Macro: slider_crank_dsp.mac

Macro	slider_crank_dsp.mac
Description	Demonstrates a slider-crank mechanism (four bar mechanism) and uses surface (dsp) models for the links, etc.
CM version	Any; some commands (commented out here) require the DAG version of the modeller
Requires	dsp files used in `setup` function, namely: sc_coupler.dsp, sc_crank.dsp, sc_pin.dsp, sc_pivot.dsp, sc_rail.dsp, sc_slider.dsp
See also	macro: slider_crank.mac

What the macro does

This macro creates a model of a slider-crank mechanism (a type of four bar mechanism) using surface (dsp) models to show the links. These were originally created using Solid Edge and then saved as STL files and then converted into DSP format. The model is shown in figure 1.

This macro is an extension of macro slider_crank.mac.

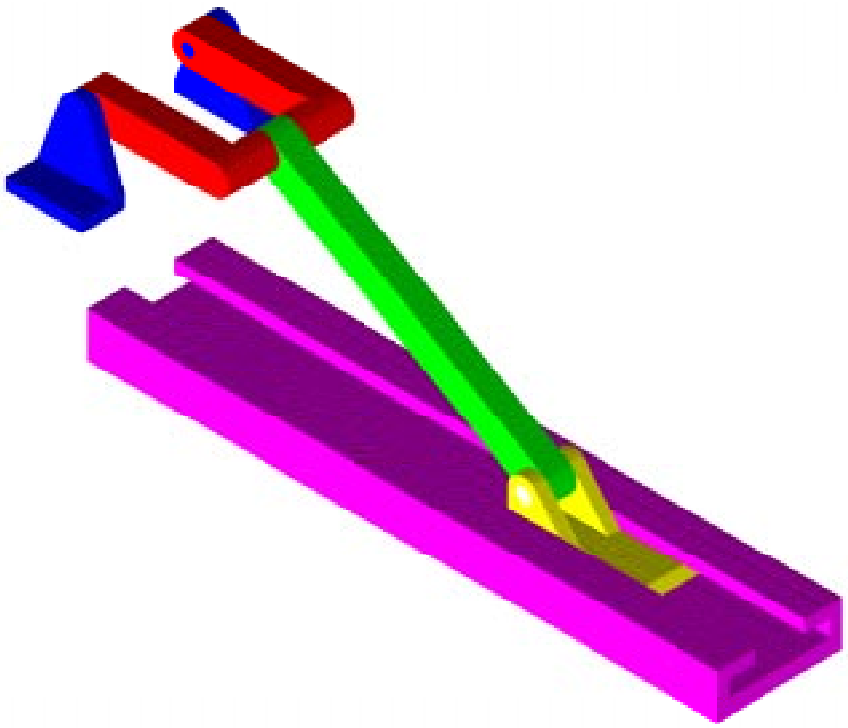

Figure 1: Slider-crank (four bar) mechanism

How the main part of the macro works

The macro is an extension of macro slider_crank.mac.

The extension is to use DSP files to define surface models of the links. Each DSP object is associated with an entity of type `geom`. Each entity is embedded into a three dimensional model space. For the moving links, this model space is itself embedded into the corresponding two dimensional space used to hold the stick geometry.

GM
May 2013

Listing of macro

```
0001   $ ========================================================================
0002   $   slider_crank_dsp.mac
0003   $ ========================================================================
0004   $   Slider crank mechanism
0005   $   based on slider_crank.mac
0006   $   revised: May 2013
0007   $ ========================================================================
0008
0009   dec int   npoint;                                    $ number of points
0010   npoint = 36;                                         $ make it 36
0011   dec int   delay_factor;                              $ delay factor
0012   delay_factor = 0;                                    $ set to zero
0013
0014   dec string file_name;                                $ name of output file
0015   dec geom   p0;                                       $ fixed pivot point
0016   dec geom   lcrank, lcoupler, lslider;                $ lines for links/rail
0017   dec geom   pcoupler;                                 $ end of coupler point
0018   dec mod2   mcrank, mcoupler, mslider;                $ model spaces
0019   dec real   len_crank, len_coupler, len_slider;       $ link lengths
0020   dec real   yoffset;                                  $ offset value
0021   dec mod3   m1crank, m1coupler, m1slider;             $ 3d spaces to ...
0022   dec mod3   m1rail, m1pin;                            $ ... hold the dsp ...
0023   dec mod3   m1pivot1, m1pivot2;                       $ ... objects
0024   dec geom   dcrank, dcoupler, dslider;                $ dsp objects
0025   dec geom   drail, dpin;                              $ ditto
0026   dec geom   dpivot1, dpivot2;                         $ ditto
0027   dec geom   qq[npoint];                               $ array of points
0028   dec real   xx[npoint];                               $ array of pos
0029   dec real   vv[npoint];                               $ array of vel
0030   dec real   aa[npoint];                               $ array of acc
0031   dec real   tstep;                                    $ time step
0032   dec int    echo_qq_flag;                             $ integer flag
0033   dec real   v_theta, v_phi, v_psi, v_inc;             $ viewing angles
0034
0035   file_name = "slider_crank.out";                      $ set the file name
0036   len_crank = 40;                                      $ length of crank
0037   len_coupler = 100;                                   $ length of coupler
0038   len_slider = 160;                                    $ length of slide rail
0039   yoffset = -50;                                       $ offset of rail
0040   tstep = 0.1;                                         $ time step
0041   echo_qq_flag = 1;                                    $ flag set to unity
0042   v_inc = 1.0;                                         $ viewing increment
0043
0044
0045   function setup                                       $ start of function
0046   {
0047      p0 = pnt(0,0,0);                                  $ define point p0
0048      cfont(7,p0);                                      $ change its font
0049      ccol(blue(),p0);                                  $ and colour
0050      mcrank = mod2(0,0,0);                             $ crank model space
```

