```matlab
function [sys,x0,str,ts,simStateCompliance]...
        = VTOL_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)

%
% call simsizes for a sizes structure, fill it in and convert it
% to a sizes array.
%
```

```
% Note that in this example, the values are hard coded.  This is
% not a recommended practice as the characteristics of the block
% are typically defined by the S-function parameters.
%
sizes = simsizes;

sizes.NumContStates  = 6;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0  = [P.z0; P.h0; P.theta0; P.zdot0; P.hdot0; P.thetadot0];

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

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

simStateCompliance = 'UnknownSimState';

% end mdlInitializeSizes

%
%===================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%===================================================================
%
function sys=mdlDerivatives(t,x,u,P)
  theta    = x(3);
  zdot     = x(4);
  hdot     = x(5);
  thetadot = x(6);
  fr       = u(1);
  fl       = u(2);

  zddot     = -(fr+fl)*sin(theta)/(P.mc+2*P.mr);
  hddot     = (-(P.mc+2*P.mr)*P.g + (fr+fl)*cos(theta))/(P.mc+2*P.mr);
  thetaddot = P.d*(fr-fl)/(P.Jc+2*P.mr*P.d^2);

sys = [zdot; hdot; thetadot; zddot; hddot; thetaddot];

% 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)
  z        = x(1);
  h        = x(2);
  theta    = x(3);

sys = [z; h; theta];

% end mdlOutputs

%
%====================================================================
% mdlGetTimeOfNextVarHit
%====================================================================
%
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
```