```
function [sys,x0,str,ts,simStateCompliance] = ballbeam_dynamics(t,x,u,flag,P)
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P);

  %%%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,x,u,P);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,x,u);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys=mdlTerminate(t,x,u);

  %%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

% end sfuntmpl

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P)

sizes = simsizes;

sizes.NumContStates  = 4;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
```

```
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0  = [P.theta0; P.z0; P.thetadot0; P.zdot0];

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts  = [0 0];

% Specify the block simStateCompliance. The allowed values are:
%    'UnknownSimState', < The default setting; warn and assume DefaultSimState
%    'DefaultSimState', < Same sim state as a built-in block
%    'HasNoSimState',   < No sim state
%    'DisallowSimState' < Error out when saving or restoring the model sim state
simStateCompliance = 'UnknownSimState';

% end mdlInitializeSizes

%
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
%
function sys=mdlDerivatives(t,x,u,P)
  theta    = x(1);
  z        = x(2);
  thetadot = x(3);
  zdot     = x(4);
  F        = u(1);

  thetaddot = (1/((P.m2*P.L^2)/3+P.m1*z^2))*...
      (-2*P.m1*z*zdot*thetadot-P.m1*P.g*z*cos(theta)...
      -P.m2*P.g*P.L/2*cos(theta)+P.L*F*cos(theta));
%  thetaddot = (1/((m2*L^2)/3+m1*z^2))*...
%       ( -m2*g*L/2*cos(theta)+L*F*cos(theta))
  zddot     = z*thetadot^2-P.g*sin(theta);

  f= [thetadot; zdot; thetaddot; zddot];
sys = f;

% end mdlDerivatives

%
%=============================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
%
function sys=mdlUpdate(t,x,u)
```

```
sys = [];

% end mdlUpdate

%
%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
%
function sys=mdlOutputs(t,x,u)
  theta    = x(1);
  z        = x(2);

sys = [z; theta];

% end mdlOutputs

%
%=============================================================================
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.  Note that the result is
% absolute time.  Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=============================================================================
%
function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit

%
%=============================================================================
% mdlTerminate
% Perform any end of simulation tasks.
%=============================================================================
%
function sys=mdlTerminate(t,x,u)

sys = [];

% end mdlTerminate
```