Figure 2: Listing of macro slider_crank_dsp.mac (part 1)

```
0051      lcrank = lin( 0,0,0, len_crank,0,0, mcrank );    $ crank line
0052      mcoupler = mod2(0,0,-45,mcrank);                 $ coupler model space
0053      lcoupler = lin(0,0,0,len_coupler,0,0,mcoupler);  $ coupler line
0054      pcoupler = pnt( len_coupler, 0, 0, mcoupler );   $ end of coupler point
0055      mslider = mod2(0,0,0);                           $ slider model space
0056      lslider = lin(0,yoffset,0,len_slider,yoffset,0); $ line for slide rail
0057      ccol( red(), lcrank );                           $ make crank red
0058      ccol( green(), lcoupler, pcoupler );             $ and coupler green
0059      ccol( magenta(), lslider );                      $ and rail magenta
0060      cfont( 4, pcoupler );                            $ make font a circle
0061      pivot( mcrank, lcrank:e1, p0 );                  $ join crank to p0
0062      pivot( mcoupler, lcoupler:e1, lcrank:e2 );       $ and crank to coupler
0063
0064      m1crank = mod3( 0,0,0, 0,0,0, mcrank );          $ model space and ...
0065      dcrank = readdsp( "sc_crank.dsp" );              $ ... dsp for crank
0066      dcrank:m = &m1crank;
0067      ccol( red(), dcrank );
0068
0069      m1coupler = mod3( 0,0,0, 0,0,0, mcoupler );      $ model space and ...
0070      dcoupler = readdsp( "sc_coupler.dsp" );          $ ... dsp for coupler
0071      dcoupler:m = &m1coupler;
0072      ccol( green(), dcoupler );
0073
0074      m1slider = mod3( 0,0,0, 0,0,0, mslider );        $ model space and ...
0075      dslider = readdsp( "sc_slider.dsp" );            $ ... dsp for slider
0076      dslider:m = &m1slider;
0077      ccol( yellow(), dslider );
0078
0079      m1rail = mod3( 0,yoffset-20,0, 0,0,0 );          $ model space and ...
0080      drail = readdsp( "sc_rail.dsp" );                $ ... dsp for rail
0081      drail:m = &m1rail;
0082      ccol( magenta(), drail );
0083
0084      m1pin = mod3( 0,0,0, 0,0,0, m1slider );          $ model space and ...
0085      dpin = readdsp( "sc_pin.dsp" );                  $ ... dsp for pin
0086      dpin:m = &m1pin;
0087      ccol( blue(), dpin );
0088
0089      m1pivot1 = mod3( 0,0,-20, 0,0,0 );               $ model space and ...
0090      dpivot1 = readdsp( "sc_pivot.dsp" );             $ ... dsp for povot 1
0091      dpivot1:m = &m1pivot1;
0092      ccol( blue(), dpivot1 );
0093
0094      m1pivot2 = mod3( 0,0,20, 0,180,0 );              $ model space and ...
0095      dpivot2 = readdsp( "sc_pivot.dsp" );             $ ... dsp for povot 2
0096      dpivot2:m = &m1pivot2;
0097      ccol( blue(), dpivot2 );
0098  }                                                    $ end of function
0099
0100
```

