Annex B: Matlab source code for Khepera Toolbox

```
function [obj_in_front,obj_in_back]=checkobj(btmap1)
%CHECKOBJ    Check for an obstacle in front or in back of the simulated khep.
%
%                   This function checks the outer edge in front and behind the khep
%                   to see if there is an object present.
%
%                   [OBJ_IN_FRONT,OBJ_IN_BACK]=CHECKOBJ(BTMAP1)
%                   OBJ_IN_FRONT - Binary indicator of presence of object in front
%                   OBJ_IN_BACK  - Binary indicator of presense of object in back
%                   BTMAP1 - The name of the mat file playpen map used to check for objects.
%                   In the map, a 44 corresponds to a blue pixel, which is interpreted
%                           here as an object.
%
%                   To achieve finer or more coarse results:
%                   A.  Reduce the middle element of each sweep vector to get a
%                   finer granularity.
%                   B.  Adjust either the radius (the numerical multiplier in front of the trig
%                   statements) or the outer bounds of the sweep vector or both to eliminate
%                   problems where a wall is detected on the outer edge near the motors and
%                   prevents forward or backward motion when logically it should be permitted.
%                   This is identified as "sticking".
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 010/12/96
% --------------------------------------------------------------------------------


%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
          return;
end
khep_data=get(khep_h,'userdata');

%  Initialize the object indication variables
obj_in_front=0;
obj_in_back=0;
%  Define the front sweep points
front_sweep=(-50:5:50);
[dummy,width]=size(front_sweep);

%  Initial khep angle is in khep_data(3)
init_angle=khep_data(3)*ones(1,width);

%  Check for obstacle up front
xsweepfront=round(29*cos((init_angle+front_sweep)*pi/180)+khep_data(1,1));
```

```
ysweepfront=round(29*sin((init_angle+front_sweep)*pi/180)+khep_data(1,2));
% btmap1 referenced with two vectors produces a matrix of cross-referenced
% values.  The one-to-one matrix entry values lie on the main diagonal.
tmp=(diag(btmap1(ysweepfront,xsweepfront)',0));
if any(tmp==44)
        % Obstacle forward
        obj_in_front=1;
end

%  Check for obstacle behind
rear_sweep=(130:5:230);
xsweeprear=round(29*cos((init_angle+rear_sweep)*pi/180)+khep_data(1,1));
ysweeprear=round(29*sin((init_angle+rear_sweep)*pi/180)+khep_data(1,2));
%line('xdata',xsweeprear,'ydata',ysweeprear);
tmp=(diag(btmap1(ysweeprear,xsweeprear)',0));
if any(tmp==44)
        % Obstacle behind
        obj_in_back=1;
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function [new_angle]=cvrtang(old_angle)
%CVRTANG     Convert angles to under 180 degrees
%
%                   [NEW_ANGLE]=CVRTANG(OLD_ANGLE)
%                   NEW_ANGLE - An angle between -180 and 180
%                   OLD_ANGLE - An angle in the range [360 -360]
%
%                   This function is used to convert angles to the range
%                   [-180 180].  It is used by ulsv which computes angles
%                   using four quadrant inverse tangent.


for i=1:length(old_angle)
        if old_angle(i)<(-180)
                new_angles(i)=old_angle(i)+360;
        elseif old_angle(i)>180
                new_angles(i)=old_angle(i)-360;
        else
                new_angles(i)=old_angle(i);
        end
end

new_angle=new_angles';
```

Annex B: Matlab source code for Khepera Toolbox

```
function kheprom(btmap1,testmode)
%KHEPROM     The simulated ROM supervisor of the Khepera
%
%
%                   KHEPROM(BTMAP1,TESTMODE)
%                   BTMAP1 - The name of the mat file playpen map used to check for objects.
%                            Although it is not used directly in the code below, it is
%                            passed to sub-functions for processing.
%
%                   TESTMODE - A dummy argument used to indicate that the test mode of
%                                  certain functions (upsv, ulsv) should be used
%
%                   This code attempts to emulate the functionality of the ROM supervisor
%                   programmed into the real Khepera.  It isolates the higher level
%                   simulator core functions (sim_rps, sim_sms, etc) from the code
%                   required to implement the instructions on the simulated khepera.
%                   This is done by using the khepera patch object's userdata property
%                   as a storage space for a vector which is broken down into fields
%                   corresponding to the registers found in the khepera.  Just as the
%                   core functions for the real khepera (rps, sms, spc, etc) read and
%                   write values to/from registers on the physical khepera (which are
%                   then acted on by the ROM), the simulator core functions read and
%                   write to fields of the vector stored in the userdata space and this
%                   vector is then used by this function to perform actions.
%
%                   The fields of the userdata property vector are set up as follows:
%                   Columns 1-3:       absolute x, y, and rotation of khep in degrees
%                   Columns 4-11:     proximity sensor data
%                   Columns 12-19:   light sensor data
%                   Columns20-21     Speed motor left, Speed motor right
%                   Columns 22-27   Motion control registers.Left: T,M,E Right: T,M,E
%                   Columns 28-29    Position counter registers
%                   Columns 30-31    Position controller registers - position to be reached.
%                   NOTE:  for the purposes of simulation, there is not a 1:1 correspondence
%                            between the fields indicated above and the registers in the khep.
%
%                   In an iterative algorithm using the simulator, KHEPROM must be executed
%                   once per iteration to allow an update of the position, sensors, etc.
%
%                   The testmode option should not ever have to be provided by the user.  It
%                   is provided internally by certain functions (simsense) to invoke special
%                   test functionality.  Except for this header, this second option will be
%                   and undocumented feature.
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------
```

Annex B: Matlab source code for Khepera Toolbox

```
%  Note:  the following constants are used in the position mode and may
%  differ from the granularity specified in MOVEKHEP or ROTKHEPsince they
%  function in the speed mode.
%  Rotation granularity
rot_const=15;
%  Translation granularity
gran=5;

%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');

%  Read the motion controller status register and determine which
%  blocks of commands are appropriate.

%  If the motion controller is in the speed mode,...
if khep_data(23)==1
        %   If The Motor Registers are set to a speed, move the khep
        if ((khep_data(20)>0)&(khep_data(21)>0))
                movekhep(1,btmap1);
        elseif ((khep_data(20)<0)&(khep_data(21)<0))
                movekhep(-1,btmap1);
        elseif (khep_data(20)<khep_data(21))
                rotkhep(rot_const);
        elseif (khep_data(20)>khep_data(21))
                rotkhep(-rot_const);
        end
end
%  If the motion controller is in the position mode, ...
if khep_data(23)==0
        %  If the position reached bit not set, move the khep
        if khep_data(22)==0
                % Dirn is the direction to move in (-1=backwards, right, 1=forwards,
                % left).  Num_pulses computes the pulses remaining until the position
                % control registers equal the position counters.  Thresh computes
                % the region in which the position counters will be considered equal
                % to the control counters.  It is the number of pulses in one movement
                % or turn iteration as defined by gran or rot_const.

                %  Check and see if:  both position control registers have
                %  the same sign, and both position counters have the same sign,
                %  then the desired motion is linear
                a=khep_data(30);
                b=khep_data(31);
                c=khep_data(28);
```

```
                d=khep_data(29);
                if ((a<=0&b<=0)|(a>=0&b>=0))&((c<=0&d<=0)|(c>=0&d>=0))
                        dirn=sign(khep_data(30)-khep_data(28));
                        num_pulses=abs(khep_data(30)-khep_data(28));
                        %  NOTE:  Matlab's rounding error can cause trouble in the
                        %  next line when num_pulses looks like the quantity on the
                        %  other side of the negative sign.  Hence, add 1 to num_pulses
                        %  to push it just under if it looks exactly equal.  This
                        %  gaurantees another cycle, in which case num_pulses will be
                        %  negative and the whole quantity will be grossly less than
                        %  zero.
                        if (num_pulses+1-gran*12.5)<0
                                khep_data(22)=1;
                                %  Note that the position counter will go as close as
                                %  the nearest full divisor of (12.5*gran) into the
                                %  position control register.
                        else
                                movekhep(dirn,btmap1);
                        end

                %  Otherwise assume the motion to be rotational
                else
                        %  Remember, we assume that the
                        %  magnitude of khep_data(30) and (31) are equal.
                        dirn=sign(khep_data(30)-khep_data(28));
                        num_pulses=abs(khep_data(30)-khep_data(28));

                        %  NOTE:  Matlab's rounding error can cause trouble in the
                        %  next line when num_pulses looks like the quantity on the
                        %  other side of the negative sign.  Hence, add 1 to num_pulses
                        %  to push it just under if it looks exactly equal.  This
                        %  gaurantees another cycle, in which case num_pulses will be
                        %  negative and the whole quantity will be grossly less than
                        %  zero.
                        if ((num_pulses+1)-2042*(abs(rot_const)/360))<0
                                khep_data(22)=1;
                                %  Note that the position counter will go as close as
                                %  the nearest full divisior of (2042*(rot_const/360))
                                %  into the position control register.
                        else
                                rotkhep(-1*dirn*rot_const);
                        end
                end
        end
end

%  Before updating the sensors in the new position, re-read the registers because
%  movekhep or rotkhep will have changed them.
khep_data=get(khep_h,'userdata');
```

Annex B: Matlab source code for Khepera Toolbox

```
%  Update the proximity sensors in the new position.
upsv(btmap1);

%  Update the light sensors in the new position
if nargin==1
        ulsv;
elseif nargin==2
        ulsv2;
end
%  Check to see if the lamps should be swapped
liteswap;
```

Annex B: Matlab source code for Khepera Toolbox

```
function liteswap()
%LITESWAP      Moves the current source of light when in random or sequence mode
%
%                       Copyright (c) 1996 Capt Rich Goyette
%                               Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------

% Remove the following statements when testing the overall controller
global tag_off tag_on

if tag_off==[]
        tag_off=0;
end
if tag_on==[]
        tag_on=0;
end



lamp_off_color=[.51 .02 .25];
thresh1=50;
%  Get the lamp sequencing mode from the main window
lamp_mode=get(findobj('tag','lamp_con'),'value');

%  If the lamp mode is not all on, the do the checks
if lamp_mode>1
        %  Read the khepera userdata file and get the absolute position
        khep_h=findobj('tag','khep');
        if khep_h==[]
                warndlg('No khepera present!');
                return;
        end
        khep_data=get(khep_h,'userdata');
        khep_pos=khep_data(1:2);

        % Now find the lamp that is on.
        % This may take too long.  Find another way
        lamp_handles=findobj('tag','lamp');
        for i=1:length(lamp_handles)
                lamp_dat=get(lamp_handles(i),'userdata');
                if lamp_dat(3)==1
                        tmpl=i;
                        break;
                end
        end
```

```
lamp_dat=get(lamp_handles(tmpl),'userdata');
lamp_pos=lamp_dat(1:2);
pixeldist=dist(lamp_pos,khep_pos');
%  Now check and see if any of the front four sensors are on.  If so, switch
%  lamps.  This code must be removed when testing modules.
obst_vals=khep_data(4:11);
wall_bump=sum(khep_data(5:8)>700);
if (wall_bump>0)&(tag_on==0)
        tag_on=1
elseif(wall_bump==0)&(tag_on==1)
        tag_on=0
end

%  If the absolute distance between centre of khep and lit lamp < thresh,
%  then swap the lamps
if (pixeldist < thresh1)|((wall_bump>0)&(tag_off==0)&(tag_on==1))
        if lamp_mode==2
                light_a_lamp=round(rand*(length(lamp_handles)-1))+1;
                        for i=1:length(lamp_handles)
                                if i~=light_a_lamp
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=0;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor',lamp_off_color,...
                                                'edgecolor',lamp_off_color);
                                else
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=1;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor','yellow');
                                end
                        end

        elseif lamp_mode==3
                %  Select the next lamp that was placed
                tmpl=tmpl-1;
                %  If there isn't one, cycle to the top
                if tmpl==0
                        tmpl=length(lamp_handles);
                end
                %  Now reset all the lamps
                        for i=1:length(lamp_handles)
                                if i~=tmpl
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=0;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor',lamp_off_color,...
                                                'edgecolor',lamp_off_color);
                                else
                                        tmp2=get(lamp_handles(i),'userdata');
```

```
                                        tmp2(3)=1;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor','yellow');
                                    end
                                end

                    end
            end
end
tag_off=tag_on;
```

Annex B: Matlab source code for Khepera Toolbox

```
function movekhep(dirn,btmap1)
%MOVEKHEP   Translate the virtual khepera
%
%                   This function moves the khepera patch object a certain
%                   distance in the playpen.  The distance-per-call is determined
%                   by the value of gran (granularity).  Larger gran translates
%                   to larger distances convered.
%
%                   MOVEKHEP(DIRN,BTMAP1)
%                   DIRN - The desired direction of motion.  This value must be
%                           passed to the function because it is very difficult to
%                           determine within the function the actual direction
%                           (forwards or backwards) of the kehp.  This information
%                           is needed to prevent the khep from "sticking" to a wall.
%                           Takes values -1 (reverse), 1 (forward)
%
%                   BTMAP1 - The name of the mat file playpen map
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------



%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');

%  If dirn wasn't specified, error
if dirn==[]
        error('Direction must be specified in MOVEKHEP');
end

%  Define the movement granularity (gran) as the number of pixels moved per call
gran=5;
dist_vect=(1:gran);

%  Define forward or backward angle modification
if dirn==1
        angle_mod=0;
else
        angle_mod=180;
end
```

Annex B: Matlab source code for Khepera Toolbox

```
%  Keep track of the old absolute position so that it can be
%  subtracted from xdata,ydata
old_abs_x=khep_data(1);
old_abs_y=khep_data(2);

xmov=round(dist_vect*cos((khep_data(3)-angle_mod)*pi/180)+khep_data(1));
ymov=round(dist_vect*sin((khep_data(3)-angle_mod)*pi/180)+khep_data(2));


%  Define gran_dist as a measure to be used in updating the position counter, and
%  use tmpx and tmpy as the last absolute position obtained before hitting a wall.


for q=1:gran
        [objinfront,objinback]=checkobj(btmap1);

        %  If no object in front or back, update
        if (objinfront==0)&(objinback==0)
                tmpx=xmov(q);
                tmpy=ymov(q);
        end
        %  If an object in front, but moving backwards, update
        if (dirn==-1)&(objinfront==1)&(objinback==0)
                        tmpx=xmov(q);
                        tmpy=ymov(q);
        end
        %  If an object behind, but moving forwards, update
        if (dirn==1)&(objinback==1)&(objinfront==0)
                        tmpx=xmov(q);
                        tmpy=ymov(q);
        end
end

%  Now update the position counters.
%  gran is used here to compute the number of pulses to add to the
%  position registers.  Each pulse is 0.08mm. Hence,
%  there are 1/0.08 pulses/mm=12.5p/mm, and
%  thus 12.5*gran_dist pulses covered by the khep
%  Note that even if the khep hits a wall, the position counter
%  will still be updated.  This corresponds to reality in this
%  case since the wheels still spin.  Also, the counter is 32 bits, so limit the
%  count to 2147483648 (MSB for positive/negative)
if abs(khep_data(28)+dirn*12.5*gran)<=2147483648
        khep_data(28)=(khep_data(28)+dirn*12.5*gran);
        khep_data(29)=(khep_data(29)+dirn*12.5*gran);
else
        khep_data(28)=0;
        khep_data(29)=0;
end
```

Annex B: Matlab source code for Khepera Toolbox

```
%  Now update the absolute position entries.
if tmpx~=[]
        khep_data(1)=tmpx;
        khep_data(2)=tmpy;

        old_x_data=get(khep_h,'xdata');
        old_y_data=get(khep_h,'ydata');
        new_x_data=(old_x_data-old_abs_x)+khep_data(1);
        new_y_data=(old_y_data-old_abs_y)+khep_data(2);



        %  Now write this back to the patch object to update position
        set(khep_h,'xdata',new_x_data,'ydata',new_y_data);

end
%  Post any changes
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function putkhep(xabs,yabs)
%PUTKHEP       Place the khepera in the playpen
%
%                   This function places a patch object representing the khepera
%                   in a virtual playpen created by the default map or by a custom map.
%                   Sensors are not displayed.  Instead, an arrow showing the
%                   direction is displayed.  The rendering of the sensors is seen as time
%                   consuming and not justified or useful.
%
%                   This function stores all relevant information about the khep (absolute
%                   position, rotation, etc) in the userdata property of the patch object
%                   representing the robot.  The fields of this vector are set up as follows:
%                   Matrix: 1X21
%                   Columns 1-3:      absolute x, y, and rotation of khep in degrees
%                   Columns 4-11:     proximity sensor data
%                   Columns 12-19:    light sensor data
%                   Columns20-21      Speed motor left, Speed motor right
%                   Columns 22-27     Motion control registers.Left: T,M,E Right: T,M,E
%                   Columns 28-29     Position counter registers
%                   Columns 30-31     Position controller registers - position to be reached.
%
%                   PUTKHEP(XABS,YABS)
%                   XABS - absolute x position of centre of khepera in playpen
%                   YABS - absolute y position
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------


% See if there's already a khep on the board
h_ppen=findobj('tag','khep_simwindow');
if findobj(h_ppen,'tag','khep')~=[]
          return;
end

%bodyx and bodyy are the points comprising the khepera circular body
%sensx and sensy add the points that define the arrow.
bodyx=28*(cos([0:10:360]*(pi/180)));
sensx=[22 0 0 22];
bodyy=28*(sin([0:10:360]*(pi/180)));
sensy=[0 22 -22 0];

khepx=[bodyx sensx];
khepy=[bodyy sensy];

%  Note that the erasemode is very important for smooth graphics
```

