Macro: crank_rocker_dsp.mac

Macro	crank_rocker_.mac
Description	Demonstrates four bar chain mechanism (as a crank-rocker) and uses surfaces models (dsp files) for the various components
CM version	Any
Requires	the dsp files read by the `setup` function, that is: fbc_coupler.dsp, fbc_crank.dsp, fbc_driven.dsp, fbc_pin.dsp, fbc_pivot.dsp, fbc_probe.dsp
See also	macro: crank_rocker.mac

What the macro does

This macro is an extension of macro `crank_rocker.mac`. Both macro set up a four bar mechanism and then allow its motion to be simulated. The simpler macro represents the links by straight lines. Each line lies in a model space. The extended version takes a facetted model of each link and embeds this also into the model space. Figure 1 shows the result.

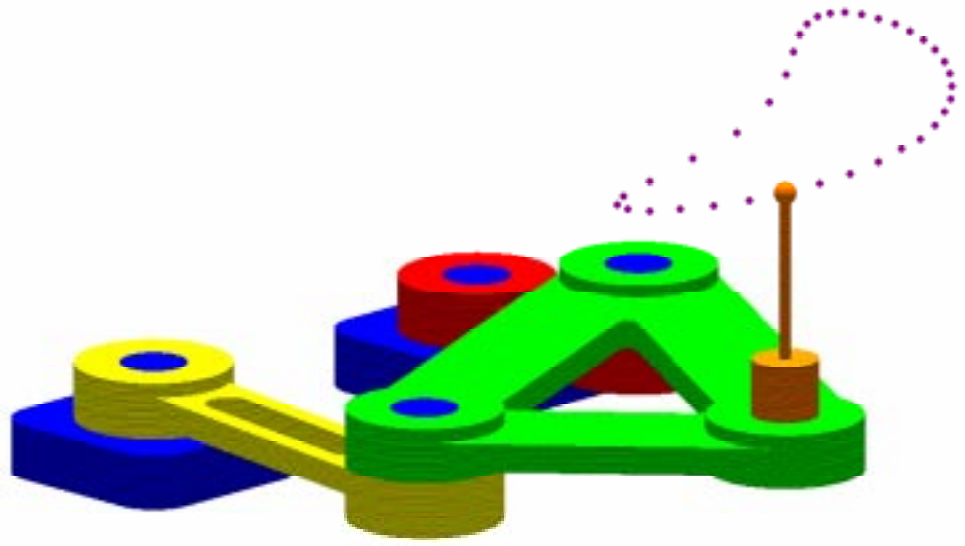

Figure 1: Crank-rocker (four bar) mechanism

How the main part of the macro works

The listing of the macro is given below. The lines of the macro are numbered for ease of reference.

The basic logic of the macro is the same as for `crank_rocker.mac`. Only some of the additional ideas are dealt with here. Note that the size of the mechanism is ten times that in the simpler macro.

The facetted models of the links are created as follows. Each link is modelled in Solid Edge and then saved as an STL file (as well as a PAR file). A separated conversion program is used to convert the STL files into the DSP format required by the constraint modeller.

The entities for the DSP models are declared in lines 0026–0028. Additionally three dimensional model spaces to hold these are declared in lines 0021–0025. Each space is embedded into one of the two dimensional spaces. The three dimensional spaces are used only to adjust the position of the facetted models in the z-direction. The DSP files are read in lines 0078–0104. After each files is read, the geometric object is embedded into the appropriate model space (e.g. line 0079) and its colour is assigned.

