I OVERVIEW

This is an introductory primer for new users of Shakey. We will content ourselves with simple outlines of functional (if inelegant) procedures by which a person can gain access to various capabilities of the robot. This primer will concentrate on the PDP-10 action software—as opposed to both STRIPS (and friends) on one side, and the PDP-15 and hardware on the other side. This action software falls naturally into three levels along an axis of sophistication (or undue complexity, depending on your intended use of the robot). The names usually used for the different levels are: ROBDEM for the simplest, the low-level actions (LLA's), and the intermediate-level actions (ILA's). Each of these three levels will be explained in a separate section of this primer; in all cases, the explanations will, for the sake of brevity, include only the detail, ramifications, or motivation needed by a beginner for his initial encounters. With experience, you will begin to understand the wide range of permissible variation, but you are urged to take the instructions more literally at first.

Common to all use of the robot are a few basic operations. First of all, one needs to be logged in on a terminal in Shakey's play area. Turn on the circuit breakers at the rear end of the robot. If the power cord would hamper the robot's maneuvers, disconnect it and turn off the charger in the rack near the robot's fixed interface. (When you are finished with the robot, don't forget to turn him off, reconnect the power cord, turn on the charger and set its knobs according to the directions posted on its front panel.) Now clear the state of the robot. To do this, you will first put the fixed interface's ON LINE/OFF LINE switch in the OFF LINE position; then push the MASTER CLEAR button (this should get some tweets), and then put the switch in the ON LINE position. (ON LINE means that the robot is connected to the PDP-15.)

Since access to the robot is billed to projects and also permitted on a one-user-at-a-time basis, users may wish to know how to release the robot when they are not using it. All the programs under discussion here have means of intentionally getting back to the exec; when this "official" exit command is used, the program will release its access to the robot. Of course, one can always exit from a program back to the
exec by using a control-C; this doesn't go through the program's normal exit routine, and so the robot is still connected to the program (even though the program's operation has been temporarily suspended). Some find this a trivially fine distinction, but others may be interested to know exec commands to release the robot. The simplest is to "@ LOGOUT" the job or to "@ RESET" the program; it will do just as well to "@ RUN" any program or call any subsystem. (This automatically includes a "@ RESET" even if you are just going back to the same program.)

In each of the following sections (one for each level of software), the description will start with the assumption that the terminal is at the exec. The syntax will not be precisely defined, but each section will include illuminating examples in which the user's input will be underlined.
II ROBDEM

ROBDEM gives the experimenter almost direct access to the PDP-15 routines. We imagine it will be most useful to those wishing to move the robot without maintaining an internal model of the robot's state, for example, to move the robot's TV camera or to test the robot. You get into the program by telling the exec '@ROBDEM', and you get back out to the exec by giving ROBDEM the command '*E'. (Both are followed by carriage returns, as are all commands to ROBDEM.) The first ROBDEM command you will give will be the INIT command. This establishes the communication linkages and resets all the internal flags.

ROBDEM has commands corresponding to the robot's degrees of freedom: FOCUS, IRIS, OV RID, PAN, RANGE, ROLL, TILT, TURN, and TVON. Some of these take arguments: FOCUS, IRIS, PAN, ROLL, TILT, and TURN all take a signed octal number specifying a number of motor counts to move from the current position; OV RID takes a code number specifying what is to be overridden (0-nothing, 1-whiskers, 2-pushbar, 3-both). ROBDEM also has a status-reading command READ, which takes one argument: FOCUS, IRIS, PAN, RANGE, ROLL, TILT, TURN, or TVON. For best results, one should use the appropriate READ when either a motion is suspected of terminating abnormally (e.g., limit switch or whisker bump) or the robot mysteriously stops responding. In the latter case, the status report will probably show an "emergency"; the proper response is another INIT, which will reset all the internal indicators and proceed from there. (See the examples.) The approximate conversion factors (where known) are:

- PAN 1.1 counts per degree left;
- ROLL 562 (octal) counts per foot forward;
- TILT 13 (octal) counts per degree up;
- TURN -7 counts per degree left.

Aside from that, about the only significant point to add is that you can expect strange results if you do strange things. For example, if the robot is in the process of turning, a ROLL command can't be expected to do the "right" thing. If that is a nuisance, do READs to make sure an action has terminated before you go on to the next one.
Load the program.