Annex B: Matlab source code for Khepera Toolbox

```
khep_h=patch('XData',khepx+xabs,'YData',khepy+yabs,'Facecolor','red',...
                    'erasemode','xor','tag','khep');
%  Now store the initial position and orientation in the patch object's
%  userdata space.  Also, set up the userdata vector for use by KHEPROM

khep_data=zeros(1,31);
khep_data(1)=xabs;
khep_data(2)=yabs;
%  Set the position controller registers appropriately
%  Set the target (T) already reached
khep_data(22)=1;
khep_data(25)=1;
%  Set the mode (M) to speed
khep_data(23)=1;
khep_data(26)=1;
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function rotkhep(alpha)
%ROTKHEP      Rotate the simulated khepera
%
%             This function will rotate the simulated khepera left
%             or right by alpha degrees
%
%             ROTKHEP(ALPHA)
%             ALPHA - The angle in degrees of the rotation, or the granularity of
%                     each rotation iteration
%
%             Copyright (c) 1996 Capt Rich Goyette
%                   Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------

% Get the userdata from the Khep patch.  If none, just leave.
khep_h=findobj('tag','khep');
if khep_h==[]
        disp('just returning')
        return;
end

% Get all the pertinent data about the robot.
khep_data=get(khep_h,'userdata');

rotate(khep_h,[0 0 1],-1*alpha,[khep_data(1) khep_data(2) 0]);

% Put in a guard to restrict total rotation to 360 degrees
khep_data(3)=round(rem(khep_data(3)+alpha,360));

%  Now update the position counters.
%  There are 2*pi*R*12.5=2042 (R=26) pulses per full
%  rotation of the robot and the fraction of a full rotation
%  done in one rotational increment is 2042*(rot_const/360)

%  Define the direction

dirn=sign(alpha);
if abs(khep_data(28)+dirn*(2042*(abs(alpha)/360)))<=2147483648
        khep_data(28)=(khep_data(28)-dirn*(2042*(abs(alpha)/360)));
        khep_data(29)=(khep_data(29)+dirn*(2042*(abs(alpha)/360)));
else
        khep_data(28)=0;
        khep_data(29)=0;
end

% Now store the change in rotation back to userdata
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

Annex B: Matlab source code for Khepera Toolbox

```
function squashed_out = sigfn(angle,squeeze_factor)
%SIGFN          Sigmoid squeezing function for approximating sensor response.
%
%

squashed_out = 1 ./ (1+exp(-squeeze_factor*angle));
i = find(~finite(squashed_out));
squashed_out(i) = sign(angle(i))*0.5 + 0.5;
```

Annex B: Matlab source code for Khepera Toolbox

```
function [sensors]=sim_rls(port_id)
%SIM_RLS        The Read Light Sensors for the simulated khepera
%
%               [SENSORS]=RPS(PORT_ID)
%               SENSORS - 1X8 vector of light sensor readings
%               PORT_ID - A dummy number used for compatibility with the
%                                 core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------


%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');
sensors=khep_data(12:19);
```

Annex B: Matlab source code for Khepera Toolbox

```
function [motor_speeds]=sim_rms(port_id)
%SIM_RMS        Read the motor speeds for the simulated khepera
%
%               [MOTOR_SPEEDS]=SIM_RMS(PORT_ID)
%               MOTOR_SPEEDS - 1X2 vector of motor speeds
%               PORT_ID - A dummy number used for compatibility with the
%                                 core functions for the real khepera.
%
%               See the core function RMS for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------


%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');
motor_speeds=khep_data(20:21);
```

Annex B: Matlab source code for Khepera Toolbox

```
function [pos_vector]=sim_rpc(port_id)
%SIM_RPC        Read the position counter for the simulated khepera
%
%               [POS_VECTOR]=SIM_RPC(PORT_ID)
%               POS_VECTOR - A two element vector representing the postion
%                            counter values of the khepera motors
%               PORT_ID - A dummy number used for compatibility with the
%                            core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------


%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');

pos_vector=[khep_data(28) khep_data(29)];

return;
```

Annex B: Matlab source code for Khepera Toolbox

```
function [sensors]=sim_rps(port_id)
%SIM_RPS        The Read Proximity Sensors for the simulated khepera
%
%               [SENSORS]=RPS(PORT_ID)
%               SENSORS - 1X8 vector of proximity sensor readings
%               PORT_ID - A dummy number used for compatibility with the
%                                 core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------


%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');
sensors=khep_data(4:11);
```

Annex B: Matlab source code for Khepera Toolbox

```
function sim_skp(port_id,pos_left,pos_right)
%SIM_SKP        Set a position to be reached by the simulator
%
%               SIM_KHEP(PORT_ID,POS_LEFT,POS_RIGHT)
%               POS_LEFT - Postion to be reached by left motor
%               POS_RIGHT - Position to be reached by right motor
%               PORT_ID - A dummy number used for compatibility with the
%                                 core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------

%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
          return;
end
khep_data=get(khep_h,'userdata');

%  Set the position to be reached registers
khep_data(30)=pos_left;
khep_data(31)=pos_right;

%  Set the appropriate motion control registers
khep_data(22)=0;
khep_data(23)=0;
khep_data(25)=0;
khep_data(26)=0;

%  Write the data to the registers
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function sim_sms(port_id,motor_left,motor_right)
%SIM_SMS        The SET MOTOR SPEED command for the simulated khepera
%
%               The simulated khepera (as of this version) can only rotate left,
%               right, move backwards and forwards.  Hence, this function must
%               reject any setting that does not result in one of these motions.
%               It does so by checking that the two numbers are equal
%
%               SIM_SMS(PORT_ID,MOTOR_LEFT,MOTOR_RIGHT)
%               PORT_ID - A dummy number used for compatibility with the
%                                       core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------

%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');


if abs(motor_left) ~= abs(motor_right)
        disp('Motor speeds must be the same.  Exiting without setting speed');
        return;
end

%  Set the speed registers
khep_data(20)=motor_left;
khep_data(21)=motor_right;

%  Set the appropriate motion control registers
khep_data(22)=1;
khep_data(23)=1;
khep_data(25)=1;
khep_data(26)=1;

%  Write the data to the registers
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function sim_spc(port_id,pos_left,pos_right)
%SIM_SPC        Set the position counter for the simulated robot
%
%               SIM_SPC(PORT_ID,POS_LEFT,POS_RIGHT)
%               POS_LEFT - Number to set the postion counter to for left motor
%               POS_RIGHT - Number to set the postion counter to for right motor
%               PORT_ID - A dummy number used for compatibility with the
%                                       core functions for the real khepera.
%
%               See the core function RPC for functionality details
%
%               Copyright (c) 1996 Capt Rich Goyette
%                       Royal Military College, Canada
%
% V1.0
% 010/12/96
% ------------------------------------------------------------------------------

%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');

khep_data(28)=pos_left;
khep_data(29)=pos_right;

%  Write the data to the registers
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function ulsv()
%ULSV          Update Ambient Light Sensor Values for the Simulation Tester
%
%
%              This function updates the simulated khepera's light sensor values.
%              It is used by simsense.m and makes available all of the modelling
%              variables currently employed to shape the response of each sensor
%
%              See PUTKHEP for a description of the Khepera Patch userdata fields
%
%              UPSV
%
%              Copyright (c) 1996 Capt Rich Goyette
%                    Royal Military College, Canada
%
% V1.0
% 10/12/96
% ------------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Define the variables that will be used to model the light sensor's response

%  Define an ambient light level
amb_lite=450;

%  Define the maximum angle at which any sensor will be able to see some light
%  (note that the actual cone of vision is twice maxangle)
maxangle=110;

%  Define a cuttoff angle for flat response
cutoff_ang=0;

%  Get a noise measure to add
noise_mux=5;

%  Get squashing function
squash=0.0045;

%  Get Sigmoid Translation
xlate=200;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Define some constants to locate the sensors on the robot

%  With respect to the robot, each sensor points in the following direction
sensor_dir=[90;45;0;0;-45;-90;-180;180];
```

```
%  With a rotation of zero degrees, the physical locations of the centre each sensor
%  with respect to the centre of the robot (in mm) is:
sensor_pos=[10 24;20 18;25 6;25 -6;20 -18;10 -24;-24 -10;-24 10];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Locate position and angle data from the robot

%  Now get the khepera's absolute position and angle
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');
abs_pos=khep_data(1:2);
%  The absolute angle of the khepera will be theta
theta=khep_data(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Now compute the new locations of the sensors and the direction in which they point
sensor_array(:,1:3)=[sensor_pos*[cos(theta*pi/180);sin(theta*pi/180)]+abs_pos(1),...
                        sensor_pos*[-sin(theta*pi/180);cos(theta*pi/180)]+abs_pos(2),...
                        round(rem(sensor_dir+theta,360))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Assign all the light values to 500
khep_data(12:19)=amb_lite*ones(1,8);
tmp_data=amb_lite*ones(1,8);

%  Find the handles of all the lamps
lamp_handles=findobj('tag','lamp');

%  If there are no lamps, skip the following loop
if lamp_handles ~= []
        %  Iterate this loop for every lamp that exists
        for j=1:length(lamp_handles)
          %  Check and see if the lamp is on.  If not, skip it
          udat=get(lamp_handles(j),'userdata');
          if udat(3)~=0
                %  Get the position of current lamp (from the userdata property)
                lamp_pos=udat(1:2);

                %  Now compute the distance from each sensor to the lamp
                distx=lamp_pos(1)-sensor_array(:,1);
                disty=lamp_pos(2)-sensor_array(:,2);
                absdist=sqrt(distx.^2+disty.^2);
```

Annex B: Matlab source code for Khepera Toolbox

```
                % Get the total angle in degrees
                angle_check=cvrtang(180/pi*(atan2(disty,distx))-sensor_array(:,3));

                % Two factors whittle down the magnitude of the light.  See the
                % documentation for detailed analysis.
                for i=1:length(angle_check)
                        %  If the light bulb is within viewing range,
                        if abs(angle_check(i))<maxangle
                                %  If the bulb is within the cuttoff angle, then the response is flat
                                if abs(angle_check(i))<cutoff_ang
                                        off_centre=0;
                                        % Model the linear noise as a sigmoid as well
                                        tmp=(amb_lite-50)*(sigfn2(absdist(i),squash,xlate))+50;
                                        if tmp<50,tmp=50;,end;
                                else
                                        % Compute straight out response
                                        straight_on=(amb_lite-50)*(sigfn2(absdist(i),squash,xlate))+50;
                                        % Compute the effect the angle has.  Note that sensors and the
                                        % bulbs act as point sources so that when the khepera approaches
                                        % very close to a bulb, it is possible to generate large angles
                                        % between bulb and sensor even if the sensor appears directly
                                        % in front.  This accounts for the curve-up on close approaches.
                                        off_centre=sigfn2((abs(angle_check(i))/maxangle*2),4.5,1);

                                        tmp=straight_on+(amb_lite-straight_on)*off_centre;
                                        %  If the value is less than 50, make it 50
                                        if tmp<50
                                                tmp=50;
                                        end
                                end
                                tmp_data(i)=tmp;
                        end
                end

        end %light on check

        %  Now compare the results for this lamp with those produced by previous
        %  lamps and keep the lowest result.
        for q=1:8
                if tmp_data(q)<khep_data(q+11)
                        khep_data(q+11)=tmp_data(q);
                end
        end

      end   %for loop that iterates through all lamps
end     %lights exist check

%  Now add noise to the values
%  Multipy the light data vector by some a vector of random numbers conditioned by a
%  percentage
```

Annex B: Matlab source code for Khepera Toolbox

```
noise_vect=khep_data(12:19).*((rand(size(khep_data(12:19)))-0.5)*noise_mux/100);
mid_vect=khep_data(12:19)+noise_vect;
%  Remove anything less than zero
mid_vect=(~(mid_vect<0)).*mid_vect;
%  Remove anything over 500
mid_vect=(mid_vect>500)*500+(~(mid_vect>500)).*mid_vect;
%  Now set the value back
khep_data(12:19)=mid_vect;

%  Post any changes
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function ulsv2()
%ULSV2                  Update Ambient Light Sensor Values for the Simulation Tester
%
%
%               This function updates the simulated khepera's light sensor values.
%               It is used by simsense.m and makes available all of the modelling
%               variables currently employed to shape the response of each sensor
%
%               See PUTKHEP for a description of the Khepera Patch userdata fields
%
%               UPSV
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 10/12/96
% ------------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Define the variables that will be used to model the light sensor's response
%  Get them from the main window, so get the simtest window handle
simwin_handle=findobj('tag','sim_sense_tst_window');

%  Define an ambient light level
amb_lite=str2num(get(findobj(simwin_handle,'tag','amb_lt'),'string'));

%  Define the maximum angle at which any sensor will be able to see some light
%  (note that the actual cone of vision is twice maxangle)
maxangle=str2num(get(findobj(simwin_handle,'tag','max_angle'),'string'));

%  Define a cuttoff angle for flat response
cutoff_ang=str2num(get(findobj(simwin_handle,'tag','cut_off'),'string'));

%  Get a noise measure to add
noise_mux=str2num(get(findobj(simwin_handle,'tag','noise_lvl'),'string'));

%  Get squashing function
squash=str2num(get(findobj(simwin_handle,'tag','squash_con'),'string'));

%  Get Sigmoid Translation
xlate=str2num(get(findobj(simwin_handle,'tag','xlate_con'),'string'));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Define some constants to locate the sensors on the robot
```

```
% With respect to the robot, each sensor points in the following direction
sensor_dir=[90;45;0;0;-45;-90;-180;180];

% With a rotation of zero degrees, the physical locations of the centre each sensor
% with respect to the centre of the robot (in mm) is:
sensor_pos=[10 24;20 18;25 6;25 -6;20 -18;10 -24;-24 -10;-24 10];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Locate position and angle data from the robot

% Now get the khepera's absolute position and angle
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');
abs_pos=khep_data(1:2);
% The absolute angle of the khepera will be theta
theta=khep_data(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now compute the new locations of the sensors and the direction in which they point
sensor_array(:,1:3)=[sensor_pos*[cos(theta*pi/180);sin(theta*pi/180)]+abs_pos(1),...
                        sensor_pos*[-sin(theta*pi/180);cos(theta*pi/180)]+abs_pos(2),...
                        round(rem(sensor_dir+theta,360))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assign all the light values to 500
khep_data(12:19)=amb_lite*ones(1,8);
tmp_data=amb_lite*ones(1,8);

% Find the handles of all the lamps
lamp_handles=findobj('tag','lamp');

% If there are no lamps, skip the following loop
if lamp_handles ~= []
        % Iterate this loop for every lamp that exists
        for j=1:length(lamp_handles)
         % Check and see if the lamp is on.  If not, skip it
         udat=get(lamp_handles(j),'userdata');
         if udat(3)~=0
                % Get the position of current lamp (from the userdata property)
                lamp_pos=udat(1:2);

                % Now compute the distance from each sensor to the lamp
                distx=lamp_pos(1)-sensor_array(:,1);
                disty=lamp_pos(2)-sensor_array(:,2);
```

```
                absdist=sqrt(distx.^2+disty.^2);

                % Get the total angle in degrees
                angle_check=cvrtang(180/pi*(atan2(disty,distx))-sensor_array(:,3));

                % Two factors whittle down the magnitude of the light.  See the
                % documentation for detailed analysis.
                for i=1:length(angle_check)
                        %  If the light bulb is within viewing range,
                        if abs(angle_check(i))<maxangle
                                %  If the bulb is within the cuttoff angle, then the response is flat
                                if abs(angle_check(i))<cutoff_ang
                                        off_centre=0;
                                        % Model the linear noise as a sigmoid as well
                                        tmp=(amb_lite-50)*(sigfn2(absdist(i),squash,xlate))+50;
                                        if tmp<50,tmp=50;,end;
                                else
                                        % Compute straight out response
                                        straight_on=(amb_lite-50)*(sigfn2(absdist(i),squash,xlate))+50;
                                        % Compute the effect the angle has.  Note that sensors and the
                                        % bulbs act as point sources so that when the khepera approaches
                                        % very close to a bulb, it is possible to generate large angles
                                        % between bulb and sensor even if the sensor appears directly
                                        % in front.  This accounts for the curve-up on close approaches.