GM
May 2013

Listing

```
0001   $ ========================================================================
0002   $   crank_rocker_dsp.mac
0003   $ ========================================================================
0004   $   simple four bar mechanism (crank-rocker)
0005   $   with surface models of links etc.
0006   $   GM - May 2013
0007   $ ========================================================================
0008   $   cf. crank_rocker.mac
0009   $ ========================================================================
0010
0011   dec int   npoint;                          $ number of points in cycle
0012   dec int   delay_factor;                    $ factor for delay in cycle
0013   npoint = 36;                               $ 36 points
0014   delay_factor = 0;                          $ no delay
0015
0016   dec real  d0, d1, d2, d3, d2x, d2y;        $ declare various lengths
0017   dec geom  p1, p2, l1, l2, l3, l2a, l2b;    $ declare various geometry
0018   dec geom  ptip;                            $ point for tip of coupler
0019   dec geom  qq[npoint];                      $ array of geom (points)
0020   dec mod3  m1, m2, m3;                      $ declare 3D model spaces
0021   dec mod3  mcrank, mcoupler, mdriven;       $ model spaces for links
0022   dec mod3  mpivot1, mpivot2;                $ model spaces for pivots
0023   dec mod3  mqq;                             $ model space for track
0024   dec mod3  mpin1, mpin3;                    $ model spaces for pins
0025   dec mod3  mprobe;                          $ model space for probe
0026   dec geom  dcrank, dcoupler, ddriven;       $ dsp for links
0027   dec geom  dpivot1, dpivot2;                $ dsp for pivots
0028   dec geom  dpin1, dpin3, dprobe;            $ dsp for pins and probe
0029   dec geom  echo_qq_flag;                    $ flag for track
0030
0031   d0  =  80;                                 $ distance between pivots
0032   d1  =  40;                                 $ crank length
0033   d2  = 100;                                 $ coupler length
0034   d2x =  80;                                 $ coupler x offset
0035   d2y =  60;                                 $ coupler y offset
0036   d3  =  80;                                 $ driven link length
0037
0038   echo_qq_flag = 1;                          $ set flag to one
0039
0040
0041   function setup                             $ start function
0042   {
0043      p1 = pnt( 0, 0, 0 );                    $ one pivot (at origin)
0044      p2 = pnt( d0, 0, 0 );                   $ second pivot
0045      ccol( blue(), p1, p2 );                 $ change colour
0046      cfont( 7, p1, p2 );                     $ change font of points
0047      m1 = mod3( 0, 0, 0, 0, 0, 90 );         $ model space for crank line
0048      m2 = mod3( 0, 0, 0, 0, 0, -45, m1 );    $ model space for coupler lineS
0049      m3 = mod3( 0, 0, 0, 0, 0, 90 );         $ model space for driven line
0050      l1 = lin( 0,0,0, d1,0,0, m1 );          $ line to represent crank
```

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

```
0051      l2 = lin( 0,0,0, d2,0,0, m2 );           $ line to represent coupler
0052      l2a = lin( 0,0,0, d2x,d2y,0, m2 );       $ line of coupler triangle
0053      l2b = lin( d2,0,0, d2x,d2y,0, m2 );      $ line of coupler triangle
0054      ptip = pnt( d2x, d2y, 0, m2 );           $ point at tip of triangle
0055      l3 = lin( 0,0,0, d3,0,0, m3 );           $ line to represent driven
0056      ccol( red(), l1 );                       $ make crank red
0057      ccol( green(), l2, l2a, l2b, ptip );     $ make coupler green
0058      ccol( yellow(), l3 );                    $ make driven link yellow
0059      cfont( 4, ptip );                        $ make it a circle
0060      pivot( m1, l1:e1, p1 );                  $ attach crank to p1
0061      pivot( m2, l2:e1, l1:e2 );               $ attach coupler to crank
0062      pivot( m3, l3:e1, p2 );                  $ attach driven to p2
0063      fix( m1:p, m1:q, m1:r, m1:ax, m1:ay );   $ fix other parts of space
0064      fix( m2:p, m2:q, m2:r, m2:ax, m2:ay );   $ fix other parts of space
0065      fix( m3:p, m3:q, m3:r, m3:ax, m3:ay );   $ fix other parts of space
0066
0067      mpivot1  = mod3( 0, 0, 0, 0,0,0 );       $ model space for pivot dsp
0068      mpivot2  = mod3(d0, 0, 0, 0,0,0 );       $ model space for pivot dsp
0069      mcrank   = mod3( 0, 0, 5, 0,0,0, m1 );   $ model space for crank dsp
0070      mcoupler = mod3( 0, 0,15, 0,0,0, m2 );   $ model space for coupler dsp
0071      mdriven  = mod3( 0, 0, 5, 0,0,0, m3 );   $ model space for driven dsp
0072      mqq      = mod3( 0, 0,60, 0,0,0 );       $ model space for track
0073      mpin1    = mod3( d1,0,10, 0,0,0, m1 );   $ model space for pin dsp
0074      mpin3    = mod3( d3,0,10, 0,0,0, m3 );   $ model space for pin dsp
0075
0076      mprobe = mod3( d2x,d2y,20, 0,0,0, m2 );  $ model space for pivot dsp
0077
0078      dpivot1 = readdsp( "fbc_pivot.dsp" );    $ read dsp data
0079      dpivot1:m = &mpivot1;                    $ embed into model space
0080      dpivot2 = readdsp( "fbc_pivot.dsp" );    $ read dsp data
0081      dpivot2:m = &mpivot2;                    $ embed into model space
0082      ccol( blue(), dpivot1, dpivot2 );        $ change colour
0083
0084      dcrank = readdsp( "fbc_crank.dsp" );     $ read dsp data
0085      dcrank:m = &mcrank;                      $ embed into model space
0086      ccol( red(), dcrank );                   $ change colour
0087
0088      dcoupler = readdsp( "fbc_coupler.dsp" ); $ read dsp data
0089      dcoupler:m = &mcoupler;                  $ embded into model space
0090      ccol( green(), dcoupler );               $ change colour
0091
0092      ddriven = readdsp( "fbc_driven.dsp" );   $ read dsp data
0093      ddriven:m = &mdriven;                    $ embed into model space
0094      ccol( yellow(), ddriven );               $ change colour
0095
0096      dpin1 = readdsp( "fbc_pin.dsp" );        $ read dsp data
0097      dpin1:m = &mpin1;                        $ embed into model space
0098      dpin3 = readdsp( "fbc_pin.dsp" );        $ read dsp data
0099      dpin3:m = &mpin3;                        $ embed into model space
0100      ccol( blue(), dpin1, dpin3 );            $ change colour
```

