	Display all parameters IDs, description, current and default values.
e or export <filename>	Saves the current parameters into a local <filename>.spm text file.
s or save	Downloads current values to the Seekur microcontroller's FLASH.
q or quit	Exit the program.
? or h or help	Displays these commands and descriptions.

```
# .spm parameter file
# ParamsManager version: 1.1
```

Table 14. SeekurOS parameters for Seekur Jr

ID	Description	Current Value
1	Serial Number	-4
2	Robot Name	Seekur Jr 1
3	Steering 1 Offset (ticks)	0
4	Steering 2 Offset (ticks)	0
5	Steering 3 Offset (ticks)	0
6	Steering 4 Offset (ticks)	0
7	State of Charge (Coulombs)	-27571964
8	Max Trans/Lat Velocity (mm/s)	1000
9	Max Rotational Velocity (deg/s)	100
10	Trans/Lat Acceleration (mm/s/s)	1000
11	Trans/Lat Deceleration (mm/s/s)	1000
12	Top Trans/Lat Accel (mm/s/s)	4000
13	Rotational Accel (deg/s/s)	300
14	Rotational Decel (deg/s/s)	300
15	Top Rotational Accel (deg/s/s)	380
16	mm/rev of tire	1226
17	Overcharge Count	0
18	Batt. Current Sensor Center (uV)	5073140
19	Batt. Curr. Sensor Sens. (nV/A)	10047040
20	Hotel Current Sensor Center (uV)	5090800
21	Hotel Curr. Sensor Sens. (nV/A)	41184000
22	Batt. Volt. Sensor Sens. (nV/A)	8156
23	Batt. Volt. Sensor Offset (uV)	-112900
24	Tire Slip Factor (* 10,000)	9758
25	Traction Current Limit (mA)	45000
26	Traction Dyn. Curr. Limit Enable	false
27	Traction Dyn. Limit Peak (mA)	40000
28	Traction Dyn. Limit Cont. (mA)	30000
29	Traction Velocity Kp	60
30	Traction Velocity Ki	2
31	Traction Velocity Kd	0
32	Traction Velocity Ki Max Windup	0
33	Traction Velocity FeedFwd Factor	0
34	Traction sVel Kp	100
35	Traction sVel Ki	1
36	Traction Current Kp	100
37	Traction Current Ki	100
38	Traction Velocity Ramp Enabled	true
39	Traction Accel dV (rpm per s)	5000
40	Traction Decel dV (rpm per s)	5000
41	Traction Motor 1 X position (mm)	0
42	Traction Motor 1 Y Position (mm)	470
43	Traction Motor 1 um/motor rev	-17519

44	Traction Motor 2 X Position (mm)	0
45	Traction Motor 2 Y Position (mm)	-470
46	Traction Motor 2 um/motor rev	17519
47	Traction Motor 3 X Position (mm)	0
48	Traction Motor 3 Y Position (mm)	0
49	Traction Motor 3 um/motor rev	0
50	Traction Motor 4 X Position (mm)	0
51	Traction Motor 4 Y Position (mm)	0
52	Traction Motor 4 um/motor rev	0
53	Steer Motor 1 counts/wheel rev	0
54	Steer Motor 2 counts/wheel rev	0
55	Steer Motor 3 counts/wheel rev	0
56	Steer Motor 4 counts/wheel rev	0
57	Robot Subtype	SeekurJr
58	Robot Has Brakes	false

Chapter 8 Calibration & Maintenance

Your MOBILEROBOTS platform is built to last a lifetime and requires little maintenance.

TIRE INFLATION

Maintain even tire inflation for proper navigation of your Seekur Jr. We ship with each pneumatic tire inflated to its maximum 28 psi.

TIGHTEN LOOSE BOLTS AND SCREWS

Check for loose bolts and screws weekly, especially wheel bolts and the bolts holding the bearing housing (knuckle). The forces involved with skidding a heavy robot can even loosen loctite-sealed bolts and screws..

BATTERIES

It's best to keep Seekur Jr's batteries on its Power/Recharging Accessory when not in use. Fully recharge each battery pack monthly particularly if put in storage.