                                        off_centre=sigfn2((abs(angle_check(i))/maxangle*2),4.5,1);
                                        tmp=straight_on+(amb_lite-straight_on)*off_centre;
                                        %  If the value is less than 50, make it 50
                                        if tmp<50
                                                tmp=50;
                                        end
                                end
                                tmp_data(i)=tmp;
                        end
                end

        end %light on check

        %  Now compare the results for this lamp with those produced by previous
        %  lamps and keep the lowest result.
        for q=1:8
                if tmp_data(q)<khep_data(q+11)
                        khep_data(q+11)=tmp_data(q);
                end
        end

    end   %for loop that iterates through all lamps
end     %lights exist check

%  Now add noise to the values
```

Annex B: Matlab source code for Khepera Toolbox

```
%  Multipy the light data vector by some a vector of random numbers conditioned by a
%  percentage
noise_vect=khep_data(12:19).*((rand(size(khep_data(12:19)))-0.5)*noise_mux/100);
mid_vect=khep_data(12:19)+noise_vect;
%  Remove anything less than zero
mid_vect=(~(mid_vect<0)).*mid_vect;
%  Remove anything over amb_lite
mid_vect=(mid_vect>amb_lite)*amb_lite+(~(mid_vect>amb_lite)).*mid_vect;
%  Now set the value back
khep_data(12:19)=mid_vect;

%  Post any changes
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function upsv(btmap1)
%UPSV           Update Proximity Sensor Values
%
%               This function updates the simulated khepera's proximity sensor values.
%               The updated values are not passed back. This function requires a pointer to
%               the playpen bitmap being used.  Data in this file - specifically proxval
%               should be modified to reflect real world data.  Another vector, found
%               in PUTKHEP, specifies the position on the perimeter of the robot at which
%               each sensor will begin its scan.  This may also require modification to
%               agree with measured results.
%
%               See PUTKHEP for a description of the Khepera Patch userdata fields
%
%               UPSV(BTMAP1)
%               BTMAP1 - The name of the mat file playpen map
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 10/12/96
% ------------------------------------------------------------------------------


%  Define what the value is that will be read at each mm distance from the robot
proxval=[1024 1024 1024 1024 1024 1024 1024 1000 850 700 550 400 250 175 125 50 30 20 10 5];

[m n]=size(proxval);
%  The following vector defines the actual direction in which each sensor points.
sensedir_data=[90; 45; 0; 0; -45; -90; 180; 180];

%  The following vector defines the position of each sensor on the peripery of the
%  khep
sense_angles=[70 45 12 -12 -45 -70 -146 146]';

%  Get the khepera data matrix
khep_h=findobj('tag','khep');
if khep_h==[]
        return;
end
khep_data=get(khep_h,'userdata');

%  Set the proximity sensors to zero
khep_data(4:11)=zeros(1,8);

%  Get the khepera's absolute angle
theta=khep_data(3);

%  Each sensor will scan out n mm  (see below).
%  The robot radius is 28 mm.  So, add the scan to the radius to get the matrix
```

Annex B: Matlab source code for Khepera Toolbox

```
%  elements to check.
r=(1:n);

%  Each sensor will sweep out 14 degrees as a series of 3 lines.  Decrease
%  the step size (middle digit) and hence the number of scan lines to increase
%  the resolution

for delta_theta=-7:7:7

        %  The following explains how x_mod and y_mod are obtained:
        % Position of each sensor on perimeter of robot
        %28*cos((theta+khep_data(2:9,1))*pi/180)*ones(1,n)
        % Line from origin following each sensor's angle + delta angle
        %cos((theta+sensedir_data+delta_theta)*pi/180)*r
        % Finally, add the absolute x and y

        x_mod=round(28*cos((theta+sense_angles)*pi/180)*ones(1,n)+...
                cos((theta+sensedir_data+delta_theta)*pi/180)*r)+...
                khep_data(1);

        y_mod=round(28*sin((theta+sense_angles)*pi/180)*ones(1,n)+...
                sin((theta+sensedir_data+delta_theta)*pi/180)*r)+...
                khep_data(2);

        % Find any values outside legal limits and squash them to the boundaries
        if any(find(x_mod<1))|any(find(x_mod>600))
                x_mod(find(x_mod<1))=1*ones(size(find(x_mod<1)));
                x_mod(find(x_mod>600)')=600*ones(size(find(x_mod>600)));
        end
        if any(find(y_mod<1))|any(find(y_mod>500))
                y_mod(find(y_mod<1))=1*ones(size(find(y_mod<1)));
                y_mod(find(y_mod>500)')=500*ones(size(find(y_mod>500)));
        end

        % Now find the values at each point.  The result will be a column, so transpose it.
        % Note that this is a matrix we're indexing.  Therefore, the index must be
        % row/column which means y/x.
        tmp=(diag(btmap1(y_mod,x_mod)',0));
        tmp=reshape(tmp,8,n);

        % Now assign the values to the proximity vector.
        for k=1:8
                tmp2=min(find(tmp(k,:)==44));
                if tmp2~=[]
                        if proxval(tmp2)>khep_data(k+3)
                                %  Keep the larger of the two values
                                khep_data(k+3)=proxval(tmp2);
                        end
                end
        end
```

```
end

%  Now add 5% noise to the values
%  10% was a bit harsh
noise_vect=khep_data(4:11).*((200*rand(size(khep_data(4:11)))-100)/2000);
mid_vect=khep_data(4:11)+noise_vect;
%  Remove anything less than zero
mid_vect=(~(mid_vect<0)).*mid_vect;
%  Remove anything over 1024
mid_vect=(mid_vect>1024)*1024+(~(mid_vect>1024)).*mid_vect;
%  Now set the value back
khep_data(4:11)=mid_vect;

% Now set the values in the userdata section of the khepera patch
set(khep_h,'userdata',khep_data);
```

Annex B: Matlab source code for Khepera Toolbox

```
function simsense(command_str)
%SIMSENSE      Perform movement tests to determine sensor values in various positions on simulated Khep.
%
%              This function takes no arguments other than those it passes to itself in
%              recursive calls.  It produces a matrix of measurements named 'dv' (for
%              data values) who's fields are arranged as follows:
%
%                      Columns 1-2:     Khep position counter values at each measurement
%                      Columns 3-10:    Measured values of light or proximity
%                                       (Depending on test performed)
%
%              Copyright (c) 1996 Capt Rich Goyette
%                    Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------------

global dv khep_pos btmap1 testmode


% Check for the number of arguments.  If none, assume starting from scratch
if nargin == 0
  command_str = 'initialize';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%INITIALIZE TAGS
if ~strcmp(command_str,'initialize')
% Find all the handles that apply to the sensor tester window
       % Assume that the current figure contains tester
       h_fig = gcf;
       if ~strcmp(get(h_fig,'tag'),'sim_sense_tst_window')
         % If the current figure does not have the right
         % tag, find the one that does.
         h_figs = get(0,'children');
         h_fig = findobj(h_figs,'flat',...
                   'tag','sim_sense_tst_window');
         if length(h_fig) == 0
              % If the sensor tester does not exist, stop.
              error('You must start the Sensor Test application first!');
              return;
         end
       end
   % At this point we know that h_fig contains the handle
   % to the figure containing the sensor tester window.
   % Use this handle to reduce the search for tagged names (if any)
       h_mainfig = findobj(h_fig(1),'tag','sim_sense_tst_window');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
% This "Initialize" section draws all the user interface controls on the
% screen.
if strcmp(command_str,'initialize')
   % Make sure that the GUI has not been already
   % initialized in another existing figure.
   h_figs = get(0,'children');
   h_fig = findobj(h_figs,'flat',...
                              'tag','sim_sense_tst_window');
          if length(h_fig) > 0
                 figure(h_fig(1));
                 return;
          end

          % Check the graphics mode screen size.  If not in 1024X768
          % then exit with an error.
          scrn_size=get(0,'screensize');
          if ((scrn_size(3)<1024)|(scrn_size(4)<768))
                 error('This terminal must be in 1024X768 or greater mode to operate')
          end

          % Some basic colors
          tan=[.85 .69 .47];
          bkgclr=[.72 .68 .35];
          aquamarine=[.2 .53 .51];

          width=400;
          height=455;
          % Draw the figure window for the sensor tester.
          figure('units','pixels','position',[5,15,width,height],...
          'menubar','none','name','Simulated Khepera Sensor Data Collector',...
          'numbertitle','off','resize','off',...
                 'tag','sim_sense_tst_window','color',bkgclr);

          % Compose a set of axes and hide them in the wall.
          axes('units','pixels','position',[0,0,width,height],'visible','off',...
                 'xlim',[0 width],'ylim',[0 height],'aspectratio',[NaN,1],...
                 'tag','khep_base_axes')

          % Compose a group of pulldowns for various options.
          % Start by defining the lower left corner as a pair of variables (for ease
          % of later modification) and define aquamarine colour triplet.
          popupx=125;
          popupy=415;

          % COLUMN TITLES
```

```
uicontrol('style','text','position',[popupx popupy-155 120 20],...
        'string','Interface Control','backgroundcolor',bkgclr,...
        'foregroundcolor','blue');

uicontrol('style','text','position',[popupx popupy-230 120 20],...
        'string','Model Variables','backgroundcolor',bkgclr,...
        'foregroundcolor','blue');

uicontrol('style','text','position',[popupx popupy+25 120 20],...
        'string','Graph Control','backgroundcolor',bkgclr,...
        'foregroundcolor','blue');

uicontrol('style','text','position',[popupx+140 popupy+25 120 20],...
        'string','Motion Control','backgroundcolor',bkgclr,...
        'foregroundcolor','blue');



% COLUMN 1
uicontrol('style','text','position',[popupx popupy 120 20],...
        'string','Y Min','backgroundcolor',bkgclr,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','y_min','backgroundcolor',...
        aquamarine,'position',[popupx popupy-25 120 400],'string',...
        '0|50|100|150|200|250|300|350|400|450');

uicontrol('style','text','position',[popupx popupy-50 120 20],...
        'string','Grid','backgroundcolor',bkgclr,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','grid_stat','backgroundcolor',...
        aquamarine,'position',[popupx popupy-75 120 400],'string',...
        'OFF|ON');

uicontrol('style','text','position',[popupx popupy-100 120 20],...
        'string','Select Graph','backgroundcolor',bkgclr,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','dis_one','backgroundcolor',...
        aquamarine,'position',[popupx popupy-125 120 400],'string',...
        'Left 90|Left 45|Left 10|Right 10|Right 45|Right 90|Right Back|Left Back');

uicontrol('style','text','position',[popupx popupy-175 120 20],...
        'string','Test Type','backgroundcolor',bkgclr,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','tst_type','backgroundcolor',...
        aquamarine,'position',[popupx popupy-200 120 400],'string',...
        'Light Levels|Proximity');

uicontrol('style','text','position',[popupx popupy-250 70 20],...
        'string','Amb Light','backgroundcolor',bkgclr,...
        'foregroundcolor','black');
```

```
uicontrol('style','edit','position',[popupx+75 popupy-250 35 20],...
                    'min',0,'max',1,'tag','amb_lt',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(500));

uicontrol('style','text','position',[popupx popupy-275 70 20],...
          'string','Squash','backgroundcolor',bkgclr,...
          'foregroundcolor','black');
uicontrol('style','edit','position',[popupx+75 popupy-275 35 20],...
                    'min',0,'max',1,'tag','squash_con',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(0.03));

uicontrol('style','text','position',[popupx popupy-300 75 20],...
          'string','Xlate','backgroundcolor',bkgclr,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+75 popupy-300 35 20],...
                    'min',0,'max',1,'tag','xlate_con',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(175));

uicontrol('style','text','position',[popupx popupy-325 75 20],...
          'string','Max Angle','backgroundcolor',bkgclr,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+75 popupy-325 35 20],...
                    'min',0,'max',1,'tag','max_angle',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(88));

uicontrol('style','text','position',[popupx popupy-350 70 20],...
          'string','Cut Off','backgroundcolor',bkgclr,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+75 popupy-350 35 20],...
                    'min',0,'max',1,'tag','cut_off',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(50));

uicontrol('style','text','position',[popupx popupy-375 70 20],...
          'string','Noise','backgroundcolor',bkgclr,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+75 popupy-375 35 20],...
                    'min',0,'max',1,'tag','noise_lvl',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(5));
```

Annex B: Matlab source code for Khepera Toolbox

```
% COLUMN 2
uicontrol('style','text','position',[popupx+140 popupy 120 20],...
          'string','Motion Type','backgroundcolor',bkgclr,...
          'foregroundcolor','black');
uicontrol('style','popup','tag','mov_on','backgroundcolor',...
          aquamarine,'position',[popupx+140 popupy-25 120 400],'string',...
          'Linear|Rotational');

uicontrol('style','text','position',[popupx+140 popupy-50 120 20],...
          'string','Rotational Dist','backgroundcolor',bkgclr,...
          'foregroundcolor','black');
uicontrol('style','popup','tag','ang_dist','backgroundcolor',...
          aquamarine,'position',[popupx+140 popupy-75 120 400],'string',...
          '45 |90 |135 |180 |225 |270 |315 |360 |-45 |-90 |-135 |-180 |-225 |-270 |-315 |-360');

uicontrol('style','text','position',[popupx+140 popupy-100 120 20],...
          'string','Linear Dist','backgroundcolor',bkgclr,...
          'foregroundcolor','black');
uicontrol('style','popup','tag','lin_dist','backgroundcolor',...
          aquamarine,'position',[popupx+140 popupy-125 120 400],'string',...
          '5 cm|10|15|20|25|30|35|40|45|50|55|60|-5 cm|-10|-15|-20|-25|-30|-35|-40|-45|-50|-55|-60');

uicontrol('style','text','position',[popupx+140 popupy-150 120 20],...
          'string','Return to Zero','backgroundcolor',bkgclr,...
          'foregroundcolor','black');
uicontrol('style','popup','tag','mov_back','backgroundcolor',...
          aquamarine,'position',[popupx+140 popupy-175 120 400],'string',...
          'No|Yes');

%  Define an editable text box for stuff to be recorded with the data
uicontrol('style','text','position',[popupx+140 popupy-200 125 20],...
          'string','File Description','backgroundcolor',bkgclr,...
          'foregroundcolor','blue');
uicontrol('style','edit','position',[popupx+140 popupy-240 125 38],...
                    'min',0,'max',20,'tag','save_text',...
                    'backgroundcolor',[1.000 0.9700  0.7800]);


% Now define a block of buttons.
buttx=10;
butty=popupy-30;
uicontrol('style','pushbutton','position',[buttx butty 110 30],...
                    'string','Gather Data','callback','simsense("get_data");');

uicontrol('style','pushbutton','position',[buttx butty-30 110 30],...
                    'string','METAFILE','callback','simsense("meta_data");');

uicontrol('style','pushbutton','position',[buttx butty-60 110 30],...
                    'string','Display All','callback','simsense("disp_data");');
```

```
        uicontrol('style','pushbutton','position',[buttx butty-90 110 30],...
                        'string','Display One','callback','simsense("disp_one");');

        uicontrol('style','pushbutton','position',[buttx butty-120 110 30],...
                        'string','Print Graph','callback','simsense("prn_graph");');

        uicontrol('style','pushbutton','position',[buttx butty-180 110 30],...
                        'string','Exit','callback','simsense("quit_sensing");');

        uicontrol('style','pushbutton','position',[buttx butty-150 110 30],...
                        'string','Save Data','callback','simsense("save_data");');

        % Place the appropriate copyright data
        uicontrol('style','text','position',[buttx+120 10 200 20],...
                'string','Copyright (c) Rich Goyette',...
                'backgroundcolor',bkgclr,...
                'foregroundcolor','blue');



elseif strcmp(command_str,'get_data')
        tmp=findobj('tag','khep');
        if tmp==[]
                warndlg('You must have a simulated robot in the playpen');
                return;
        end
        h_tmp=findobj('tag','mov_on');
        if get(h_tmp(1),'value')==1
                % This is the linear movement data gathering section
                i=1;
                k=[0,0];
                dv=[];
                % Define a vector containing the number of pulses required to get the
                % motor to the desired position as requested in the lin_dist pulldown.
                % Use the 'value' property of lin_dist to index this vector.
                pos_vec=[625 1250 1875 2500 3125 3750 4375 5000 5625 6250 6875 7500 -625 -1250 -1875
-2500 -3125 -3750 -4375 -5000 -5625 -6250 -6875 -7500];
                h_tmp=findobj('tag','lin_dist');
                khep_pos=pos_vec(get(h_tmp(1),'value'));

                sim_spc(1,0,0);
                sim_skp(1,khep_pos,khep_pos);
                while max(abs(k(1)))<(abs(khep_pos)-1) % Subtract -50 if trouble
                        k=sim_rpc(1);
                        %  Determine the type of values desired.  IE light or proximity
                        h_tmp=findobj('tag','tst_type');
                        if get(h_tmp(1),'value')==1
                                b=sim_rls(1);
                        else
                                b=sim_rps(1);
```

```
                end
                % Fill the dv values backwards since the khep will be
                % approaching a zero distance mark.

                dv(i,1:2)=[khep_pos khep_pos]-k;
                dv(i,3:10)=b;

                i=i+1;
                %  Now change the khep's position
                testmode=1;
                kheprom(btmap1,testmode);
        end
        %  Make an adjustment to khep_pos so that ddl.m and ddlg.m can plot
        %  correctly (without leaving a space at the start of the graph since
        %  dv will contain data points up to khep_pos-625.  The graph is plotted
        %  from khep_pos/125 to zero).
        khep_pos=khep_pos-62.5;
    else
        % This is the rotational movement data gathering section.
        i=1;
        k=[0,0];
        dv=[];
        % Define a vector containing the number of pulses required to get the
        % motor to the desired angle as requested in the ang_dist pulldown.
        % Use the 'value' property of ang_dist to index this vector.
        angle_vec=[255.25 510.5 765.75 1021.0 1276.25 1531.5 1786.75 2042.0 -255.25 -510.5 -765.75
-1021.0 -1276.25 -1531.5 -1786.75 -2042.0];
        h_tmp=findobj('tag','ang_dist');
        khep_pos=angle_vec(get(h_tmp(1),'value'));
        sim_spc(1,0,0);
        sim_skp(1,khep_pos,-khep_pos);

        while max(abs(k(1)))<(abs(khep_pos)-1) % Subtract -1 if trouble
                k=sim_rpc(1);
                %  Determine the type of values desired.  IE light or proximity
                h_tmp=findobj('tag','tst_type');
                if get(h_tmp(1),'value')==1
                        b=sim_rls(1);
                else
                        b=sim_rps(1);
                end

                dv(i,1:2)=k;
                dv(i,3:10)=b;

                i=i+1;
                %  Now change the khep's position
                testmode=1;
                kheprom(btmap1,testmode);
        end
```

```
        end
        dv(1,:)=[];

        % Reset the khepera to its original position if the appropriate menu option is set
        h_tmp=findobj('tag','mov_back');
        if get(h_tmp(1),'value')==2
                pause(1);
                khep_h=findobj('tag','khep');
                khep_data=get(khep_h,'userdata');
                indx=khep_data(28);
                sim_skp(1,0,0);
                while indx~=0
                        kheprom(btmap1)
                        khep_data=get(khep_h,'userdata');
                        indx=khep_data(28);
                end
        end

elseif strcmp(command_str,'disp_data')
        % Display all of the collected data
        h1=figure('units','pixels','position',[10,10,790,700],...
        'name','Khepera Sensor Output Data',...
        'numbertitle','off','menubar','none','resize','off',...
                'tag','graph_data');
        whitebg(h1);

        h_tmp=findobj('tag','grid_stat');
        h_tmp2=findobj('tag','mov_on');
        if get(h_tmp2(1),'value')==1
                % Display linear data
                if get(h_tmp(1),'value')==1
                        ddl(dv,khep_pos);
                else
                % Display the linear data with a grid
                        ddlg(dv,khep_pos);
                end
        else
                % Display rotational data
                if get(h_tmp(1),'value')==1
                        ddr(dv,khep_pos);
                else
                % Display the rotational data with a grid
                        ddrg(dv,khep_pos);
                end
        end

elseif strcmp(command_str,'disp_one')

        h_tmp=findobj('tag','dis_one');
        disonel(dv,khep_pos,get(h_tmp(1),'value'));
```

Annex B: Matlab source code for Khepera Toolbox

```
elseif strcmp(command_str,'meta_data')
        prn_tag=findobj('tag','graph_data');
        if prn_tag ~=[]
                figure(prn_tag(1));
                print -dmeta;
        else
                return;
        end

elseif strcmp(command_str,'prn_graph')
        dataprnt;

elseif strcmp(command_str,'quit_sensing')
        delete(h_mainfig);
        return;

elseif strcmp(command_str,'save_data')
        [save_file save_path]=uiputfile('*.mat','Save Data Variable As...');
        if save_file~=0
                filepath=[save_path save_file];
                dlgstr=get(findobj('tag','save_text'),'string');
                save(filepath,'dv','dlgstr');
        else
                warndlg('Data not saved!');
        end
end
```