*INIT
Initialize the communication channel.

*PAN 200
Pan to the left by 200 counts (about 116.4 degrees—just past the limit.)

READ PAN
Read the status of the pan before it has finished.

ASV: IN PROGRESS RESIDUAL COUNT: 0

READ PAN
Read the status again after it has finished—-it did indeed hit the

ASV: LIMIT RESIDUAL COUNT: 11
limit just before exhausting the count.

PAN -167
Pan back to the original position, i.e., pan 110.5 degrees to the right.

READ PAN
Check its status after it finishes.

ASV: NORMAL RESIDUAL COUNT: 0

TURN 473
Turn 45 degrees to the right.

READ TURN
It reports this status before it finishes.

ASV: IN PROGRESS RESIDUAL COUNT: 0 CATWHISKER: 0

READ TURN
When it finishes, it reports that fact too.

ASV: NORMAL RESIDUAL COUNT: 0 CATWHISKER: 0

ROLL 2710
Roll 4 feet forward.

The status of the roll shows that it bumped into something after 1.4 feet

ASV: STOPPED BY WHISKER RESIDUAL COUNT: 1706 CATWHISKER: 10000ERROR 1

**IN JSYS NOUT

ROLL -1706
Roll back by 2.5 feet.

READ ROLL
Now, how did it go? Actually not at all—the whisker is still engaged.

ASV: STOPPED BY WHISKER RESIDUAL COUNT: -1707 CATWHISKER: 10000ERROR

**IN JSYS NOUT

OVRID 1
Override the whisker.

ROLL -1700
Success—we went all the way, and the whisker came free.

ASV: NORMAL RESIDUAL COUNT: -0 CATWHISKER: 0

E
Go back to the exec and release the channel.
III THE LLA's

A. Introduction

The LLA package can be accessed by telling the exec "@LISP" and telling LISP to "-SYSIN((ROBOT)ACTIONROUTINES, SYSOUT)"; you can get back out to the exec by telling the LLA package to "SHAKEY--QUIT". The LLA package provides the experimenter with simple-minded access to the robot along with some simple error recovery; it maintains part of the robot's world model and as such, it is well adapted to use in conjunction with the ILA package. (The ILA package is loaded along with the LLA package, but the two are explained separately for simplicity.) The first ILA commands you will want to give will be SM, PDP15, REAL, IC, and IF. The first three disengage the robot simulator and as such need to be done only when you first load the program; the next two initialize the communication channel and the internal flags, respectively, and want to be done any time you have released the robot channel.

B. Interesting SHAKEY Commands

SHAKEY is a command interpreter in the same sense as the LISP editor. It accepts commands from the terminal and executes the associated actions; IC, SM, and QUIT are examples we have already seen. Any time SHAKEY cannot make sense of its input as a command, it will evaluate its input line; this can also be forced by SHAKEY's E command, which acts like the LISP editor's E command. A typeout of all commands SHAKEY will recognize can be elicited by the ? command*

Of most direct interest are SHAKEY's nine commands corresponding to nine of the LLA's†: ROLL, ROLLTO, TURN, TURNTO, TILT, TILTO, PAN, PANO, and OVRID. Eight of them take a signed number, which is interpreted as follows:

- ROLL feet forward from current;
- TURN degrees left from current;
- TILT degrees up from current;
- PAN degrees left from current;
- ROLLTO (Two numbers) the given X and Y coordinates;
- TURNTO the given bearing (in degrees to the left of the positive Y-axis);
- TILTO the given elevation (in degrees from horizontal);
- PANTO the given angle (in degrees) to the left of the robot's bearing.

*The evaluation is done by LISP, so the command line "E ?" would, for example, produce the LISP commands, and "E E ?" will get the value of the atom ?.
†Described in §III of Reference 1. That in turn is a condensation of Reference 2.
OVRID takes one number specifying what is to be overridden (0-nothing, 1-catwhiskers, 2-pushbar, 3-both) on subsequent rolls and turns. When you have rolled the robot into a box, you may wish to use the SHAKEY command BO (back off), which has the same effect as the sequence of three SHAKEY commands OVRID 3, ROLL -1, OVRID 0.