FACTORY REPAIRS

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

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

Describe the problem in as much detail as possible. Also include your **robot's serial number** (IMPORTANT!) as well as name, email and mail addresses, along with phone and fax numbers.

Tell us your robot's SERIAL NUMBER

Tell us when and how we can best contact you (we will assume email is the best manner, unless otherwise notified).

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

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

PORTS & CONNECTORS

This Appendix contains pin-out and electrical specifications for the external I/O and communication ports and connectors for Seekur Jr.

Table 15. Microcontroller serial; 13-pos amphenol ALOOF11-35S

Pin	Function	Notes
1	NC	No connection
2	RxD from µC	Maintenance connection
3	TxD to µC	Maintenance connection
4	TxD to µC	HOST connection, only on robots without onboard computer
5	RxD from µC	HOST connection, only on robots without onboard computer
6	DSR	Maintenance connection
7	Signal GND	Only on robots without onboard computer
8	Signal GND	Only on robots without onboard computer
9	RI	Maintenance connection
10	Signal GND	Maintenance connection
11	NC	
12	+12V @ 1A max.	Only on robots without onboard computer
13	Power GND	Only on robots without onboard computer

Table 16. Charging/Power; 10-pos CPC

Pin	Function	Notes
1	NC	No connection
2	+ charge power	
3-4	NC	
5	charger detect	
6	charger detect	
7-8	NC	
9	- charge power	
10	NC	

Table 17. E-stops; 4-pos CPC AMP 206430-1

Pin	Function	Notes
1	e-stop	Short pins 1 and 2 to allow robot to power ON
2	e-stop	
3-4	NC	No connection

Table 18. Bumpers, 8-pos AMP CPC

Pin	Function	Notes
1	bumper 0 or 5	Active OPEN
2	GND	
3	bumper 1 or 6	
4	GND	
5	bumper 2 or 7	
6	GND	
7	bumper 3 or 8	
8	GND	

Table 19. GPS/compass connector; 66-pos Amphelite connector ALOOF19-35S

Pin	Function	Notes
1-8	NC	No connection
9	+12V compass	
10	Power GND compass	
11	NC	
12	RxD compass	db9 pin 2
13	TxD compass	db9 pin 3
14	NC	
15	Signal GND compass	db9 pin 5
16-55	NC	
56	+12V GPS	
57	Power GND GPS	
58	NC	
59	RxD GPS	db9 pin 2

60	TxD GPS	db9 pin 3
61	NC	
62	Signal GND GPS	db9 pin 5
63-66	NC	

Table 20. LMS111 LRF connector; 14-pos CPC Tyco 796272-1

Pin	**function**	**Notes**
1	NC	No connection
2	NC	RxD Seekur Sr
3	NC	TxD Seekur Sr
4	LRF power +24V regulated	
5	LRF power GND	
6-8	NC	
9	NC	Signal GND Seekur Sr
10-11	NC	
12	LRF heat +24V raw	
13	NC	
14	LRF heat GND	

Appendix B

Power Distribution Board

Although most power ports are allocated to standard features of Seekur Jr, there are a few available for user accessories. Most ports are regulated, though raw battery/charger power is available. All ports are protected. Power is switched through software: Use client command #116 with a 2-byte argument specifying the power port number and state (0=OFF; 1=ON). Each port has an associated LED indicator which is lit when ON. The circuit breaker will disconnect power to the port (associated LED OFF) device if its current draw exceeds the available current or if you attach it when the system is active. To recover through software without cycling power, first send the client command to turn the port OFF, then the command to turn it back ON.