```
function tstsense(command_str)
%TSTSENSE      Perform movement tests to determine sensor values in various positions on real Khep.
%
%              This function takes no arguments other than those it passes to itself in
%              recursive calls.  It produces a matrix of measurements named 'dv' (for
%              data values) who's fields are arranged as follows:
%
%                      Columns 1-2:     Khep position counter values at each measurement
%                      Columns 3-10:    Measured values of light or proximity
%                                       (Depending on test performed)
%
%
%
%              Copyright (c) 1996 Capt Rich Goyette
%                    Royal Military College, Canada
%
% V1.0
% 04/11/96
% --------------------------------------------------------------------------------

global dv khep_pos

% Check for the number of arguments.  If none, assume starting from scratch
if nargin == 0
  command_str = 'initialize';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%INITIALIZE TAGS
if ~strcmp(command_str,'initialize')
% Find all the handles that apply to the sensor tester window
        % Assume that the current figure contains tester
        h_fig = gcf;
        if ~strcmp(get(h_fig,'tag'),'sense_tst_window')
          % If the current figure does not have the right
          % tag, find the one that does.
          h_figs = get(0,'children');
          h_fig = findobj(h_figs,'flat',...
                  'tag','sense_tst_window');
          if length(h_fig) == 0
                % If the sensor tester does not exist, stop.
                error('You must start the Sensor Test application first!');
                return;
          end
        end
   % At this point we know that h_fig contains the handle
   % to the figure containing the sensor tester window.
   % Use this handle to reduce the search for tagged names (if any)
        h_mainfig = findobj(h_fig(1),'tag','sense_tst_window');
```

```
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
% This "Initialize" section draws all the user interface controls on the
% screen.
if strcmp(command_str,'initialize')
   % Make sure that the GUI has not been already
   % initialized in another existing figure.
   h_figs = get(0,'children');
   h_fig = findobj(h_figs,'flat',...
                                    'tag','sense_tst_window');
          if length(h_fig) > 0
                 figure(h_fig(1));
                 return;
          end

          % Check the graphics mode screen size.  If not in 1024X768
          % then exit with an error.
          scrn_size=get(0,'screensize');
          if ((scrn_size(3)<1024)|(scrn_size(4)<768))
                 error('This terminal must be in 1024X768 or greater mode to operate')
          end

          % Some basic colors
          tan=[.85 .69 .47];
          aquamarine=[.2 .53 .51];

          width=600;
          height=400;
          % Draw the figure window for the sensor tester.
          figure('units','pixels','position',[100,230,width,height],...
          'menubar','none','name','Real Khepera Sensor Data Collector',...
          'numbertitle','off','resize','off',...
                 'tag','sense_tst_window','color',tan);

          % Compose a set of axes and hide them in the wall.
          axes('units','pixels','position',[0,0,width,height],'visible','off',...
                 'xlim',[0 width],'ylim',[0 height],'aspectratio',[NaN,1],...
                 'tag','khep_base_axes')

          % Compose a group of pulldowns for various options.
          % Start by defining the lower left corner as a pair of variables (for ease
          % of later modification) and define aquamarine colour triplet.
          popupx=125;
          popupy=340;

          % COLUMN TITLES
```

```
uicontrol('style','text','position',[popupx popupy+25 150 20],...
        'string','Interface Control','backgroundcolor',tan,...
        'foregroundcolor','blue');

uicontrol('style','text','position',[popupx+160 popupy+25 150 20],...
        'string','Graph Control','backgroundcolor',tan,...
        'foregroundcolor','blue');

uicontrol('style','text','position',[popupx+320 popupy+25 150 20],...
        'string','Motion Control','backgroundcolor',tan,...
        'foregroundcolor','blue');

% COLUMN 1
uicontrol('style','text','position',[popupx popupy 150 20],...
        'string','Port ID','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','port_num','backgroundcolor',...
        aquamarine,'position',[popupx popupy-25 150 400],'string',...
        'port 1|port 2|port 3|port 4');

uicontrol('style','text','position',[popupx popupy-50 150 20],...
        'string','Test Type','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','tst_type','backgroundcolor',...
        aquamarine,'position',[popupx popupy-75 150 400],'string',...
        'Light Levels|Proximity');

%  Define an editable text box for stuff to be recorded with the data
uicontrol('style','text','position',[popupx popupy-125 125 20],...
        'string','File Description','backgroundcolor',tan,...
        'foregroundcolor','blue');
uicontrol('style','edit','position',[popupx popupy-165 125 38],...
                'min',0,'max',20,'tag','real_save_text',...
                'backgroundcolor',[1.000 0.9700  0.7800]);


% COLUMN 2
uicontrol('style','text','position',[popupx+160 popupy 150 20],...
        'string','Y Min','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','y_min','backgroundcolor',...
        aquamarine,'position',[popupx+160 popupy-25 150 400],'string',...
        '0|50|100|150|200|250|300|350|400|450');

uicontrol('style','text','position',[popupx+160 popupy-50 150 20],...
        'string','Grid','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','grid_stat','backgroundcolor',...
        aquamarine,'position',[popupx+160 popupy-75 150 400],'string',...
        'OFF|ON');
```

```
uicontrol('style','text','position',[popupx+160 popupy-100 150 20],...
        'string','Select Graph','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','dis_one','backgroundcolor',...
        aquamarine,'position',[popupx+160 popupy-125 150 400],'string',...
        'Left 90|Left 45|Left 10|Right 10|Right 45|Right 90|Right Back|Left Back');



% COLUMN 3
uicontrol('style','text','position',[popupx+320 popupy 150 20],...
        'string','Motion Type','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','mov_on','backgroundcolor',...
        aquamarine,'position',[popupx+320 popupy-25 150 400],'string',...
        'Linear|Rotational');

uicontrol('style','text','position',[popupx+320 popupy-50 150 20],...
        'string','Rotational Dist','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','ang_dist','backgroundcolor',...
        aquamarine,'position',[popupx+320 popupy-75 150 400],'string',...
        '45 |90 |135 |180 |225 |270 |315 |360 |-45 |-90 |-135 |-180 |-225 |-270 |-315 |-360');

uicontrol('style','text','position',[popupx+320 popupy-100 150 20],...
        'string','Linear Dist','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','lin_dist','backgroundcolor',...
        aquamarine,'position',[popupx+320 popupy-125 150 400],'string',...
        '5 cm|10|15|20|25|30|35|40|45|50|55|60|-5 cm|-10|-15|-20|-25|-30|-35|-40|-45|-50|-55|-60');

uicontrol('style','text','position',[popupx+320 popupy-150 150 20],...
        'string','Return to Zero','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','mov_back','backgroundcolor',...
        aquamarine,'position',[popupx+320 popupy-175 150 400],'string',...
        'No|Yes');

uicontrol('style','text','position',[popupx+320 popupy-210 150 20],...
        'string','Max Speed','backgroundcolor',tan,...
        'foregroundcolor','black');

uicontrol('style','text','position',[popupx+340 popupy-230 40 20],...
        'string','Left','backgroundcolor',tan,...
        'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+340 popupy-250 35 20],...
                'min',0,'max',1,'tag','left_speed',...
```

```
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(20));

uicontrol('style','text','position',[popupx+410 popupy-230 40 20],...
          'string','Right','backgroundcolor',tan,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+410 popupy-250 35 20],...
                    'min',0,'max',1,'tag','right_speed',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(20));

uicontrol('style','text','position',[popupx+320 popupy-270 150 20],...
          'string','Max Acceleration','backgroundcolor',tan,...
          'foregroundcolor','black');

uicontrol('style','text','position',[popupx+340 popupy-290 40 20],...
          'string','Left','backgroundcolor',tan,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+340 popupy-310 35 20],...
                    'min',0,'max',1,'tag','left_acc',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(64));

uicontrol('style','text','position',[popupx+410 popupy-290 40 20],...
          'string','Right','backgroundcolor',tan,...
          'foregroundcolor','black');

uicontrol('style','edit','position',[popupx+410 popupy-310 35 20],...
                    'min',0,'max',1,'tag','right_acc',...
                    'backgroundcolor',[1.000 0.9700  0.7800],...
                    'string',num2str(64));


% Now define a block of buttons.
buttx=10;
butty=popupy-30;
uicontrol('style','pushbutton','position',[buttx butty 110 30],...
                    'string','Gather Data','callback','tstsense("get_data");');

uicontrol('style','pushbutton','position',[buttx butty-30 110 30],...
                    'string','METAFILE','callback','tstsense("meta_data");');

uicontrol('style','pushbutton','position',[buttx butty-60 110 30],...
                    'string','Display All','callback','tstsense("disp_data");');

uicontrol('style','pushbutton','position',[buttx butty-90 110 30],...
                    'string','Display One','callback','tstsense("disp_one");');
```

```
        uicontrol('style','pushbutton','position',[buttx butty-150 110 30],...
                        'string','Save Data','callback','tstsense("save_data");');

        uicontrol('style','pushbutton','position',[buttx butty-120 110 30],...
                        'string','Print Graph','callback','tstsense("prn_graph");');

        uicontrol('style','pushbutton','position',[buttx butty-180 110 30],...
                        'string','Exit','callback','tstsense("quit_sensing");');

        % Place the appropriate copyright data
        uicontrol('style','text','position',[buttx+220 10 200 20],...
                'string','Copyright (c) Rich Goyette',...
                'backgroundcolor',tan,...
                'foregroundcolor','blue');

elseif strcmp(command_str,'get_data')
        %  Set the speed controller so that the robot proceeds reasonably slow during
        %  the sampling period
        tstwin_handle=findobj('tag','sense_tst_window');
        spd_left=str2num(get(findobj(tstwin_handle,'tag','left_speed'),'string'));
        spd_right=str2num(get(findobj(tstwin_handle,'tag','right_speed'),'string'));
        acc_left=str2num(get(findobj(tstwin_handle,'tag','left_acc'),'string'));
        acc_right=str2num(get(findobj(tstwin_handle,'tag','right_acc'),'string'));
        port_id=get(findobj('tag','port_num'),'value');
        cspc(port_id,spd_left,acc_left,spd_right,acc_right);
        h_tmp=findobj('tag','mov_on');
        if get(h_tmp(1),'value')==1
                % This is the linear movement data gathering section
                port_id=get(findobj('tag','port_num'),'value');
                i=1;
                k=[0,0];
                dv=[];
                % Define a vector containing the number of pulses required to get the
                % motor to the desired position as requested in the lin_dist pulldown.
                % Use the 'value' property of lin_dist to index this vector.
                pos_vec=[625 1250 1875 2500 3125 3750 4375 5000 5625 6250 6875 7500 -625 -1250 -1875
-2500 -3125 -3750 -4375 -5000 -5625 -6250 -6875 -7500];
                h_tmp=findobj('tag','lin_dist');
                khep_pos=pos_vec(get(h_tmp(1),'value'));

                spc(port_id,0,0);
                skp(port_id,khep_pos,khep_pos);
                while max(abs(k))<(abs(khep_pos)) % Subtract -50 if trouble
                        k=rpc(port_id);
                        %  Determine the type of values desired.  IE light or proximity
                        h_tmp=findobj('tag','tst_type');
                        if get(h_tmp(1),'value')==1
                                b=rls(port_id);
                        else
                                b=rps(port_id);
```

```
                        end
                        % Fill the dv values backwards since the khep will be
                        % approaching a zero distance mark.
                        dv(i,1:2)=[khep_pos khep_pos]-k;
                        dv(i,3:10)=b;
                        % Periodicly, abnormal values come back.  Smooth them out
                        % by assigning them the previous value (unless there was
                        % no previous value - then just assign zero)
                        if max(abs(k))>3*abs(khep_pos)
                                if (i>1)
                                        dv(i,1:2)=dv(i-1,1:2);
                                        k=[0 0];
                                else
                                        dv(i,1:2)=[khep_pos khep_pos];
                                        k=[0 0];
                                end
                        end
                        if max(b)>2000
                                if (i>1)
                                        dv(i,3:10)=dv(i-1,3:10);
                                else
                                        dv(i,3:10)=[0 0 0 0 0 0 0 0];
                                end
                        end
                        i=i+1;
                end

        else
                % This is the rotational movement data gathering section.
                port_id=get(findobj('tag','port_num'),'value');
                i=1;
                k=[0,0];
                dv=[];
                % Define a vector containing the number of pulses required to get the
                % motor to the desired angle as requested in the ang_dist pulldown.
                % Use the 'value' property of ang_dist to index this vector.
                angle_vec=[255 510 765 1021 1276 1531 1786 2042 -255 -510 -765 -1021 -1276 -1531 -1786
-2042];
                h_tmp=findobj('tag','ang_dist');
                khep_pos=angle_vec(get(h_tmp(1),'value'));
                spc(port_id,0,0);
                skp(port_id,khep_pos,-khep_pos);
                while max(abs(k))<(abs(khep_pos)) % Subtract -50 if trouble
                        k=rpc(port_id);
                        %  Determine the type of values desired.  IE light or proximity
                        h_tmp=findobj('tag','tst_type');
                        if get(h_tmp(1),'value')==1
                                b=rls(port_id);
                        else
                                b=rps(port_id);
```

```
                        end

                        dv(i,1:2)=k;
                        dv(i,3:10)=b;
                        % Periodic abnormal values come back.  Smooth them out
                        % by assigning them the previous value (unless there was
                        % no previous value - then just assign zero)
                        if max(abs(k))>3*abs(khep_pos)
                                if (i>1)
                                        dv(i,1:2)=dv(i-1,1:2);
                                        k=[0 0];
                                else
                                        dv(i,1:2)=[0 0];
                                        k=[0 0];
                                end
                        end
                        if max(b)>2000
                                if (i>1)
                                        dv(i,3:10)=dv(i-1,3:10);
                                else
                                        dv(i,3:10)=[0 0 0 0 0 0 0 0];
                                end
                        end
                        i=i+1;
                end

        end
        % Reset the khepera to its original position if the appropriate menu option is set
        h_tmp=findobj('tag','mov_back');
        if get(h_tmp(1),'value')==2
                pause(1);
                skp(port_id,0,0);
        end

elseif strcmp(command_str,'disp_data')
        % Display all of the collected data
        h1=figure('units','pixels','position',[10,10,790,700],...
        'name','Khepera Sensor Output Data',...
        'numbertitle','off','menubar','none','resize','off',...
                'tag','graph_data');
        whitebg(h1);

        h_tmp=findobj('tag','grid_stat');
        h_tmp2=findobj('tag','mov_on');
        if get(h_tmp2(1),'value')==1
                % Display linear data
                if get(h_tmp(1),'value')==1
                        ddl(dv,khep_pos);
                else
                % Display the linear data with a grid
```

```
                        ddlg(dv,khep_pos);
                end
        else
                % Display rotational data
                if get(h_tmp(1),'value')==1
                        ddr(dv,khep_pos);
                else
                % Display the rotational data with a grid
                        ddrg(dv,khep_pos);
                end
        end

elseif strcmp(command_str,'disp_one')

        h_tmp=findobj('tag','dis_one');
        disonel(dv,khep_pos,get(h_tmp(1),'value'));


elseif strcmp(command_str,'meta_data')
        prn_tag=findobj('tag','graph_data');
        if prn_tag ~=[]
                figure(prn_tag(1));
                print -dmeta;
        else
                return;
        end

elseif strcmp(command_str,'prn_graph')
        dataprnt;

elseif strcmp(command_str,'quit_sensing')
        delete(h_mainfig);
        return;

elseif strcmp(command_str,'save_data')
        [save_file save_path]=uiputfile('*.mat','Save Data Variable As...');
        if save_file~=0
                filepath=[save_path save_file];
                dlgstr=get(findobj('tag','real_save_text'),'string');
                save(filepath,'dv','dlgstr');
        else
                warndlg('Data not saved!');
        end
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function datameta()
%DATAMETA   METAfile the current plot
%
%                   Low Level Simulator Object.  This object copies the contents of the
%                   topmost figure window with the tag 'graph_data' to the Windows
%                   clipboard.
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------


prn_tag=findobj('tag','graph_data');
if prn_tag ~=[]
        figure(prn_tag(1));
        print -dmeta;
else
        return;
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function dataprnt()
%DATAPRNT    Print the current plot
%
%                   Low Level Simulator Object.  This object prints the contents of the
%                   topmost figure window with the tag 'graph_data'.  Modify the print
%                   command and/or print arguments below to suit your system.
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------



prn_tag=findobj('tag','graph_data');
if prn_tag ~=[]
        figure(prn_tag(1));
        print -dbj10e;
else
        return;
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function ddl(dv,khep_pos)
% DDL           Display Linear Data with no grid
%
%               Low Level Simulator Object.  This object is used by TSTSENSE and
%               SIMSENSE for the production of a figure containing eight graphs,
%               one for each sensor.  This particular function provides the graph
%               without a grid for data taken from a linear test trajectory.
%
%               DDL(DV,KHEP_POS)
%               DV - a matrix of data values taken during a test run
%                       Fields in the matrix as follows:
%                       Columns 1-2:    Khep position counter values at each measurement
%                       Columns 3-10:   Measured values of light or proximity
%                                       (Depending on test performed)
%               KHEP_POS -  The max linear or rotational distance travelled during
%                               the test (in pulses, 1 pulse=0.08mm)
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% --------------------------------------------------------------------------------


% Use the following vector and the 'value' property of the Y Axis UICONTROL
% to set the y axis range
axes_vec=[0 50 100 150 200 250 300 350 400 450];
h_tmp=findobj('tag','y_min');
y_min_val=get(h_tmp(1),'value');

% Define the x and y bottom corners of the graphs
xcorn=75;
ycorn=40;

% Condition linear data for plotting
ydat_L90=fliplr(dv(:,3)');
ydat_L45=fliplr(dv(:,4)');
ydat_L10=fliplr(dv(:,5)');
ydat_R10=fliplr(dv(:,6)');
ydat_R45=fliplr(dv(:,7)');
ydat_R90=fliplr(dv(:,8)');
ydat_BR=fliplr(dv(:,9)');
ydat_BL=fliplr(dv(:,10)');
xdat=abs(fliplr(dv(:,1)'./125));