Figure 3: Listing of macro slider_crank_dsp.mac (part 2)

```
0101   function   echo_wire_frame                          $ start of function
0102   {
0103      dec  int  code;
0104      inp       code;                                  $ one input argument
0105
0106      echo( code, p0, pcoupler );                      $ echo on/off
0107      echo( code, lcrank, lcoupler, lslider );
0108
0109      rpnt( 1 );
0110   }                                                   $ end of function
0111
0112
0113   function   echo_dsp                                 $ start of function
0114   {
0115      dec  int  code;
0116      inp       code;                                  $ one input argument
0117
0118      echo( code, dcrank, dcoupler, dslider );         $ echo on/off
0119      echo( code, drail, dpin );
0120      echo( code, dpivot1, dpivot2 );
0121
0122      rpnt( 1 );
0123   }                                                   $ end of function
0124
0125
0126   function   echo_qq                                  $ start of function
0127   {
0128      dec  int  code, i;
0129      inp       code;                                  $ one input argument
0130
0131      loop( i, 0, npoint )                             $ start loop
0132      { echo( code, qq[i] );                           $ echo on/off
0133      }                                                $ end loop
0134
0135      rpnt( 1 );                                       $ clear and repaint
0136      echo_qq_flag = code;                             $ update flag value
0137   }                                                   $ end of function
0138
0139
0140
0141   function   do_delay                                 $ start of function
0142   {
0143      dec  int    i, j;                                $ local variables
0144      dec  real   a, b;
0145
0146      loop( i, 0, delay_factor )                       $ start loop
0147      { a = i;
0148        loop( j, 0, 1000 )                             $ start another loop
0149        { b = j;                                       $ do pointless calc ...
0150          a = a + sin( a + b );                        $ ... to waste time
```

Figure 4: Listing of macro slider_crank_dsp.mac (part 3)

```
0151        }                                          $ end inner loop
0152      }                                            $ end outer loop
0153    }                                              $ end of function
0154
0155
0156    function assemble_inner                        $ start of function
0157    {
0158       var mcoupler;                               $ coupler angle varies
0159
0160       rule( pcoupler on lslider );                $ put coupler on rail
0161    }                                              $ end of function
0162
0163
0164    function   assemble                            $ start of function
0165    {
0166       assemble_inner();                           $ do basic assembly
0167       mslider:p = transf( pcoupler ):x;           $ adjust slider pos ...
0168       mslider:q = transf( pcoupler ):y;           $ ... in x and y
0169    }                                              $ end of function
0170
0171
0172
0173    function cycle                                 $ start of function
0174    {
0175       dec int i, code;                            $ local variables
0176       inp code;                                   $ one argument
0177
0178       loop( i, 0, npoint )                        $ loop for cycle
0179       { mcrank:a = i*360/npoint;                  $ increment crank angle
0180         assemble();                               $ call assemble
0181         rpnt(code);                               $ repaint graphics
0182         qq[i] = transf( pcoupler );               $ get end of coupler
0183         ccol( cyan(), qq[i] );                    $ change colour
0184         cfont( 6, qq[i] );                        $ and its font
0185         echo( echo_qq_flag, qq[i] );              $ echo on/off
0186         xx[i] = qq[i]:x;                          $ get x coordinate
0187         if( i < npoint )
0188         { do_delay();
0189         }
0190       }
0191    }                                              $ end of function
0192
0193
0194    function find_vel_acc                          $ start of function
0195    {
0196       vv = deriv( 1, xx, tstep );                 $ first derivative
0197       aa = deriv( 2, xx, tstep );                 $ second derivative
0198    }                                              $ end of function
0199
0200
```

Figure 5: Listing of macro slider_crank_dsp.mac (part 4)

```
0201   function do_output                                    $ start of function
0202   {
0203      dec int  i;                                        $ declare local int
0204
0205      fwriteln( 0, "Opening file:", file_name );         $ message to screen
0206      fopen( 2, 2, file_name );                          $ open file to write
0207      fwriteln( 2, asc(36), "File:", file_name );        $ output file name
0208      fwriteln( 2, asc(36), "Date:", date() );           $ output date/time
0209      fwriteln( 2 );                                     $ blank line
0210
0211      loop( i, 0, npoint )
0212      { fwrite( 2, "%3d", i );                           $ output counter
0213        fwrite( 2, "%12.5lf", xx[i] );                   $ output pos
0214        fwrite( 2, "%12.5lf", vv[i] );                   $ output vel
0215        fwrite( 2, "%12.5lf", aa[i] );                   $ output acc
0216        fwriteln( 2 );                                   $ end output line
0217      }
0218
0219      fwriteln( 2 );                                     $ blank line
0220      fwriteln( 2, asc(36), "End of file" );             $ output end of file
0221      fwriteln( 2 );                                     $ blank line
0222      fclose( 2 );                                       $ close file
0223      fwriteln( 0, "File closed:", file_name );          $ write to screen
0224   }                                                     $ end of function
0225
0226   graphics();                                           $ graphics window
0227   deflight();                                           $ default lighting
0228   lposition( 0, 1,0,4 );                                $ adjust light position
0229   vangles( 45.0, 35.5, -120.0 );                        $ set view
0230   setup();                                              $ call setup
0231   assemble();                                           $ call assemble
0232   echo_wire_frame( 0 );                                 $ wires off
0233   echo_dsp( 1 );                                        $ surfaces on
0234   echo_qq( 0 );                                         $ track points off
0235   rpnt();                                               $ repaint screen
0236   zoom();                                               $ and zoom all
0237   zoom(0.8);                                            $ zoom down a little
0238
0239   menu slider                                           $ create menu
0240   {
0241      button Setup
0242      { setup();                                         $ call setup function
0243      }
0244      submenu Background>
0245      { button Black
0246        { background( 0, 0, 0 );
0247          rpnt();
0248        }
0249        button Grey (0.33)
0250        { background( 0.33, 0.33, 0.33 );
```