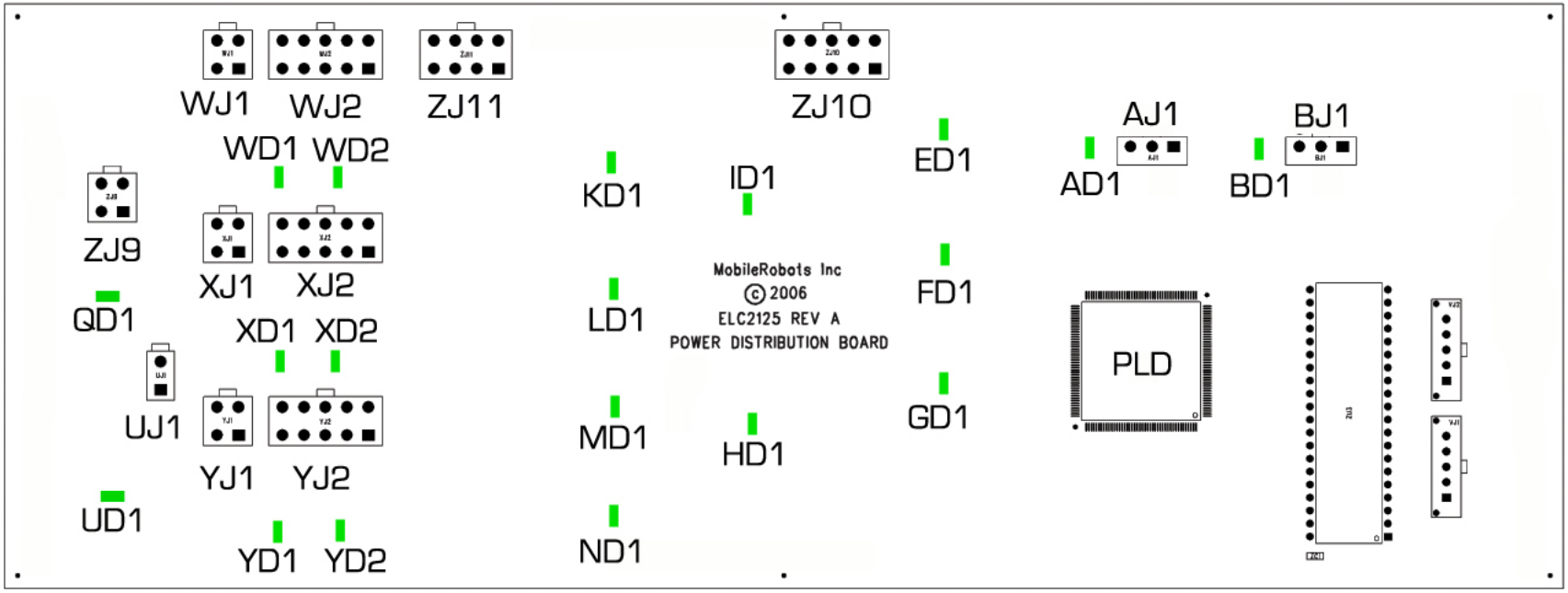


Figure 14. Seekur Jr Power Distribution Board Connectors and Indicators

Table 21. Seekur Jr PDB Connections

Connector	Voltage	Available Current	Port Number	Intended Function	Pin	Description	Indicator LED
AJ1	24V Raw	5A	11	LRF Heat	1	18-36Vout	AD1
					2	GND	
					3	N/C	
BJ1	24V Raw	5A	12	PTC Camera	1	18-36Vout	BD1
					2	GND	
					3	N/C	
ZJ10	24V Regulated	1A	n/a	CAN 2	1	24Vout	ED1
	24V Regulated	1A	n/a	CAN 3	2	24Vout	FD1
	24V Regulated	2A	8	Available	3	24Vout	GD1
	24V Regulated	1A	9	LRF1	4	24Vout	HD1
	24V Regulated	1A	10	PTU1	5	24Vout	ID1
					6	GND	
					7	GND	
					8	GND	
					9	GND	
					10	GND	
ZJ11	12V Regulated	2A	4	PoE 12v	1	12Vout	KD1
	12V Regulated	2A	5	Ethernet Switch	2	12Vout	LD1
	12V Regulated	2A	6	GPS	3	12Vout	MD1
	12V Regulated	2A	7	Available	4	12Vout	ND1