%  Find out what kind of test is happening and set the ymax accordingly
h_tmp=findobj('tag','tst_type');
if get(h_tmp(1),'value')==1
        ymax=520;
```

```
else
        ymax=1030;
end
ymin=axes_vec(y_min_val);

h1=axes('units','pixels','position',[xcorn ycorn 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisBL','xdir','reverse');
h2=text('string','Left Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BL,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL90','xdir','reverse');
h2=text('string','Left 90 Sensor ','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L90,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL45','xdir','reverse');
h2=text('string','Left 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L45,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL10','xdir','reverse');
h2=text('string','Left 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L10,'color','red');

xdis=390;
h1=axes('units','pixels','position',[xcorn+xdis ycorn 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL90','xdir','reverse');
h2=text('string','Right Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BR,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR90','xdir','reverse');
h2=text('string','Right 90 Sensor','fontsize',8,'fontweight','bold',...
```

```
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R90,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR45','xdir','reverse');
h2=text('string','Right 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R45,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR10','xdir','reverse');
h2=text('string','Right 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R10,'color','red');


return;
```

Annex B: Matlab source code for Khepera Toolbox

```
function ddlg(dv,khep_pos)
% DDLG                  Display Linear Data with a grid
%
%               Low Level Simulator Object.  This object is used by TSTSENSE and
%               SIMSENSE for the production of a figure containing eight graphs,
%               one for each sensor.  This particular function provides the graph
%               with a grid for data taken from a linear test trajectory.
%
%               DDLG(DV,KHEP_POS)
%               DV - a matrix of data values taken during a test run
%                       Fields in the matrix as follows:
%                       Columns 1-2:    Khep position counter values at each measurement
%                       Columns 3-10:   Measured values of light or proximity
%                                       (Depending on test performed)
%               KHEP_POS -  The max linear or rotational distance travelled during
%                               the test (in pulses, 1 pulse=0.08mm)
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------


% Use the following vector and the 'value' property of the Y Axis UICONTROL
% to set the y axis range
axes_vec=[0 50 100 150 200 250 300 350 400 450];
h_tmp=findobj('tag','y_min');
y_min_val=get(h_tmp(1),'value');

% Define the x and y bottom corners of the graphs
xcorn=75;
ycorn=40;

% Condition linear data for plotting
ydat_L90=fliplr(dv(:,3)');
ydat_L45=fliplr(dv(:,4)');
ydat_L10=fliplr(dv(:,5)');
ydat_R10=fliplr(dv(:,6)');
ydat_R45=fliplr(dv(:,7)');
ydat_R90=fliplr(dv(:,8)');
ydat_BR=fliplr(dv(:,9)');
ydat_BL=fliplr(dv(:,10)');
xdat=abs(fliplr(dv(:,1)'./125));


%  Find out what kind of test is happening and set the ymax accordingly
h_tmp=findobj('tag','tst_type')
if get(h_tmp(1),'value')==1
        ymax=520;
```

```
else
        ymax=1030;
end
ymin=axes_vec(y_min_val);

h1=axes('units','pixels','position',[xcorn ycorn 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisLB','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Left Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BL,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL90','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Left 90 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L90,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL90','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Left 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L45,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisL10','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Left 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L10,'color','red');

xdis=390;
h1=axes('units','pixels','position',[xcorn+xdis ycorn 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisRB','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Right Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BR,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR90','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Right 90 Sensor','fontsize',8,'fontweight','bold',...
```

```
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R90,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR45','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Right 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R45,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax],...
        'tag','axisR10','xdir','reverse','xgrid','on','ygrid','on','xminorgrid','on','yminorgrid','on');
h2=text('string','Right 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R10,'color','red');

return;
```

Annex B: Matlab source code for Khepera Toolbox

```
function ddr(dv,khep_pos)
% DDR           Display Rotational Data without a grid
%
%               Low Level Simulator Object.  This object is used by TSTSENSE and
%               SIMSENSE for the production of a figure containing eight graphs,
%               one for each sensor.  This particular function provides the graph
%               without a grid for data taken from a rotational test trajectory.
%
%               DDR(DV,KHEP_POS)
%               DV - a matrix of data values taken during a test run
%                       Fields in the matrix as follows:
%                       Columns 1-2:    Khep position counter values at each measurement
%                       Columns 3-10:   Measured values of light or proximity
%                                       (Depending on test performed)
%               KHEP_POS -  The max linear or rotational distance travelled during
%                               the test (in pulses, 1 pulse=0.08mm)
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------


% Use the following vector and the 'value' property of the Y Axis UICONTROL
% to set the y axis range
axes_vec=[0 50 100 150 200 250 300 350 400 450];
h_tmp=findobj('tag','y_min');
y_min_val=get(h_tmp(1),'value');


% Define the x and y bottom corners of the graphs
xcorn=75;
ycorn=40;

% Condition rotational data for plotting
ydat_L90=dv(:,3)';
ydat_L45=dv(:,4)';
ydat_L10=dv(:,5)';
ydat_R10=dv(:,6)';
ydat_R45=dv(:,7)';
ydat_R90=dv(:,8)';
ydat_BR=dv(:,9)';
ydat_BL=dv(:,10)';
xdat=abs(dv(:,1)'*360/2042);



%  Find out what kind of test is happening and set the ymax accordingly
```

```
h_tmp=findobj('tag','tst_type');
if get(h_tmp(1),'value')==1
        ymax=520;
else
        ymax=1030;
end
ymin=axes_vec(y_min_val);


% Display the data without a grid
h1=axes('units','pixels','position',[xcorn ycorn 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisBL');
h2=text('string','Left Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BL,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL90');
h2=text('string','Left 90 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L90,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL45');
h2=text('string','Left 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L45,'color','red');

h1=axes('units','pixels','position',[xcorn ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL10');
h2=text('string','Left 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L10,'color','red');

xdis=390;
h1=axes('units','pixels','position',[xcorn+xdis ycorn 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisBR');
h2=text('string','Right Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BR,'color','red');
```

Annex B: Matlab source code for Khepera Toolbox

```
h1=axes('units','pixels','position',[xcorn+xdis ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR90');
h2=text('string','Right 90 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R90,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR45');
h2=text('string','Right 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R45,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR10');
h2=text('string','Right 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R10,'color','red');


return;
```

Annex B: Matlab source code for Khepera Toolbox

```
function ddrg(dv,khep_pos)
% DDRG                    Display Rotational Data with grid
%
%                 Low Level Simulator Object.  This object is used by TSTSENSE and
%                 SIMSENSE for the production of a figure containing eight graphs,
%                 one for each sensor.  This particular function provides the graph
%                 with a grid for data taken from a rotational test trajectory.
%
%                 DDRG(DV,KHEP_POS)
%                 DV - a matrix of data values taken during a test run
%                         Fields in the matrix as follows:
%                         Columns 1-2:    Khep position counter values at each measurement
%                         Columns 3-10:   Measured values of light or proximity
%                                         (Depending on test performed)
%                 KHEP_POS -  The max linear or rotational distance travelled during
%                                 the test (in pulses, 1 pulse=0.08mm)
%                 Copyright (c) 1996 Capt Rich Goyette
%                       Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------


% Use the following vector and the 'value' property of the Y Axis UICONTROL
% to set the y axis range
axes_vec=[0 50 100 150 200 250 300 350 400 450];
h_tmp=findobj('tag','y_min');
y_min_val=get(h_tmp(1),'value');

% Define the x and y bottom corners of the graphs
xcorn=75;
ycorn=40;

% Condition rotational data for plotting
ydat_L90=dv(:,3)';
ydat_L45=dv(:,4)';
ydat_L10=dv(:,5)';
ydat_R10=dv(:,6)';
ydat_R45=dv(:,7)';
ydat_R90=dv(:,8)';
ydat_BR=dv(:,9)';
ydat_BL=dv(:,10)';
xdat=abs(dv(:,1)'*360/2042);



%  Find out what kind of test is happening and set the ymax accordingly
h_tmp=findobj('tag','tst_type');
if get(h_tmp(1),'value')==1
```

```
        ymax=520;
else
        ymax=1030;
end
ymin=axes_vec(y_min_val);


h1=axes('units','pixels','position',[xcorn ycorn 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisBL','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Left Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BL','color','red');

h1=axes('units','pixels','position',[xcorn ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL90','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Left 90 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L90','color','red');

h1=axes('units','pixels','position',[xcorn ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL45','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Left 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L45','color','red');

h1=axes('units','pixels','position',[xcorn ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisL10','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Left 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_L10','color','red');

xdis=390;
h1=axes('units','pixels','position',[xcorn+xdis ycorn 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisBR','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Right Back Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
```

Annex B: Matlab source code for Khepera Toolbox

```
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_BR,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+170 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR90','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Right 90 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R90,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+340 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR45','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Right 45 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R45,'color','red');

h1=axes('units','pixels','position',[xcorn+xdis ycorn+510 300 120],...
        'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax],...
        'tag','axisR10','xgrid','on','ygrid','on','xminorticks','on',...
        'yminorticks','on');
h2=text('string','Right 10 Sensor','fontsize',8,'fontweight','bold',...
        'rotation',90);
set(h1,'ylabel',h2);
line('xdata',xdat,'ydata',ydat_R10,'color','red');

return;
```

Annex B: Matlab source code for Khepera Toolbox

```
%DISONEL        Display data graph for information on one sensor.
%
%               Low Level Simulator Object.  This object is used by TSTSENSE and
%               SIMSENSE for the production of a figure containing a single graph
%               of one sensor, chosen from within the TESTSENSE or SIMSENSE
%               window.
%
%               DISONEL(DV,KHEP_POS,WHICH_ONE)
%               DV - a matrix of data values taken during a test run
%                       Fields in the matrix as follows:
%                       Columns 1-2:    Khep position counter values at each measurement
%                       Columns 3-10:   Measured values of light or proximity
%                                       (Depending on test performed)
%               KHEP_POS -  The max linear or rotational distance travelled during
%                               the test (in pulses, 1 pulse=0.08mm)
%               WHICH_ONE - A value passed from the caller specifying which sensor
%                               is to appear in the graph
%
%               Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------


% Use the following vector and the 'value' property of the Y Axis UICONTROL
% to set the y axis range
axes_vec=[0 50 100 150 200 250 300 350 400 450];
h_tmp=findobj('tag','y_min');
y_min_val=get(h_tmp(1),'value');

%  Find out what kind of test is happening and set the ymax accordingly
h_tmp=findobj('tag','tst_type');
if get(h_tmp(1),'value')==1
        ymax=520;
else
        ymax=1030;
end
ymin=axes_vec(y_min_val);


h1=figure;
set(h1,'tag','graph_data');
set(h1,'name','Graph Data for One Sensor');

whitebg(h1);

h_tmp=findobj('tag','mov_on');
if get(h_tmp(1),'value')==1
```

```
% Linear display.  Set up the x and y data accordingly
% Note that the entries of "display one" must be the same as the numbering
% returned by rps because of the following line.
ydat=fliplr(dv(:,which_one+2)');
xdat=abs(fliplr(dv(:,1)'./120));
plot(xdat,ydat);
h2=findobj(h1,'type','axes');
set(h2,'xdir','reverse',...
        'xlim',[0 abs(khep_pos/120)],'ylim',[ymin ymax]);

%  Check to see if a grid is desired
h_tmp=findobj('tag','grid_stat');
if get(h_tmp(1),'value')==1
        % No grid
else
        % Grid wanted
        set(h2,'xgrid','on','ygrid','on');
        set(h2,'xminorticks','on','yminorticks','on');
end
h_tmp=findobj('tag','tst_type');
if get(h_tmp(1),'value')==1
        if which_one==1
                title('Light Level Data for Left 90 Degree Sensor and Linear Movement');
        elseif which_one==2
                title('Light Level Data for Left 45 Degree Sensor and Linear Movement');
        elseif which_one==3
                title('Light Level Data for Left 10 Degree Sensor and Linear Movement');
        elseif which_one==4
                title('Light Level Data for Right 10 Degree Sensor and Linear Movement');
        elseif which_one==5
                title('Light Level Data for Right 45 Degree Sensor and Linear Movement');
        elseif which_one==6
                title('Light Level Data for Right 90 Degree Sensor and Linear Movement');
        elseif which_one==7
                title('Light Level Data for Back Right Sensor and Linear Movement');
        elseif which_one==8
                title('Light Level Data for Back Left Sensor and Linear Movement');
        end
        ylabel('Inverse Light Level');
        xlabel('Distance From Start to Finish (cm)');
else
        if which_one==1
                title('Proximity Data for Left 90 Degree Sensor and Linear Movement');
        elseif which_one==2
                title('Proximity Data for Left 45 Degree Sensor and Linear Movement');
        elseif which_one==3
                title('Proximity Data for Left 10 Degree Sensor and Linear Movement');
        elseif which_one==4
                title('Proximity Data for Right 10 Degree Sensor and Linear Movement');
        elseif which_one==5
```

```
                title('Proximity Data for Right 45 Degree Sensor and Linear Movement');
            elseif which_one==6
                title('Proximity Data for Right 90 Degree Sensor and Linear Movement');
            elseif which_one==7
                title('Proximity Data for Back Right Sensor and Linear Movement');
            elseif which_one==8
                title('Proximity Data for Back Left Sensor and Linear Movement');
            end
            ylabel('Proximity Detector Level');
            xlabel('Distance From Start to Finish (cm)');
        end

else
        % Rotational display.  Set up the x and y data accordingly
        % Note that the entries of "display one" must be the same as the numbering
        % returned by rps because of the following line.
        ydat=dv(:,which_one+2)';
        xdat=abs(dv(:,1)'*360/2072);
        plot(xdat,ydat);
        h2=findobj(h1,'type','axes');
        set(h2,'xlim',[0 abs(khep_pos*360/2072)],'ylim',[ymin ymax]);
        %  Check to see if a grid is desired
        h_tmp=findobj('tag','grid_stat');
        if get(h_tmp,'value')==1
            % No grid
        else
            % Grid wanted
            set(h2,'xgrid','on','ygrid','on');
            set(h2,'xminorticks','on','yminorticks','on');
        end
        h_tmp=findobj('tag','tst_type');
        if get(h_tmp(1),'value')==1
            if which_one==1
                title('Light Level Data for Left 90 Degree Sensor and Rotational Movement');
            elseif which_one==2
                title('Light Level Data for Left 45 Degree Sensor and Rotational Movement');
            elseif which_one==3
                title('Light Level Data for Left 10 Degree Sensor and Rotational Movement');
            elseif which_one==4
                title('Light Level Data for Right 10 Degree Sensor and Rotational Movement');
            elseif which_one==5
                title('Light Level Data for Right 45 Degree Sensor and Rotational Movement');
            elseif which_one==6
                title('Light Level Data for Right 90 Degree Sensor and Rotational Movement');
            elseif which_one==7
                title('Light Level Data for Back Right Sensor and Rotational Movement');
            elseif which_one==8
                title('Light Level Data for Back Left Sensor and Rotational Movement');
            end
            ylabel('Inverse Light Level');
```

```
            xlabel('Distance From Start to Finish (degrees)');
        else
            if which_one==1
                title('Proximity Data for Left 90 Degree Sensor and Rotational Movement');
            elseif which_one==2
                title('Proximity Data for Left 45 Degree Sensor and Rotational Movement');
            elseif which_one==3
                title('Proximity Data for Left 10 Degree Sensor and Rotational Movement');
            elseif which_one==4
                title('Proximity Data for Right 10 Degree Sensor and Rotational Movement');
            elseif which_one==5
                title('Proximity Data for Right 45 Degree Sensor and Rotational Movement');
            elseif which_one==6
                title('Proximity Data for Right 90 Degree Sensor and Rotational Movement');
            elseif which_one==7
                title('Proximity Data for Back Right Sensor and Rotational Movement');
            elseif which_one==8
                title('Proximity Data for Back Left Sensor and Rotational Movement');
            end
            ylabel('Proximity Detector Level');
            xlabel('Distance From Start to Finish (degrees)');
        end
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function khepbase(arg3,arg4,arg5)
%KHEPBASE    Create a visual display for the Khepera Base unit in operation.
%            arg3=8 element row vector containing proximity data from sensors
%            arg4=8 element row vector containing ambient light data from sensors
%            arg5=2 element row vector containing motor speed data
%
%            Usage:
%
%            khepbase;
%                    Initializes the display window and bargraphs.  This
%                    command MUST be issued before any of the commands below.
%                    It only needs to be issued once
%
%
%            khepbase(prox_data, light_data, motor_speed);
%                    Updates the proximity, ambient light, and speed
%                    bars.
%
%            Copyright (c) 1996 Capt Rich Goyette
%                  Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------

global xoffset yoffset
xoffset=200;
yoffset=200;
% Check for the number of arguments.  If none, assume starting from scratch
if nargin == 0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
% This "Initialize" section draws all the user interface controls on the
% screen.