Figure 6: Listing of macro slider_crank_dsp.mac (part 5)

```
0251          rpnt();
0252        }
0253        button Grey (0.67)
0254        { background( 0.67, 0.67, 0.67 );
0255          rpnt();
0256        }
0257        button White
0258        { background( 1, 1, 1 );
0259          rpnt();
0260        }
0261      }
0262      submenu Views>
0263      { button View 1
0264        { vangles( 45.0, 35.5, -120.0 );
0265          rpnt(1);
0266        }
0267        button View 2
0268        { vangles( 34.7, 17.7, -101.8 );
0269          rpnt(1);
0270        }
0271        button Front
0272        { vvector( 0, 0, 1 );
0273          rpnt(1);
0274        }
0275
0276    $ ========================================================================
0277    $   function getvangles is only available in DAG version of modeller
0278    $ ========================================================================
0279    $     button Rotate+
0280    $     { v_theta, v_phi, v_psi = getvangles();
0281    $       vangles( v_theta, v_phi, v_psi+v_inc );
0282    $       rpnt(1);
0283    $     }
0284    $     button Rotate-
0285    $     { v_theta, v_phi, v_psi = getvangles();
0286    $       vangles( v_theta, v_phi, v_psi-v_inc );
0287    $       rpnt(1);
0288    $     }
0289    $     button Double v_inc
0290    $     { v_inc = 2.0*v_inc;
0291    $       fwriteln( 0, "v_inc =", v_inc );
0292    $     }
0293    $     button Halve v_inc
0294    $     { v_inc = 0.5*v_inc;
0295    $       fwriteln( 0, "v_inc =", v_inc );
0296    $     }
0297    $ ========================================================================
0298
0299      }
0300
```

Figure 7: Listing of macro slider_crank_dsp.mac (part 6)

```
0301      button Cycle
0302      { cycle(1);                                        $ call cycle function
0303      }
0304      submenu Delay>
0305      { button 0
0306        { delay_factor = 0;                              $ no delay
0307        }
0308        button 1
0309        { delay_factor = 1;                              $ set delay_factor
0310        }
0311        button 2
0312        { delay_factor = 2;                              $ set delay_factor
0313        }
0314        button 5
0315        { delay_factor = 5;                              $ set delay_factor
0316        }
0317        button 10
0318        { delay_factor = 10;                             $ set delay_factor
0319        }
0320        button 20
0321        { delay_factor = 20;                             $ set delay_factor
0322        }
0323        button 50
0324        { delay_factor = 50;                             $ set delay_factor
0325        }
0326        button 100
0327        { delay_factor = 100;                            $ set delay_factor
0328        }
0329      }
0330      submenu Echo>
0331      { button Wireframe ON
0332        { echo_wire_frame( 1 );
0333        }
0334        button Wireframe OFF
0335        { echo_wire_frame( 0 );
0336        }
0337        button Surface (dsp) ON
0338        { echo_dsp( 1 );
0339        }
0340        button Surface (dsp) OFF
0341        { echo_dsp( 0 );
0342        }
0343        button Rail (dsp) OFF
0344        { echo( 0, drail );
0345          rpnt( 1 );
0346        }
0347        button Track points (qq) ON
0348        { echo_qq( 1 );
0349        }
0350        button Track points (qq) OFF
```

Figure 8: Listing of macro slider_crank_dsp.mac (part 7)

```
0351        { echo_qq( 0 );
0352        }
0353      }
0354      button Vel/acc
0355      { find_vel_acc();                                  $ find vel and acc
0356        fwriteln( 0, "Completed" );                      $ write to screen
0357      }
0358      button Output
0359      { do_output();                                     $ output values
0360      }
0361    }
0362
0363    remmenu();                                           $ remove previous menu
0364    addmenu( slider );                                   $ put up new menu
0365
0366    $ End of file
0367
```

Figure 9: Listing of macro slider_crank_dsp.mac (part 8)