					5	GND	
					6	GND	
					7	GND	
					8	GND	
ZJ9	5V Regulated	2A	2	Available	1	5Vout	QD1
		2A	3	Available	2	5Vout	QD1
					3	GND	
					4	GND	
UJ1	5V Regulated	2A	n/a	Microcontroller	1	5Vout	UD1
					2	GND	
WJ1	5V Regulated	8A (total for WJ1+WJ2)	n/a	Not available	1	CompHD5V	WD1
					2	N/C	
					3	N/C	
					4	GND	
WJ2	5V Regulated	8A (total for WJ1+WJ2)	n/a	Not available	1	GND	
	12V Regulated	2A			2	GND	
					3	GND	
					4	CompHD12V	WD2
					5	N/C	
					6	N/C	
					7	CompHD5V	WD1
					8	CompHD5V	WD1
					9	N/C	
					10	GND	
XJ1	5V Regulated	8A (total for XJ1+XJ2)	0	SBC 2	1	CompHD5V	XD1
					2	N/C	
					3	N/C	
					4	GND	

XJ2	5V Regulated	8A (total for XJ1+XJ2)	0	SBC 2	1	GND	
	12V Regulated	2A			2	GND	
					3	GND	
					4	CompHD12V	XD2
					5	N/C	
					6	N/C	
					7	CompHD5V	XD1
					8	CompHD5V	XD1
					9	N/C	
					10	GND	
YJ1	5V Regulated	8A (total for YJ1+YJ2)	1	SBC3	1	CompHD5V	YD1
					2	N/C	
					3	N/C	
					4	GND	
YJ2	5V Regulated	8A (total for YJ1+YJ2)	1	SBC3	1	GND	
	12V Regulated	2A			2	GND	
					3	GND	
					4	CompHD12V	YD2
					5	N/C	
					6	N/C	
					7	CompHD5V	YD1
					8	CompHD5V	YD1
					9	N/C	
					10	GND	

Appendix C

SPECIFICATIONS

DIMENSIONS**:** 105mm x 840mm x 500mm (lwh)
WEIGHT: 70kg (1 battery)
GROUND CLEARANCE: 100mm
TIRES: 400mm
WHEELBASE: 425mm
BODY: Lightweight aluminum all-weather unibody
IP RATING: 54
TEMPERATURE RATING: –5°C-+35°C standard operation
ENERGY STORAGE: NiMH
RUN TIME: up to 3 hr in continuous motion
PAYLOAD: 50kg

SENSORS (optional)
SICK LMS100 Laser rangefinder
Inertial measurement unit
840-tick per revolution Hall Effect sensors on traction motors
MobileRanger outdoor stereovision system
MobileRanger C3D obstacle avoidance system
Weatherized Pan-Tilt-Zoom camera with near IR sensitivity

SOFTWARE *included unless otherwise noted:*
Seekur server operating system
ARIA API, available in source under GPL
Mapper3 indoor mapping software
MobileEyes real-time robot command & control GUI
Optional ARNL, with Laser Mapping & Navigation package
Optional mOGS, with DGPS package

NAVIGATION & MOTION
STEERING & SUSPENSION: 4 wheel skid steering
MAXIMUM SPEED: expected 1.2 mps
TERRAIN: Pavement, grass, snow, and dirt terrain
SLOPE: expected 75% grade traversibility
TYPES: guarded tele-operation, manual remote or manual direct drive, plus indoor/outdoor autonomous operation options

OPTIONS
Programmable SBCs with CompactFlash or solid state drive
Expandable user digital and analog I/O on external high speed CANbus
Provision for DGPS, PTZ camera, 2 laser rangefinders, 3 stereocameras, and more
Optional LCD with backlight with diagnostic screens
24V power with +/- conversion capability
5/12/24VDC regulated export power

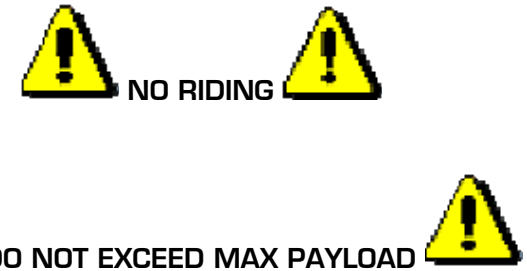
COMMUNICATIONS
Wireless 900MHz or WiFi option

WARRANTY
1 year included; 3 year optional

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.



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