   % Make sure that the GUI has not been already
   % initialized in another existing figure.
   h_figs = get(0,'children');
   h_fig = findobj(h_figs,'flat',...
                           'tag','khep_base_visualizer');
         if length(h_fig) > 0
                  figure(h_fig(1));
                  return;
         end

         width=400;
         height=400;
         % Draw the figure window for the visualizer.
```

```
figure('units','pixels','position',[100,230,width,height],...
'menubar','none','name','Khepera Sensor Visualizer V1.0',...
'numbertitle','off','resize','off',...
        'tag','khep_base_visualizer');

% Compose a set of axes and hide them in the wall.
axes('units','pixels','position',[0,0,width,height],'visible','off',...
        'xlim',[0 width],'ylim',[0 height],'aspectratio',[NaN,1],...
        'tag','khep_base_axes')

% Store the maximum and minimum x and y positions of the playpen region
% in the user data region so that they are available to all functions.
% Format:  [xmin ymin xmax ymax]
w_info=[0 0 width height];
h_tmp=findobj('tag','khep_base_visualizer');
set(h_tmp,'userdata',w_info);

% Draw the khep
radius=28;
scale_factor=90/28;

bodyx=scale_factor*radius*(cos([0:10:360]*(pi/180)))+xoffset;
bodyy=scale_factor*radius*(sin([0:10:360]*(pi/180)))+yoffset;
patch('XData',bodyx,'YData',bodyy,'Facecolor',[.2 .53 .51],...
        'edgecolor',[.31 .57 .35],'linewidth',2);

% Place the sensors
patch('XData',scale_factor*[-25 -23 -23 -25]+xoffset,...
        'YData',scale_factor*[8 8 12 12]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[-19 -18 -15 -16]+xoffset,...
        'YData',scale_factor*[19.5 18 20.5 22]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[-8 -4 -4 -8]+xoffset,...
        'YData',scale_factor*[24 24 26 26]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[8 4 4 8]+xoffset,...
        'YData',scale_factor*[24 24 26 26]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[19 18 15 16]+xoffset,...
        'YData',scale_factor*[20.5 18 19.5 22]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[25 23 23 25]+xoffset,...
        'YData',scale_factor*[8 8 12 12]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[-12 -8 -8 -12]+xoffset,...
        'YData',scale_factor*[-25 -25 -23 -23]+yoffset,...
        'Facecolor','blue');
patch('XData',scale_factor*[12 8 8 12]+xoffset,...
        'YData',scale_factor*[-25 -25 -23 -23]+yoffset,...
```

```
            'Facecolor','blue');

% Place the wheels
patch('XData',scale_factor*[-27 -25 -25 -27]+xoffset,...
            'YData',scale_factor*[-7 -7 7 7]+yoffset,...
            'Facecolor',[.68 .66 .43],...
            'Edgecolor',[.68 .66 .43]);
patch('XData',scale_factor*[27 25 25 27]+xoffset,...
            'YData',scale_factor*[-7 -7 7 7]+yoffset,...
            'Facecolor',[.68 .66 .43],...
            'Edgecolor',[.68 .66 .43]);

% Place background patches
patch('XData',[-30 -10 -10 -30]+xoffset,...
            'YData',[95 95 195 195]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[30 10 10 30]+xoffset,...
            'YData',[95 95 195 195]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[-70 -50 -50 -70]+xoffset,...
            'YData',[85 85 185 185]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[70 50 50 70]+xoffset,...
            'YData',[85 85 185 185]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[-110 -90 -90 -110]+xoffset,...
            'YData',[35 35 135 135]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[110 90 90 110]+xoffset,...
            'YData',[35 35 135 135]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[-40 -20 -20 -40]+xoffset,...
            'YData',[-195 -195 -95 -95]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[40 20 20 40]+xoffset,...
            'YData',[-195 -195 -95 -95]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[-125 -115 -115 -125]+xoffset,...
            'YData',[-70 -70 30 30]+yoffset,...
            'facecolor','black','edgecolor','white');
patch('XData',[125 115 115 125]+xoffset,...
            'YData',[-70 -70 30 30]+yoffset,...
            'facecolor','black','edgecolor','white');

% Place the initial bargraphs and save the tags

patch('XData',[-29 -21 -21 -29]+xoffset,...
            'YData',[95 95 96 96]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s2prox','edgecolor','red');
patch('XData',[-19 -11 -11 -19]+xoffset,...
```

```
            'YData',[95 95 96 96]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s2light','edgecolor','yellow');
patch('XData',[19 11 11 19]+xoffset,...
            'YData',[95 95 96 96]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s3prox','edgecolor','red');
patch('XData',[29 21 21 29]+xoffset,...
            'YData',[95 95 96 96]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s3light','edgecolor','yellow');
patch('XData',[-69 -61 -61 -69]+xoffset,...
            'YData',[85 85 86 86]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s1prox','edgecolor','red');
patch('XData',[-59 -51 -51 -59]+xoffset,...
            'YData',[85 85 86 86]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s1light','edgecolor','yellow');
patch('XData',[59 51 51 59]+xoffset,...
            'YData',[85 85 86 86]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s4prox','edgecolor','red');
patch('XData',[69 61 61 69]+xoffset,'erasemode','xor',...
            'YData',[85 85 86 86]+yoffset,...
            'facecolor','yellow','tag','s4light','edgecolor','yellow');
patch('XData',[-109 -101 -101 -109]+xoffset,...
            'YData',[35 35 36 36]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s0prox','edgecolor','red');
patch('XData',[-99 -91 -91 -99]+xoffset,...
            'YData',[35 35 36 36]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s0light','edgecolor','yellow');
patch('XData',[99 91 91 99]+xoffset,...
            'YData',[35 35 36 36]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s5prox','edgecolor','red');
patch('XData',[109 101 101 109]+xoffset,...
            'YData',[35 35 36 36]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s5light','edgecolor','yellow');
patch('XData',[-39 -31 -31 -39]+xoffset,...
            'YData',[-195 -195 -194 -194]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s7prox','edgecolor','red');
patch('XData',[-29 -21 -21 -29]+xoffset,...
            'YData',[-195 -195 -194 -194]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s7light','edgecolor','yellow');
patch('XData',[29 21 21 29]+xoffset,...
            'YData',[-195 -195 -194 -194]+yoffset,'erasemode','xor',...
            'facecolor','red','tag','s6prox','edgecolor','red');
patch('XData',[39 31 31 39]+xoffset,...
            'YData',[-195 -195 -194 -194]+yoffset,'erasemode','xor',...
            'facecolor','yellow','tag','s6light','edgecolor','yellow');
patch('XData',[-124 -116 -116 -124]+xoffset,...
            'YData',[-70 -70 -71 -71]+yoffset,'erasemode','xor',...
            'facecolor','magenta','tag','motorleft','edgecolor','magenta');
patch('XData',[124 116 116 124]+xoffset,...
            'YData',[-70 -70 -71 -71]+yoffset,'erasemode','xor',...
            'facecolor','magenta','tag','motorright','edgecolor','magenta');
```

```
        return;

else
        % Find all the handles that apply to the KhepBase visualizer
        % Assume that the current figure contains visualizer
        h_fig = gcf;
        if ~strcmp(get(h_fig,'tag'),'khep_base_visualizer')
          % If the current figure does not have the right
          % tag, find the one that does.
          h_figs = get(0,'children');
          h_fig = findobj(h_figs,'flat',...
                    'tag','khep_base_visualizer');
          if length(h_fig) == 0
                % If the visualizer does not exist, stop.
                error('You must initialize the base window before first use');
                return;
          end
        end
   % At this point we know that h_fig contains the handle
   % to the figure containing the khepbase visual gui.
   % Use this handle to reduce the search for tagged names.
        h_kbaxes = findobj(h_fig(1),'tag','khep_base_axes');
        h_khepbase = findobj(h_fig(1),'tag','khep_base_visualizer');
        h_s1prox = findobj(h_fig(1),'tag','s1prox');
        h_s2prox = findobj(h_fig(1),'tag','s2prox');
        h_s3prox = findobj(h_fig(1),'tag','s3prox');
        h_s4prox = findobj(h_fig(1),'tag','s4prox');
        h_s5prox = findobj(h_fig(1),'tag','s5prox');
        h_s6prox = findobj(h_fig(1),'tag','s6prox');
        h_s7prox = findobj(h_fig(1),'tag','s7prox');
        h_s0prox = findobj(h_fig(1),'tag','s0prox');
        h_s1light = findobj(h_fig(1),'tag','s1light');
        h_s2light = findobj(h_fig(1),'tag','s2light');
        h_s3light = findobj(h_fig(1),'tag','s3light');
        h_s4light = findobj(h_fig(1),'tag','s4light');
        h_s5light = findobj(h_fig(1),'tag','s5light');
        h_s6light = findobj(h_fig(1),'tag','s6light');
        h_s7light = findobj(h_fig(1),'tag','s7light');
        h_s0light = findobj(h_fig(1),'tag','s0light');
        h_motorleft=findobj(h_fig(1),'tag','motorleft');
        h_motorright=findobj(h_fig(1),'tag','motorright');

        %Condition the inputs

        % Set any of the proximity sensors over 1024 to 1024
        arg3=abs((abs(arg3)>1024)*1024+(~(abs(arg3)>1024)).*abs(arg3));

        % Set any of the light sensors over 500 to 500
        arg4=abs((abs(arg4)>500)*500+(~(abs(arg4)>500)).*abs(arg4));
```

```
% Set any of the speed over 30 to 30
arg5=abs((abs(arg5)>30)*30+(~(abs(arg5)>30)).*abs(arg5));

%Now update the bargraphs by modifying the x and y data properties
%of each of the patch objects representing bars.

%Proximity first
tmp=round([36 36]+yoffset+(100/1023)*arg3(1));
set(h_s0prox,'YData',[35+yoffset 35+yoffset tmp(1) tmp(2)]);


tmp=round([86 86]+yoffset+(100/1023)*arg3(2));
set(h_s1prox,'YData',[85+yoffset 85+yoffset tmp(1) tmp(2)]);


tmp=round([96 96]+yoffset+(100/1023)*arg3(3));
set(h_s2prox,'YData',[95+yoffset 95+yoffset tmp(1) tmp(2)]);


tmp=round([96 96]+yoffset+(100/1023)*arg3(4));
set(h_s3prox,'YData',[95+yoffset 95+yoffset tmp(1) tmp(2)]);


tmp=round([86 86]+yoffset+(100/1023)*arg3(5));
set(h_s4prox,'YData',[85+yoffset 85+yoffset tmp(1) tmp(2)]);


tmp=round([36 36]+yoffset+(100/1023)*arg3(6));
set(h_s5prox,'YData',[35+yoffset 35+yoffset tmp(1) tmp(2)]);


tmp=round([-194 -194]+yoffset+(100/1023)*arg3(7));
set(h_s6prox,'YData',[-195+yoffset -195+yoffset tmp(1) tmp(2)]);


tmp=round([-194 -194]+yoffset+(100/1023)*arg3(8));
set(h_s7prox,'YData',[-195+yoffset -195+yoffset tmp(1) tmp(2)]);

%Light next
tmp=round([36 36]+yoffset+(100/500)*arg4(1));
set(h_s0light,'YData',[35+yoffset 35+yoffset tmp(1) tmp(2)]);


tmp=round([86 86]+yoffset+(100/500)*arg4(2));
set(h_s1light,'YData',[85+yoffset 85+yoffset tmp(1) tmp(2)]);


tmp=round([96 96]+yoffset+(100/500)*arg4(3));
set(h_s2light,'YData',[95+yoffset 95+yoffset tmp(1) tmp(2)]);
```

```
        tmp=round([96 96]+yoffset+(100/500)*arg4(4));
        set(h_s3light,'YData',[95+yoffset 95+yoffset tmp(1) tmp(2)]);


        tmp=round([86 86]+yoffset+(100/500)*arg4(5));
        set(h_s4light,'YData',[85+yoffset 85+yoffset tmp(1) tmp(2)]);


        tmp=round([36 36]+yoffset+(100/500)*arg4(6));
        set(h_s5light,'YData',[35+yoffset 35+yoffset tmp(1) tmp(2)]);


        tmp=round([-194 -194]+yoffset+(100/500)*arg4(7));
        set(h_s6light,'YData',[-195+yoffset -195+yoffset tmp(1) tmp(2)]);


        tmp=round([-194 -194]+yoffset+(100/500)*arg4(8));
        set(h_s7light,'YData',[-195+yoffset -195+yoffset tmp(1) tmp(2)]);

        %Finally motors
        tmp=round([-70 -70]+yoffset+100/30*(abs(arg5(1))));
        set(h_motorleft,'YData',[-70+yoffset -70+yoffset tmp(1) tmp(2)]);

        tmp=round([-70 -70]+yoffset+100/30*(abs(arg5(2))));
        set(h_motorright,'YData',[-70+yoffset -70+yoffset tmp(1) tmp(2)]);

end
```

Annex B: Matlab source code for Khepera Toolbox

```
function modpen2(command_str)
%MODPEN2       Modify the playpen according to the action in the simulator window
%
%              Low Level Simulator Object.  This object actions the command
%              string issued with its call for the ROBORUN interface window.
%
%              Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% ------------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ACTION MOUSE BUTTON
% This command string is the call-back for windowbuttondown of the playpen

if strcmp(command_str,'screen_action')
        % Find out what action is currently selected in the simulator pulldown
        % and then execute the appropriate recursion.
        tmp1=findobj('tag','robo_runner');
        if tmp1==[]
                roborun2;
                return;
        end
        h_acts=findobj('tag','act_stat');
        if get(h_acts,'value')==1
                modpen2('placekhep');
        end
        if get(h_acts,'value')==2
                modpen2('move_khep');
        end
        if get(h_acts,'value')==3
                modpen2('rot_left_khep');
        end
        if get(h_acts,'value')==4
                modpen2('rot_right_khep');
        end
        if get(h_acts,'value')==5
                modpen2('del_khep');
        end

        if get(h_acts,'value')==6
                modpen2('putlamp');
        end
        if get(h_acts,'value')==7
                modpen2('del_lamp');
        end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS PUTTING SOMETHING ON THE CANVAS
% That is, if the action selection pulldown says "Place Khep"

elseif strcmp(command_str,'placekhep')
	h_ppen=findobj('tag','khep_simwindow');

	% If the button press was normal (left mouse button) then
	% lets place the khep, otherwise, lets delete the khep (short cut)
	if strcmp(get(h_ppen,'selectiontype'),'normal')

		% Find out where the pointer is.  mp represents "mouse position"
		mp=get(h_ppen,'currentpoint');
		% Get the width and heigth of the playpen window
		tmp=get(h_ppen,'position');
		width=tmp(3);
		height=tmp(4);
		if (mp(1)<28|mp(1)>(width-28)|mp(2)<28|mp(2)>(height-28))
			% The khep will fall outside of the window.
			return;
		end
		putkhep(mp(1),mp(2));
	elseif strcmp(get(h_ppen,'selectiontype'),'extend')
		modpen2('del_khep');
		return;
	end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS ROTATING THE SIMULATED KHEP LEFT

elseif strcmp(command_str,'rot_left_khep')
	rotkhep(15);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS ROTATING THE SIMULATED KHEP RIGHT

elseif strcmp(command_str,'rot_right_khep')
	rotkhep(-15);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS REMOVING SOMETHING FROM THE CANVAS
% That is, if the action selection pulldown says "delete object"
```

```
elseif strcmp(command_str,'del_khep')
	h_ppen=findobj('tag','khep_simwindow');
	% Find out if there's a khep and delete it.
	if findobj('tag','khep')
		delete(findobj('tag','khep'));
		return;
	end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS PUTTING A LAMP ON THE SCREEN

elseif strcmp(command_str,'putlamp')

	% Now find out where the pointer is.  mp represents "mouse position"
	h_ppen=findobj('tag','khep_simwindow');
	mp=get(h_ppen,'currentpoint');
	%  Now make the light default 'on' by setting the last bit of userdata to 1
	udat=[mp 1];
	% Draw a Small Lamp
	% Let the radius be 10 mm
	% No error checking.  You can put the lamp anywhere in the window and
	% on top of anything.
	lampx=8*(cos([0:10:360]*(pi/180)));
	lampy=8*(sin([0:10:360]*(pi/180)));
	patch('xdata',lampx+mp(1),...
		'ydata',lampy+mp(2),...
		'facecolor','yellow','edgecolor','yellow','erasemode','xor',...
		'tag','lamp','userdata',udat)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS REMOVING A LAMP FROM THE CANVAS
% That is, if the action selection pulldown says "delete lamp"
% This function is a bit more complex than normal.  When "currentobj" is
% querried (to find out if we're on a lamp with the mouse), it always
% returns the handle of the bitmap in the window and never anything else.
% So, find all objects with the tag 'lamp' and then compute the one
% that is closest to the current mouse position (Euclidian sense), and
% then delete that lamp.

elseif strcmp(command_str,'del_lamp')
	h_ppen=findobj('tag','khep_simwindow');
	mp2=get(h_ppen,'currentpoint');
	lamp_locs=findobj('tag','lamp');
	if lamp_locs ~=[]
		for i=1:length(lamp_locs)
			udat=get(lamp_locs(i),'userdata');
			tmp=udat(1:2);
			tmp2=tmp-mp2;
```

```
                    closest(i,:)=sqrt(tmp2(1)^2+tmp2(2)^2);
                end
                obj_to_delete=min(closest);
                idx=find(closest'==obj_to_delete);
                delete(lamp_locs(idx));
        end
end
```

Annex B: Matlab source code for Khepera Toolbox

```
function play_pen(command_str)
%PLAY_PEN      Create a virtual playpen for the Khepera robot.
%
%              This function implements a GUI for creating a virtual playpen
%              for the Khepera robot.  The matlab image command is used to
%              store the virtual lab.  The lab is stored in a 600X500 matrix
%              and each element is interpreted as a one millimeter square
%              area of an equivalent real lab.
%
%              This function is intended to take no user arguments, although
%              it is recursive and calls itself quite often.
%
%              This function was written first and was used as the learning
%              environment for this author.  Hence, the programming style is
%              not modular.  It will be broken down into smaller, simpler
%              functions when hell freezes over.
%
%              Copyright (c) 1996 Capt Rich Goyette
%                     Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------

% The following variables are only needed while an object is being moved.  Hence,
% there is little concern that they will be erased by the clear command so they
% will not be stored in the main figure's "userdata" property or any other long
% term variable.
global CUR_OBJ CUR_OBJ_TYPE OLD_XY


% Check for the number of arguments.  If none, assume starting from scratch
if nargin == 0
  command_str = 'initialize';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
%INITIALIZE TAGS
if ~strcmp(command_str,'initialize')
% Find all the handles that apply to the Virtual Playpen
        % Assume that the current figure contains the Virtual Playpen
        h_fig = gcf;
        if ~strcmp(get(h_fig,'tag'),'playpen')
          % If the current figure does not have the right
          % tag, find the one that does.
          h_figs = get(0,'children');
          h_fig = findobj(h_figs,'flat',...
                  'tag','playpen');
          if (length(h_fig) == 0)
```

```
                    % If the playpen editor box does not exist, then
                    % initialize it. Then run the command string
                    % that was originally requested (if there was one).
                    warndlg('Playpen editor workspace closed.  Restart Playpen editor');
                    return;
            end
          end
   % At this point we know that h_fig contains the handle
   % to the figure containing the playpen gui.
   % Use this handle to reduce the search for tagged names.
          h_axes = findobj(h_fig(1),'tag','playpen_axes');
          h_wall = findobj(h_fig(1),'tag','playpen_wall');
          h_bkgnd = findobj(h_fig(1),'tag','bkgnd_patch');
          h_ppen = findobj(h_fig(1),'tag','playpen');