The internal model maintained by the LLA's can be interrogated by the SHAKEY commands W (where), DW ("dwhere"), HP (head position) and WC (whisker check). Certain aspects of the robot's position information can be set from the terminal, notably the "where" and "dwhere" information: to do this, execute the SHAKEY command SW (set where). This enters a subcommand mode (exited by the OK subcommand) where one can use the subcommands X, Y, THETA, DX*, DY*, DTHETA*, (each followed by a number) to set the corresponding part of the robot's internal model of its position. Additional subcommands are W and DW*, which are like the SHAKEY W and DW; moreover, ? shows the available subcommands while E or anything not recognized causes evaluation. The head position cannot be set from the terminal because the limit switches provide a means of calibrating the head; soon we should be able to set the hardware head to the internally modelled position by means of the "dickeybird," but timing problems currently prevent proper operation of that routine. The position of any box can be set (or interrogated) by means of the SB (set box) command*; it wants the name of a box and then enters a subcommand mode (which again is exited by the OK subcommand). (If the box is new, this will cause a new box to appear in the model.) In the subcommand mode, X, Y, DX, DY, and RADIUS (again, each followed by a number) each set the corresponding property of the box; W causes the information to issue forth, and ?, E and anything not recognized behave in the same old way.

*This feature is probably not of interest except in conjunction with the ILA's. It is described for symmetry only.
Get into LISP from the exec.

@L~.StIdâ' Get into LISP from the exec.

GOOD MORNING, MIKE. This much abbreviation can be used in the initial loadup command.

"(d1-SEP-72 1139159' (d<ROBOT>ACTUATION<SYSTUJ1)"

Here 'F' is a control-F and 'S' is an ALT MODE.

We can again check the position information.

((X 20.3) (Y 18.5) (BEARING 71) (ROOM NIL))

The robot turns 45 degrees to the right.

((X 20.3) (Y 18.5) (BEARING 26.4) (ROOM NIL))

Now he is told to move forward four feet.

We check his whiskers to see if he bumped into something. It turns out that his horizontal whisker is engaged. (The symbol FW means that at least one front whisker is engaged.)

Now he has no whiskers engaged.

The robot is told to pan 200 degrees to the left.

Instead, the pan is stopped by a limit switch at 110.5 degrees to the left.

((PAN 11.5) (TILT 0) (OPAN 0) (UTILT 0))

Now the robot is told to pan back to zero degrees, that is straight ahead.

That is where he arrives (or nearly so).

((PAN 9.3367432-7) (TILT 0) (OPAN 3.1199991) (UTILT 0))

This position is initially loaded into the model.

((X 4) (Y 4) (BEARING 4) (ROOM NIL))

We wish to change it.

These three commands

These three commands

set the new position

into the model.

as we can see.

((X 26.3) (Y 18.5) (BEARING 71) (ROOM NIL))

Now we can leave the "Set Where" subcommand mode.

We can again check the position information.

((X 20.3) (Y 18.5) (BEARING 71) (ROOM NIL))

The robot turns 45 degrees to the right.

That is where he arrives.

((X 20.3) (Y 18.5) (BEARING 26.4) (ROOM NIL))

Now he is told to move forward four feet.

We check his whiskers to see if he bumped into something. It turns out that his horizontal whisker is engaged. (The symbol FW means that at least one front whisker is engaged.)

Now he has no whiskers engaged.

We check his position again and find out that he has indeed backed up by one foot.

Now he has no whiskers engaged.

We go back to the exec, releasing the communication channels.

The vision simulator types a message indicating the simulated release of the vision channels.)

Now we can leave the "Set Where" subcommand mode.
IV THE ILA's

A. Introduction

The ILA's* can be gotten by telling the exec "@LISP" and then telling LISP to "-SYSIN(ROBOT)ACTIONROUTINES.SYSOUT". This will put you into the command interpreter SHAKEY; you can get back to the exec by giving the command "SHAKEY--QUIT". The ILA's are the lowest-level routines seen by STRIPS and PLANEX. As such, they have enough skill built into them to enable them plan a route through a room containing obstacles; they will visually acquire information (about the robot's own position or about the various boxes in the room) whenever appropriate; and they can handle recovery from many kinds of errors (such as incomplete or incorrect modeling of the world or mechanical or electronic failure on board the vehicle). The ILA package contains the LLA package and is most properly viewed as an extension of it; the reader is assumed familiar with the LLA section of this primer. (This section is included for completeness only; the vision routines have not yet been functionally connected to the ILA package.) To perform the necessary initialization, perform the five commands "SHAKEY--SM PDP15 REAL IC IF"; the first three disengage the robot simulator, the fourth initializes the communication channel and the fifth initializes some internal flags.

