Using the Robotics Interface:

This verison of Scheme has a simple programmer's interface for robotics sensors and effectors, such as in Lego® Technic Control kits, using the Hyperbot™ controller.

Hardware

The Hyperbot controller hardware can be obtained from Bots, 905 South Springer Rd., Los Altos, CA 94024, USA. Phone: (415) 949-2126; Fax: (415) 949-2566; CompuServe: 75500,2027; Internet: 75500.2027@compuserve.com. The serial cable they supply can plug directly into either NeXT™ serial port.

Lego kits compatible with the Hyperbot controller can be obtained from Bots, or directly from Lego Dacta, 555 Taylor Rd., P.O. Box 1600, Enfield, CT 06083-1600, USA. Phone: (800) 526-8339; Fax: (203) 763-2466.

Other kits, e.g. fischertechnik® and Capsella®, may also work with Hyperbot; contact Bots for more information.

Software

The following procedures are available in the SICP Scheme dialect, which is the default preference setting:

`(maybe-gc #!optional free-fraction-threshold)`

does a garbage collection if the fraction of the heap that is free is less than the specified free-fraction-threshold, which defaults to .1. If the threshold is explicitly specied, it must be a positive real number not greater than 1. (1 may be used to force an unconditional garbage collection.) This procedure is useful before turning motors on so that a garbage collection won't occur while the motors are on, perhaps delaying turning them off again.

`(real-time-clock)`
returns real time in milliseconds since scheme started.

`(wait-microseconds delay)`
sleeps for delay microseconds. The delay must be an exact integer.

`(wait-until predicate)`
repeatedly applies the nilary predicate until it returns true; currently sleeps .01 seconds between each attempt, but that may change.

`(make-hyperbot port)`
creates a hyperbot object, which must be passed to all the hyperbot operations; does all the resetting/initialization described for `hyperbot/reset`. Port must be either the symbol `a` or the symbol `b`, indicating which serial port the controller is plugged into.

Other argument values are reserved for future use with networking, etc.

`(hyperbot/reset hyperbot)`

resets all state: motors off, maximum power 14, sensitivity 11, fast sensor sampling, all counters zeroed, communications channels cleared. This is useful if you have managed to wedge things. Also, this should always be done after pressing the stop button on the controller box. If you have wedged things so thoroughly that even this procedure doesn't work, it is worth trying to press the stop button and then use this procedure again.

`(hyperbot/close hyperbot)`

severs communications and frees resources. It is illegal to try doing any further operations on this hyperbot object, but a new make-hyperbot can be done on the same port.

`(hyperbot/make-action hyperbot #!optional power1 power2 power3 power4)`

returns a nilary procedure which when invoked sets the output powers as specified. Any powers which are either omitted or specified as `#f` are left unchanged by the action. Any power which isn't false must be a real number in the range `-1` to `1` inclusive. `-1` means maximum reverse power, `0` means off, `1` means maximum forward power, and fractional values can be used to specify reduced power levels.

`(hyperbot/get-sensor hyperbot sensor)`

returns true iff the specified sensor contacts are closed; sensor must be an exact integer in the range 1±4.

`(hyperbot/get-counter hyperbot sensor)`

returns the number of times the specifed sensor has cycled from

open to closed and back to open since the counter was reset; the sensor number must be an exact integer 1±4.

`(hyperbot/reset-counter hyperbot sensor)`
resets to zero the counter associated with the specified sensor; the sensor number must be an exact integer 1±4.

`(hyperbot/get-period hyperbot sensor)`
returns the number of milliseconds the last cycle from open to closed and back to open took on the specified sensor; the sensor number must be an exact integer 1±4.

`(hyperbot/set-sensitivity hyperbot sensitivity)`
sets the specified sensitivity, which must be an exact integer in the range 0±15.

`(hyperbot/get-sensitivity hyperbot)`

returns current sensitivity, an exact integer in the range 0±15.

`(hyperbot/calibrate-light hyperbot)`
assuming an opto-sensor is connected to input number one and it is being exposed to ªlightº conditions, sets the sensitivity.

`(hyperbot/calibrate-dark hyperbot)`
as with `hyperbot/calibrate-light`, but assumes ªdarkº exposure.

`(hyperbot/calibrate-average hyperbot)`
sets sensitivity by averaging light and dark settings.

`(hyperbot/set-sensor-mode hyperbot mode)`
If mode is true, uses slow sampling; false means fast sampling, the default.

`(hyperbot/set-maximum-power hyperbot limit)`

Limit must be an exact integer in the range 1-24. The default value, 14, is recommended for Lego 4.5v motors; use of higher values could cause damage.

`(hyperbot/get-maximum-power hyperbot)`

returns the current setting.