   % Now find the handles associated with the menubar
   % First, find the right tag
          h_figs = get(0,'children');
          h_fig = findobj(h_figs,'flat',...
                        'tag','playpen_menubar');
            if (length(h_fig) == 0)
                    % If the playpen editor box does not exist, then
                    % initialize it. Then run the command string
                    % that was originally requested (if there was one).
                    warndlg('Menu window missing.  Restart playpen editor');
                    return;
            end
          h_obj = findobj(h_fig(1),'tag','playpen_obj');
          h_acts = findobj(h_fig(1),'tag','playpen_acts');
          h_menubar = findobj(h_fig(1),'tag','playpen_menubar');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
% This "Initialize" section draws all the user interface controls on the
% screen.
if strcmp(command_str,'initialize')

          % Check the graphics mode screen size.  If not in 1024X768
          % then exit with an error.
          scrn_size=get(0,'screensize');
          if ((scrn_size(3)<1024)|(scrn_size(4)<768))
                    error('This terminal must be in 1024X768 or greater mode to operate')
          end

          % Define the standard background color.
          tan=[.85 .69 .47];
```

```
aquamarine=[.2 .53 .51];

% Draw the figure window for the playpen.
figure('units','pixels','position',[305,150,600,500],...
'menubar','none','name','Khepera Virtual Playpen Designer V1.0',...
'numbertitle','off','resize','off',...
        'tag','playpen',...
        'windowbuttondownfcn','play_pen("screen_action")');

% Compose a set of axes and hide them in the wall.
axes('units','pixels','position',[0,0,600,500],'visible','off',...
        'xlim',[0 600],'ylim',[0 500],'aspectratio',[NaN,1],...
        'tag','playpen_axes')

% Draw the menubar on the side.  Remember that this must be flexible
figure('units','pixels','position',[50,150,250,500],...
        'menubar','none','name','Design Palette',...
'numbertitle','off','resize','off','color',tan,...
        'tag','playpen_menubar')
% Store the maximum and minimum x and y positions of the playpen region
% in the user data region so that they are available to all functions.
% Also, store a pair of "twiddle factors" that will be required for
% boundary checking later.  The format is:
% [xmin ymin xmax ymax twiddlex twiddley]
w_info=[0 0 600 500 8 9];
h_tmp=findobj('tag','playpen');
set(h_tmp,'userdata',w_info);


% Compose a boundary for the virtual lab.  Note that for some reason, the
% maximum size that matrix that can be captured using getframe seems to be
% 600X500 so 10cm is lost from the original lab spec.  Also, the linewidth
% is going to be thick, so knock 5 pixels off of each max and add
% a 5 to each min.  This will bring the centre of the line inwards on all
% sides so that the whole line can be seen.  If we did not add and subtract
% 5 pixels, only half the line would be visible.  Note that this means we
% need the twiddle factors when we're checking to see if an object is
% out of bounds.

xvertices=[w_info(1)+5,w_info(3)-5,w_info(3)-5,w_info(1)+5,w_info(1)+5];
yvertices=[w_info(2)+5,w_info(2)+5,w_info(4)-5,w_info(4)-5,w_info(2)+5];
% Swap figures and draw border
figure(findobj('tag','playpen'));
line('xdata',xvertices,'ydata',yvertices,'color','blue','linewidth',8,...
        'tag','playpen_wall','erasemode','xor');

% Swap figures and draw buttons and pulldowns
figure(findobj('tag','playpen_menubar'));
% Compose a pulldown menu of all the possible objects that can be placed.
% Start by defining the top left corner as a pair of variables (for ease
```

```
% of later modification) and define aquamarine colour triplet.
popupx=75;
popupy=500-50;

% Place the text title
uicontrol('style','text','position',[popupx popupy+25 110 20],...
        'string','Object Types','backgroundcolor',tan,...
        'foregroundcolor','black');
% Place the menu
uicontrol('style','popup','tag','playpen_obj','backgroundcolor',...
        aquamarine,'position',[popupx popupy 110 400],'string',...
        'Wall 0|Wall -45|Wall 45|Wall 90');


% Compose a pulldown menu of all the possible actions that can be performed.
% Put it beside the objects menu, so index off of objects.

uicontrol('style','text','position',[popupx popupy-25 110 20],...
        'string','Actions','backgroundcolor',tan,...
        'foregroundcolor','black');
uicontrol('style','popup','tag','playpen_acts','backgroundcolor',...
        aquamarine,'position',[popupx popupy-50 110 400],'string',...
        'Add Object|Delete Object|Move Object|Rotate Object');

% Now define a block of buttons.  Not all will be used immediately.
buttx=75;
butty=500-150;
uicontrol('style','pushbutton','position',[buttx butty 110 30],...
                'string','Scan Playpen','callback','play_pen("scan_data");');
uicontrol('style','pushbutton','position',[buttx butty-90 110 30],...
                'string','Help','callback','play_pen("help_data");');
uicontrol('style','pushbutton','position',[buttx butty-30 110 30],...
                'string','Save Playpen','callback','play_pen("save_data");');
uicontrol('style','pushbutton','position',[buttx butty-60 110 30],...
                'string','Load Playpen','callback','play_pen("restore_data");');
uicontrol('style','pushbutton','position',[buttx butty-120 110 30],...
                'string','Blank','enable','off');
uicontrol('style','pushbutton','position',[buttx butty-150 110 30],...
                'string','Exit','callback','play_pen("leave");');

% Place the appropriate copyright data
uicontrol('style','text','position',[popupx popupy-430 110 20],...
        'string','Copyright (c)','backgroundcolor',tan,...
        'foregroundcolor','blue');
uicontrol('style','text','position',[popupx popupy-450 110 20],...
        'string','Rich Goyette','backgroundcolor',tan,...
        'foregroundcolor','blue');

return;
```

Annex B: Matlab source code for Khepera Toolbox

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% ACTION MOUSE BUTTON
% This command string is the call-back for windowbuttondown of the main figure

elseif strcmp(command_str,'screen_action')

	% Find out what action is currently selected in the pulldown menu
	% and then execute the appropriate recursion.

	if get(h_acts,'value')==1
		play_pen('putobj');
	end
	if get(h_acts,'value')==2
		play_pen('delobj');
	end
	if get(h_acts,'value')==3
		play_pen('movobj');
	end
	if get(h_acts,'value')==4
		warndlg('Object rotation is not yet available','INFORMATION');
		% Add a rotation command or call-back later.
	end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% Scan the playpen.
elseif strcmp(command_str,'scan_data')

	% Now save the bitmap and colourmap using the save command (which
	% puts the data in .mat format) for use by the main calling program.
	% save custmap btmap1 clrmap1;
	%  Make a bitmap and a colourmap of the desired region.

	[btmap1,clrmap1]=getframe(h_ppen,[0,0,600,500]);
	[save_file save_path]=uiputfile('*.mat','SCANFILE NAME...');
	if save_file~=0
		filepath=[save_path save_file];
		save(filepath,'btmap1','clrmap1');
	else
		warndlg('File not saved!');
	end
	return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% Save the playpen.
```

```
elseif strcmp(command_str,'save_data')
        % Go through and find every patch object and save its xdata and ydata
        % properties.
        obj_tags=findobj(h_ppen,'tag','block');
        for i=1:length(obj_tags)
                objectsxdata(i,:)=get(obj_tags(i),'xdata');
                objectsydata(i,:)=get(obj_tags(i),'ydata');
        end
        [save_file save_path]=uiputfile('*.mat','Save Playpen As...');
        if save_file~=0
                filepath=[save_path save_file];
                save(filepath,'objectsxdata','objectsydata');
        else
                warndlg('File not saved!');
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Restore the playpen.
elseif strcmp(command_str,'restore_data')
        [save_file save_path]=uigetfile('*.mat','Load Saved Playpen...');
        if save_file~=0
                filepath=[save_path save_file];
                load(filepath);
                [m,n]=size(objectsxdata);
                figure(h_ppen);
                for i=1:m
                        patch('xdata',objectsxdata(i,:),...
                        'ydata',objectsydata(i,:),...
                        'facecolor','blue','edgecolor','blue','erasemode','xor',...
                        'tag','block')
                end
        else
                warndlg('File not loaded!');
        end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Leave the Playpen Creator.
elseif strcmp(command_str,'leave')
        %h=findobj('tag','h_ppen');
        %h2=findobj('tag','h_menubar');
        delete(h_ppen);
        delete(h_menubar);
        return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%
% Print some help information
elseif strcmp(command_str,'help_data')
        helpdlg('No help currently available');
        return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS PUTTING SOMETHING ON THE CANVAS
% That is, if the action selection pulldown says "add object"

elseif strcmp(command_str,'putobj')
        % Find out if the cursor location is valid.  First get the allowable boundaries.
        w_inf=get(h_ppen,'userdata');
        % Now find out where the pointer is.  mp represents "mouse position"
        mp=get(h_ppen,'currentpoint');
        if (mp(1)<w_inf(1)|mp(1)>w_inf(3)|mp(2)<w_inf(2)|mp(2)>w_inf(4))
                % Leave if the pointer's not in the playpen
                errordlg('You cannot place an object outside of the playpen','Stop!!');
                return;
        end

        % Now find out what kind of object is being placed
        if get(h_obj,'value')==1
                % Draw a small block at 0 degrees at the specified location
                % if the block fits within legal boundaries.
                % Let sbx0 be small block x width from centre for zero degree rotation
                % and similarly for y (for ease of changing block size later).
                % Note that w_inf(5)and w_inf(6) are the twiddle factors used to
                % compensate for the width of the boundary line
                sbx0=25;
                sby0=7;
                % Check the boundaries...
                if (mp(1)-sbx0-w_inf(5)<w_inf(1)...
                        |mp(1)+sbx0+w_inf(5)>w_inf(3)...
                        |mp(2)-sby0-w_inf(6)<w_inf(2)...
                        |mp(2)+sby0+w_inf(6)>w_inf(4))
                        % No good.  Leave.
                        return;
                else
                % Boundaries good.  Place the patch.
                patch('xdata',[mp(1)-sbx0 mp(1)+sbx0 mp(1)+sbx0 mp(1)-sbx0 mp(1)-sbx0],...
                        'ydata',[mp(2)-sby0 mp(2)-sby0 mp(2)+sby0 mp(2)+sby0 mp(2)-sby0],...
                        'facecolor','blue','edgecolor','blue','erasemode','xor',...
                        'tag','block')
                end
        end

        % Draw a small block at -45 degrees at the specified location
        if get(h_obj,'value')==2
```

```
            sbx0=21;
            sby0=21;
            % Check the boundaries...
            if (mp(1)-sbx0-w_inf(5)<w_inf(1)...
                        |mp(1)+sbx0+w_inf(5)>w_inf(3)...
                        |mp(2)-sby0-w_inf(6)<w_inf(2)...
                        |mp(2)+sby0+w_inf(6)>w_inf(4))
                        % No good.  Leave.
                        return;
            else
            % Boundaries good.  Place the patch.
            patch('xdata',[mp(1)+8 mp(1)+18 mp(1)-8 mp(1)-18 mp(1)+8],...
                        'ydata',[mp(2)-18 mp(2)-8 mp(2)+18 mp(2)+8 mp(2)-18],...
                        'facecolor','blue','edgecolor','blue','erasemode','xor',...
                        'tag','block')
            end

end



% Draw a small block at 45 degrees at the specified location
if get(h_obj,'value')==3
            sbx0=21;
            sby0=21;
            % Check the boundaries...
            if (mp(1)-sbx0-w_inf(5)<w_inf(1)...
                        |mp(1)+sbx0+w_inf(5)>w_inf(3)...
                        |mp(2)-sby0-w_inf(6)<w_inf(2)...
                        |mp(2)+sby0+w_inf(6)>w_inf(4))
                        % No good.  Leave.
                        return;
            else
            % Boundaries good.  Place the patch.
            patch('xdata',[mp(1)-8 mp(1)+18 mp(1)+8 mp(1)-18 mp(1)-8],...
                        'ydata',[mp(2)-18 mp(2)+8 mp(2)+18 mp(2)-8 mp(2)-18],...
                        'facecolor','blue','edgecolor','blue','erasemode','xor',...
                        'tag','block')
            end

end

if get(h_obj,'value')==4
            % Draw a small block at 90 degrees at the specified location
            % Let sbx90 be small block x width from centre for zero degree rotation
            % and similarly for y (for ease of changing block size later)
            sbx90=7;
            sby90=25;
            if (mp(1)-sbx90-w_inf(5)<w_inf(1)...
                        |mp(1)+sbx90+w_inf(5)>w_inf(3)...
```

```
                        |mp(2)-sby90-w_inf(6)<w_inf(2)...
                        |mp(2)+sby90+w_inf(6)>w_inf(4))
                        return;
                else
                patch('xdata',[mp(1)-sbx90 mp(1)+sbx90 mp(1)+sbx90 mp(1)-sbx90 mp(1)-sbx90],...
                        'ydata',[mp(2)-sby90 mp(2)-sby90 mp(2)+sby90 mp(2)+sby90 mp(2)-sby90],...
                        'facecolor','blue','edgecolor','blue','erasemode','xor',...
                        'tag','block')
                end
        end

        % Add more objects here by extending the object menu string in the initialization
        % section and then searching for them here using the appropriate 'value'.
        % Don't forget to do error checking on the boundaries of the patch before
        % placement.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS REMOVING SOMETHING FROM THE CANVAS
% That is, if the action selection pulldown says "delete object"
elseif strcmp(command_str,'delobj')
        if ~strcmp(get(get(h_ppen,'currentobj'),'type'),'patch')
                return;
        end
        delete(get(h_ppen,'currentobj'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%% FUNCTION NOT COMPLETE
% IF THE DESIRED ACTION WAS ROTATING SOMETHING ON THE CANVAS
% That is, if the action selection pulldown says "rotate object"
elseif strcmp(command_str,'rotobj')
        CUR_OBJ=get(h_ppen,'currentobj');
        CUR_OBJ_TYPE=get(CUR_OBJ,'type');
        %  If a patch wasn't selected, then quit
        if ~strcmp(CUR_OBJ_TYPE,'patch')
                        return;
        end
        set(CUR_OBJ,'erasemode','xor');
        %  If the object was a block, compute the centre and rotate
        if strcmp(get(CUR_OBJ,'tag'),'block')
        end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% IF THE DESIRED ACTION WAS MOVING SOMETHING ON THE CANVAS
% That is, if the action selection pulldown says "move object"
```

```
elseif strcmp(command_str,'movobj')
        % Find out the object type and handle being pointed at.  Store in global
        % variables for later use.
        CUR_OBJ=get(h_ppen,'currentobj');
        CUR_OBJ_TYPE=get(CUR_OBJ,'type');
        % If the button press was normal (left mouse button) then:
        if strcmp(get(h_ppen,'selectiontype'),'normal')
                % If the object being pointed to is not a patch, leave
                if ~strcmp(CUR_OBJ_TYPE,'patch')
                        return;
                end
                %Set up moving the object
                set(CUR_OBJ,'erasemode','xor');
                set(h_ppen,'pointer','fleur');
                set(h_ppen,'windowbuttonupfcn','play_pen(''move_done'')');
                set(h_ppen,'windowbuttonmotionfcn','play_pen(''move_obj'')');
                set(CUR_OBJ,'selected','on');
                % Since patches rely on xdata and ydata, we have to use the
                % differential between mouse points.  Keep the old mouse location.
                OLD_XY=get(h_ppen,'currentpoint');

        % Now, if the middle mouse was pressed, add the currently selected object.
        % This is a hotkey action.  It only works in "Move object" mode.
        elseif strcmp(get(h_ppen,'selectiontype'),'alt')
                play_pen('putobj');
                return;

        % If the right mouse was pressed, delete the currently selected object.
        % Another hotkey.
        elseif strcmp(get(h_ppen,'selectiontype'),'extend')
                play_pen('delobj');
                return;
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% This is the callback that was set for the windowbuttonmotionfcn above.
elseif strcmp(command_str,'move_obj')
        % Get the required data to set compute the objects new x and y data.
        w_inf=get(h_ppen,'userdata');
        new_xy=get(h_ppen,'currentpoint');
        old_x=get(CUR_OBJ,'xdata');
        old_y=get(CUR_OBJ,'ydata');
        % If the object is a block, it will have five xdata and ydata entries
        % in a regular order.
        if strcmp(get(CUR_OBJ,'tag'),'block')
                % Compute the vertices and see if they exceed the allowable
                % boundaries.  Note that the min/max arguement can also be used
                % below instead of the absolute vertices as shown.
```

```
        if((old_x(1)+new_xy(1)-OLD_XY(1))<(w_inf(1)+w_inf(5))|...
            (old_x(2)+new_xy(1)-OLD_XY(1))>(w_inf(3)-w_inf(5))|...
            (old_y(1)+new_xy(2)-OLD_XY(2))<(w_inf(2)+w_inf(6))|...
            (old_y(3)+new_xy(2)-OLD_XY(2))>(w_inf(4)-w_inf(6)))
            return;
        end
    end
    if strcmp(get(CUR_OBJ,'tag'),'lamp')
        % Compute the vertices and see if they exceed the allowable
        % boundaries.
        if(min(old_x+new_xy(1)-OLD_XY(1))<(w_inf(1)+w_inf(5))|...
            max(old_x+new_xy(1)-OLD_XY(1))>(w_inf(3)-w_inf(5))|...
            min(old_y(1)+new_xy(2)-OLD_XY(2))<(w_inf(2)+w_inf(6))|...
            max(old_y(3)+new_xy(2)-OLD_XY(2))>(w_inf(4)-w_inf(6)))
            return;
        end
    end
    % If they object is still in the safe movement region, then update the
    % x and y data.
    set(CUR_OBJ,'xdata',old_x+new_xy(1)-OLD_XY(1));
    set(CUR_OBJ,'ydata',old_y+new_xy(2)-OLD_XY(2));
    OLD_XY=new_xy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clean up after a move.
elseif strcmp(command_str,'move_done')
    set(CUR_OBJ,'selected','off');
    set(CUR_OBJ,'erasemode','normal');
    set(h_ppen,'windowbuttonupfcn','','windowbuttonmotionfcn','');
    set(h_ppen,'pointer','arrow');

end
```