B. SHAKEY Commands for the ILA's

The LLA section mentions (without explanation) several SHAKEY commands which most properly pertain to the ILA package. Probably the most conspicuous is the commands to handle uncertainties: DW types out the uncertainties associated with the W typeout; the DX, DY, etc. subcommands of the SW and SB commands set the uncertainties associated with the X, Y, etc. subcommands. Each uncertainty is maintained by the appropriate effector routines, which means that the ILA's maintain the uncertainties in the robot's position; however, the uncertainties are only used at the ILA level. Uncertainties reflect not only the imprecision of measurement and motion but also certain gross errors, such as are induced by telemetry failures. The ILA's use the uncertainties to indicate the appropriateness of failure recovery procedures or other alternate courses of action, such as retry or visually ascertaining the state of the world.

Another SHAKEY command mentioned but not explained in the LLA section was SB (set box) command. It is used to set the attributes of a box (and to insert new boxes into the model); this information is used by the navigation routines which form part of the ILA package, but it is not used by the LLA's.

* Described in §IV of Reference 1. For experimental convenience, the ILA package contains a default model consisting of the basic floor plan of the experimental area, three boxes (B3, B4 and B5) and arbitrary position information for the robot. The floor plan and boxes (except their names) are as described in §II of Reference 1, and the robot's information is detailed in Appendix D of Reference 2.
Finally, the ILA's can be entered by the OE (operation execute) command. It takes one argument: the operation to be executed. (See the accompanying example.) The format right now is quite rigid: the operation is specified in the so-called "eval" format, that is, it is a form to be evaluated, rather than a function and a list of arguments.
We can call the ILA's the same way we call the LLA's because they are loaded together. (In fact, the ILA's use the LLA's as their subroutines.)

We can use the "set where" command to give the robot a position just on the RCLK side of DPOPClk and pointing into the room as if he had just come in.

(set where x 18)
(set where y 18)
(set where theta m)
(set where room rclk)
(set where dx o)
(set where dy o)
(set where theta 0)
(set where)

Now we can check the position information:
((x 28.6) (y 23.2) (theta 90) (room rclk))

Finally, we can set the position information for the box B3.

The on-line user will note that this position information is initially in the model. These commands are given here for illustrative purposes only.)

(set box y 27)
(set box dx o)
(set box dy o)

Now we can see the box's properties, including its position.
((box 837 (name nil) (x 27) (dx o) (dy o) (theta 1.5) (room rclk)))

(set box dx o)
This subcommand mode is exited in the usual way.

Finally, we can block door DPOPClk from the RCLK side using box B3.

(set box dy o)
(set box theta 0)
(set box)

(set box dx o)
(set box dy o)
(set box theta 0)
(set box)

Now we can set the position information for the robot:
((x 28.5947) (y 23.19273) (theta 90) (room rclk))

(set box dx o)
(set box dy o)
(set box theta 0)
(set box)

(set box dx o)
(set box dy o)
(set box theta 0)
(set box)

Finally, the robot finishes the task.

(set box dx o)
(set box dy o)
(set box theta 0)
(set box)

We can cleanly return to the exec.

This includes releasing the channels.)

(set box dx o)
(set box dy o)
(set box theta 0)
(set box)

We can call the ILA's the same way we call the LLA's because they are loaded together. (In fact, the ILA's use the LLA's as their subroutines.)
The next page is a map of Shakey's play area. You will notice that for the sake of convenience, the origin and orientation are adjusted so that the entire play area is in the first quadrant. The map is pretty much self-explanatory, with three exceptions. First, one should know that the floor tiles are two feet square; their boundaries form a grid of even-numbered meridians and parallels. Secondly, our canonical north is toward the top of the map, i.e., toward increasing y-coordinates. Finally, bearings are measured from the northerly direction, with angles increasing counterclockwise (i.e., to the left); thus, for example, canonical east (the positive x-axis) has a bearing of -90 degrees.
FIGURE 1  MAP OF SHAKEY'S EXPERIMENTAL ENVIRONMENT
VI  BIBLIOGRAPHY