Annex B: Matlab source code for Khepera Toolbox

```
function roborun2(command_str)
%ROBORUN2    Open a GUI interface for running an algorithm on real or simulated Khep.
%
%                   This function presents a GUI interface that gives control over the
%                   number of cycles and number of runs of a particular algorithm.  It
%                   takes no arguments other than those that it feeds to itself in
%                   recursive calls.
%
%                   This function also provides the option to save data between runs
%                   along with a text description of the data.  To run an algorithm,
%                   it must be placed in the "program" space of the companion m-file,
%                   'template.m'.  The companion function provides the necessary code
%                   for a seamless algorithm plug-in.
%
%                   This function is also the launching point for many of the utilities
%                   such as tstsense (real khepera sensor testing), simsense (virtual
%                   khepera sensor testing), the khep playpen creator, and many
%                   user defined functions.
%
%                   Copyright (c) 1996 Capt Rich Goyette
%                         Royal Military College, Canada
%
% V1.0
% 04/11/96
% ---------------------------------------------------------------------------

global ctrl_file ctrl_path save_file save_path filepath btmap1

%  Variables of interest
global xdata_stat ydata_stat W1 W2 b1 b2 a1_previous
% Check for the number of arguments.  If none, assume starting from scratch
if nargin == 0
  command_str = 'initialize';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%INITIALIZE TAGS
if ~strcmp(command_str,'initialize')
% Find all the handles that apply to the roborunner
        % Assume that the current figure contains runner
        h_fig = gcf;
        if ~strcmp(get(h_fig,'tag'),'robo_runner')
          % If the current figure does not have the right
          % tag, find the one that does.
          h_figs = get(0,'children');
          h_fig = findobj(h_figs,'flat',...
                    'tag','robo_runner');
          if length(h_fig) == 0
                % If the roborunner does not exist, stop.
```

```
                error('You must start the RoboRun application first!');
                return;
          end
        end
   % At this point we know that h_fig contains the handle
   % to the figure containing the roborunner.
   % Use this handle to reduce the search for tagged names (if any)
        h_mainfig = findobj(h_fig(1),'tag','robo_runner');
        h_sim_window = findobj('tag','khep_simwindow');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE THE GUI
% This "Initialize" section draws all the user interface controls on the
% screen.
if strcmp(command_str,'initialize')
   % Make sure that the GUI has not been already
   % initialized in another existing figure.
   h_figs = get(0,'children');
   h_fig = findobj(h_figs,'flat',...
                        'tag','robo_runner');
        if length(h_fig) > 0
                figure(h_fig(1));
                return;
        end

        % Check the graphics mode screen size.  If not in 1024X768
        % then exit with an error.
        scrn_size=get(0,'screensize');
        if ((scrn_size(3)<1024)|(scrn_size(4)<768))
                error('This terminal must be in 1024X768 or greater mode to operate')
        end

        % Some basic colors
        tan=[.85 .69 .47];
        aquamarine=[.2 .53 .51];

        width=400;
        height=300;
        % Draw the figure window for the roborunner.
        figure('units','pixels','position',[5,408,width,height],...
        'menubar','none','name','RoboRunner Window',...
        'numbertitle','off','resize','off',...
                'tag','robo_runner','color',tan);

        % Compose a set of axes and hide them in the wall.
        axes('units','pixels','position',[0,0,width,height],'visible','off',...
                'xlim',[0 width],'ylim',[0 height],'aspectratio',[NaN,1],...
```

```
            'tag','robo_run_axes')

        % Define the upper left corner of all UICONTROLS in the window frame
        buttx=10;
        butty=height-40;

        % COLUMN TITLES
        % ================================================================
        uicontrol('style','text','position',[buttx butty-8 110 40],...
                'string','Algorithm Control','backgroundcolor',tan,...
                'foregroundcolor','blue');

        uicontrol('style','text','position',[buttx+120 butty-8 110 40],...
                'string','Cycle Control','backgroundcolor',tan,...
                'foregroundcolor','blue');

        % FIRST COLUMN OF UICONTROL ITEMS

        uicontrol('style','pushbutton','position',[buttx butty-45 110 30],...
                        'string','LOAD','callback','roborun2("get_file");');

        uicontrol('style','pushbutton','position',[buttx butty-75 110 30],...
                        'string','INITIALIZE','callback','roborun2("init_file");');

        uicontrol('style','pushbutton','position',[buttx butty-105 110 30],...
                        'string','Blank','enable','off');

        uicontrol('style','pushbutton','position',[buttx butty-135 110 30],...
                        'string','Blank','enable','off');

        uicontrol('style','pushbutton','position',[buttx butty-165 110 30],...
                        'string','RUN','backgroundcolor','red',...
                        'callback','roborun2("run_file");');

        %  SECOND COLUMN OF CONTROLS

        uicontrol('style','text','position',[buttx+120 butty-20 110 20],...
                'string','Cycles','backgroundcolor',tan,...
                'foregroundcolor','black');
        uicontrol('style','popup','tag','num_cycles','backgroundcolor',...
                aquamarine,'position',[buttx+120 butty-40 120 400],'string',...
                '500|1000|1500|2000|2500|3000|3500|4000|4500|5000|5500|6000|6500|7000|7500|8000|8500|9000|
9500|10000');

        uicontrol('style','text','position',[buttx+120 butty-60 120 20],...
                'string','Runs','backgroundcolor',tan,...
                'foregroundcolor','black');
        uicontrol('style','popup','tag','num_runs','backgroundcolor',...
                aquamarine,'position',[buttx+120 butty-80 120 400],'string',...
                '1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20');
```

```
        uicontrol('style','text','position',[buttx+120 butty-100 120 20],...
                'string','Save @ Run End','backgroundcolor',tan,...
                'foregroundcolor','black');
        uicontrol('style','popup','tag','pause_stat','backgroundcolor',...
                aquamarine,'position',[buttx+120 butty-120 120 400],'string',...
                'Yes|No');

        uicontrol('style','text','position',[buttx+120 butty-140 120 20],...
                'string','Port ID','backgroundcolor',tan,...
                'foregroundcolor','black');
        uicontrol('style','popup','tag','port_id_tag','backgroundcolor',...
                aquamarine,'position',[buttx+120 butty-160 120 400],'string',...
                '1|2|3|4');

        %  THIRD COLUMN OF SIMULATOR CONTROLS - PUT A COLOURED BOX AROUND THEM
        bkgclr=[.72 .68 .35];
        frame_w=130;
        frame_h=255;
        uicontrol('style','frame','position',[buttx+245 butty-220 frame_w frame_h],...
                        'backgroundcolor',bkgclr);

        uicontrol('style','text','position',[buttx+260 butty 90 30],...
                'string','Simulator Controls','backgroundcolor',bkgclr,...
                'foregroundcolor','blue');


        uicontrol('style','text','position',[buttx+250 butty-30 120 20],...
                'string','Playpen Control','backgroundcolor',bkgclr,...
                'foregroundcolor','black');

        uicontrol('style','pushbutton','position',[buttx+250 butty-60 120 30],...
                        'string','Open Default','callback','roborun2("loaddef");');

        uicontrol('style','pushbutton','position',[buttx+250 butty-90 120 30],...
                        'string','Open Custom','callback','roborun2("loadpen");');

        uicontrol('style','pushbutton','position',[buttx+250 butty-120 120 30],...
                        'string','Close Playpen','callback','roborun2("closepen");');

        uicontrol('style','text','position',[buttx+250 butty-150 120 20],...
                'string','Mouse Action','backgroundcolor',bkgclr,...
                'foregroundcolor','black');

        uicontrol('style','popup','tag','act_stat','backgroundcolor',...
                aquamarine,'position',[buttx+250 butty-170 120 40],'string',...
                'Place Khep|Move Khep|Rotate Left|Rotate Right|Delete Khep|Place Lamp|Remove
Lamp','enable','off');

        uicontrol('style','text','position',[buttx+250 butty-190 120 20],...
```

```
            'string','Lamp Control','backgroundcolor',bkgclr,...
            'foregroundcolor','black');

uicontrol('style','popup','tag','lamp_con','backgroundcolor',...
            aquamarine,'position',[buttx+250 butty-210 120 40],'string',...
            'All On|Random|Sequenced','enable','off');


% MENU ITEMS
%  Now define some menus at the top for other functionality
% Create top level menus.
menu_file_stuff = uimenu(findobj('tag','robo_runner'),'label','File');
menu_utils_stuff = uimenu(findobj('tag','robo_runner'),'label','Utilities');
menu_ud_stuff= uimenu(findobj('tag','robo_runner'),'label','UserDefined');



% Create menu items.
% Submenu of "File"
menu_exit = uimenu(menu_file_stuff,'label','Exit',...
                        'callback','roborun2("quit_robo");');

% Submenu of "Utilities"
menu_real=uimenu(menu_utils_stuff,'label','Real Khepera');
menu_sim=uimenu(menu_utils_stuff,'label','Khepera Simulation',...
                        'separator','on');

% Submenu of "Real Khepera"
menu_sensetest = uimenu(menu_real,'label','Sensor Testing',...
                        'callback','tstsense;');

% Submenu of "Khepera Simulation"
menu_make_pen = uimenu(menu_sim,'label','Create New Playpen',...
                        'callback','play_pen;');
menu_simtest = uimenu(menu_sim,'label','Sensor Testing',...
                        'callback','simsense;');

% Submenu of "User Defined"
menu_ud1= uimenu(menu_ud_stuff,'label','USER 1',...
                        'callback','roborun2("ud1");');

menu_ud2= uimenu(menu_ud_stuff,'label','USER 2',...
                        'callback','roborun2("ud2");');

menu_ud3= uimenu(menu_ud_stuff,'label','USER 3',...
                        'callback','roborun2("ud3");');

menu_ud4= uimenu(menu_ud_stuff,'label','USER 4',...
                        'callback','roborun2("ud4");');
```

```
        menu_ud5= uimenu(menu_ud_stuff,'label','USER 5',...
                        'callback','roborun2("ud5");');

        menu_ud6= uimenu(menu_ud_stuff,'label','USER 6',...
                        'callback','roborun2("ud6");');

        % Place the appropriate copyright data
        uicontrol('style','text','position',[buttx+120 10 200 20],...
                'string','Copyright (c) Rich Goyette',...
                'backgroundcolor',tan,...
                'foregroundcolor','blue');

        return;

elseif strcmp(command_str,'get_file')
        [ctrl_file, ctrl_path]=uigetfile('*.m',...
                                        'Control Algorithm Filename',...
                                        200,200);

elseif strcmp(command_str,'run_file')
        %  Check and see if there is a simulation window open.  If so, and there
        %  are lamps in it, ensure the lamp configuration reflects the current
        %  choice in this window (regardless of running simulated file or real)
        tmp=findobj(h_sim_window);
        if tmp ~= []
                %  Window there.  Are there lamps in it?
                lamp_handles=findobj('tag','lamp');
                if lamp_handles ~= []
                        %  Set a color for lamps off
                        lamp_off_color=[.51 .02 .25];
                        %  Get the menu setting for the lamps
                        lamp_arrange=get(findobj('tag','lamp_con'),'value');
                        %  Set up a counter to track how many lamps are on
                        onoff=0;
                        for i=1:length(lamp_handles)
                                lamp_dat=get(lamp_handles(i),'userdata');
                                %  Find out how many are on.
                                onoff=onoff+lamp_dat(3);
                        end
                        %  If random is chosen, then shake up lamps at every run start
                        if (lamp_arrange==2)
                                light_a_lamp=round(rand*(length(lamp_handles)-1))+1;
                                for i=1:length(lamp_handles)
                                        if i~=light_a_lamp
                                                tmp2=get(lamp_handles(i),'userdata');
                                                tmp2(3)=0;
                                                set(lamp_handles(i),'userdata',tmp2);
                                                set(lamp_handles(i),'facecolor',lamp_off_color,...
                                                        'edgecolor',lamp_off_color);
                                        else
```

```
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=1;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor','yellow');
                                end
                        end
                end
                %  If all lights are on or off and sequence is chosen, then find
                %  and turn on the first (or nearest to first) lamp placed
                if ((onoff>1)|(onoff==0))&(lamp_arrange==3)
                        light_a_lamp=length(lamp_handles);
                        for i=1:length(lamp_handles)
                                if i~=light_a_lamp
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=0;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor',lamp_off_color,...
                                                'edgecolor',lamp_off_color);
                                else
                                        tmp2=get(lamp_handles(i),'userdata');
                                        tmp2(3)=1;
                                        set(lamp_handles(i),'userdata',tmp2);
                                        set(lamp_handles(i),'facecolor','yellow');
                                end
                        end

                end
                %  If only one is on and lamp control is selected to all on
                %  then reset the lamps to all on.
                if ((onoff==1)|(onoff==0))&(lamp_arrange==1)
                        for i=1:length(lamp_handles)
                                tmp2=get(lamp_handles(i),'userdata');
                                tmp2(3)=1;
                                set(lamp_handles(i),'userdata',tmp2);
                                set(lamp_handles(i),'facecolor','yellow');
                        end
                end

        end

end

%  Proceed with running the file
if ctrl_file==[]|ctrl_file==0
        errordlg('Must have a controller file name first!');
        return;
else
        %  Cut off the .m
        ctrl_name=ctrl_file(1:(length(ctrl_file)-2));
        %  Get the setting data to pass to the controller function
```

```
                % Get port id, number cycles, number runs, pause status.
                port_id=get(findobj('tag','port_id_tag'),'value');
                numcycles=get(findobj('tag','num_cycles'),'value')*500;
                numruns=get(findobj('tag','num_runs'),'value');
                pstat=get(findobj('tag','pause_stat'),'value');
                % Check to see if there is an object on the screen
                % Later, add this check to a flag saying "update at end"...
                playpenthere=findobj('tag','khep_simwindow');
                khepthere=findobj('tag','khep');
                if (playpenthere == [])|((playpenthere ~= [])&(khepthere ~= []))
                        % No playpen, assume using a real khepera, or a playpen
                        % and a simulated khep exist
                        %%%
                        % Modify the following line to add variables to be
                        % passed back from the looping function.  List your variables
                        % in square brackets, separated by commas in followed
                        % by an equals sign.  For example:
                        % [var1,var2]=feval(ctrl_name,port_id,numcycles,numruns,pstat);
                        %%
                        feval(ctrl_name,port_id,numcycles,numruns,pstat);

                        % The following statement is an example for SRN1, a neural simulation
                        % I did.  There were several variables I wanted to bring back here.
                        %[xdata_stat,ydata_stat,W1,b1,W2,b2,a1_previous]=feval(ctrl_name,port_id,numcycles,
numruns,pstat);
                else
                        warndlg('There must be a khep in the playpen!');
                        return;
                end
        end

elseif strcmp(command_str,'init_file')
        if ctrl_file==[]|ctrl_file==0
                errordlg('Must have a controller file name first!');
                return;
        else
                % Cut off the .m
                ctrl_name=ctrl_file(1:(length(ctrl_file)-2));
                feval(ctrl_name);
        end

elseif strcmp(command_str,'quit_robo')
        delete(h_mainfig);
        if findobj('tag','khep_simwindow')
                delete(findobj('tag','khep_simwindow'));
        end
        return;

elseif strcmp(command_str,'loadpen')
        [save_file save_path]=uigetfile('*.mat','Load Custom Playpen...');
```

```
        if save_file~=0
                filepath=[save_path save_file];
                load(filepath);
                tmp=findobj(h_sim_window);
                if tmp == []
                        % Draw the figure window for the playpen.
                        figure('position',[412,260,600,500],...
                        'menubar','none','name','Khepera Simulator Window',...
                        'numbertitle','off','resize','off',...
                                'tag','khep_simwindow','windowbuttondownfcn',...
                                'modpen2("screen_action")');
                        axes('units','pixels','position',[0,0,600,500],'visible','off',...
                                'xlim',[0 600],'ylim',[0 500],'aspectratio',[1,NaN],...
                                'tag','simwindow_axes');
                else
                        figure(tmp(1));

                end
                img_handle=image(btmap1);
                %colormap(clrmap1)
                colormap('bone')
                % Image sets the axis to (ij).  Set it back to (xy)
                axis('xy');
                % Enable simulator controls
                roborun2('en_sim');
        else
                warndlg('Custom Playpen not loaded!');
        end

elseif strcmp(command_str,'loaddef')
        % Draw the figure window for the playpen.
        tmp=findobj(h_sim_window);
        if tmp==[]
                figure('position',[412,260,600,500],...
                'menubar','none','name','Khepera Simulator Window',...
                'numbertitle','off','resize','off',...
                        'tag','khep_simwindow','windowbuttondownfcn',...
                        'modpen2("screen_action")');
                axes('units','pixels','position',[0,0,600,500],'visible','off',...
                        'xlim',[0 600],'ylim',[0 500],'aspectratio',[1,NaN],...
                        'tag','simwindow_axes')
        else
                figure(tmp(1));
        end
        % Draw the default bitmap in the playpen region.  One of the user menu options
        % will allow getting a new or custom frame.
        load defmap;
        img_handle=image(btmap1);
        colormap(clrmap1);
        % Image sets the axis to (ij).  Set it back to (xy)
```

```
        axis('xy');
        % Enable simulator controls now that there's a map
        roborun2('en_sim');

elseif strcmp(command_str,'closepen')
        tmp=findobj(h_sim_window);
        if tmp ~= []
                delete(tmp(1));
                roborun2('en_sim');
        end

elseif strcmp(command_str,'en_sim')
        tmp=findobj('tag','khep_simwindow');
        if tmp==[]
                set(findobj('tag','act_stat'),'enable','off');
                set(findobj('tag','lamp_con'),'enable','off');
        else
                set(findobj('tag','act_stat'),'enable','on');
                set(findobj('tag','lamp_con'),'enable','on');
                % set enable properties to "on" for the other
                % entries in the simulator box
        end

%% Userdefined Actions

elseif strcmp(command_str,'ud1')
        if exist('user1')
                user1;
        end
elseif strcmp(command_str,'ud2')
        if exist('user2')
                user2;
        end
elseif strcmp(command_str,'ud3')
        if exist('user3')
                user3;
        end
elseif strcmp(command_str,'ud4')
        if exist('user4')
                user4;
        end
elseif strcmp(command_str,'ud5')
        if exist('user5')
                user5;
        end
elseif strcmp(command_str,'ud6')
        if exist('user6')
                user6;
        end
end
```

Annex B: Matlab source code for Khepera Toolbox