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

```
0101
0102      dprobe = readdsp( "fbc_probe.dsp" );     $ read dsp data
0103      dprobe:m = &mprobe;                      $ embed into model space
0104      ccol( rgb(1.0,0.5,0.0), dprobe );        $ change colour
0105   }                                           $ end of function
0106
0107
0108
0109   function   echo_wire_frame                  $ start of function
0110   {
0111      dec  int  code;                          $ local variable
0112      inp       code;                          $ function has one argument
0113
0114      echo( code, p1, p2, ptip );              $ echo points on/off
0115      echo( code, l1, l2, l2a, l2b, l3 );      $ echo lines on/off
0116      rpnt();                                  $ clear and repaint
0117   }                                           $ end of function
0118
0119
0120   function   echo_dsp                         $ start of function
0121   {
0122      dec  int  code;                          $ local variable
0123      inp       code;                          $ function has one argument
0124
0125      echo( code, dcrank, dcoupler, ddriven ); $ echo on/off some dsp variables
0126      echo( code, dpin1, dpin3, dprobe );      $ and some more
0127      echo( code, dpivot1, dpivot2 );          $ and some more
0128      rpnt();                                  $ clear and repaint
0129   }                                           $ end of function
0130
0131
0132   function   echo_qq                          $ start of function
0133   {
0134      dec  int  i, code;                       $ declare local variables
0135      inp          code;                       $ functio has one argument
0136
0137      loop( i, 0, npoint )                     $ start loop
0138      { echo( code, qq[i] );                   $ echo one track point on/off
0139      }                                        $ end loop
0140      rpnt();                                  $ clear and repaint
0141
0142      echo_qq_flag = code;                     $ set flag value
0143   }                                           $ end of function
0144
0145
0146
0147
0148   function   do_delay                         $ start of function
0149   {
0150      dec  int    i, j;                        $ declare local variables
```

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

```
0151      dec  real   a, b;
0152
0153      loop( i, 0, delay_factor )              $ start loop
0154      { a = i;
0155        loop( j, 0, 1000 )                    $ start another loop
0156        { b = j;                              $ do pointless calculation ...
0157          a = a + sin( a + b );               $ ... to waste some time
0158        }                                     $ end inner loop
0159      }                                       $ end outer loop
0160   }                                          $ end of function
0161
0162
0163   function assemble                          $ start of function
0164   {
0165      var  m2, m3;                           $ just change m2, m3 (angles az)
0166
0167      rule( l2:e2 on l3:e2 );                $ connect coupler and driven
0168   }                                          $ end of function
0169
0170
0171   function cycle                             $ start of function
0172   {
0173      dec int  i, code;                      $ declare local variables
0174      dec real ahold, astep;
0175      inp code;                              $ function has one argument
0176
0177      ahold = m1:az;                         $ hold current crank angle
0178      astep = 360/npoint;                    $ angular step
0179
0180      loop( i, 0, npoint )                   $ start of loop for cycling
0181      { m1:az = ahold + i*astep;             $ advance crank angle
0182        assemble();                          $ reassemble
0183        qq[i] = transf( ptip );              $ transform ptip to world space
0184        qq[i]:m = &mqq;                      $ embed in space mmqq
0185        ccol( magenta(), qq[i] );            $ change colour of held point
0186        cfont( 6, qq[i] );                   $ and its font
0187        echo( echo_qq_flag, qq[i] );         $ apply existing flag
0188        rpnt( code );                        $ repaint the graphics
0189        if( i < npoint )                     $ if not at the last step ...
0190        { do_delay();                        $ ... do a delay
0191        }
0192      }                                      $ end of loop
0193
0194      m1:az = ahold;                         $ restore original crank angle
0195
0196      fwriteln( 0, "Cycle finished" );       $ output messages
0197      fwriteln( 0, "delay_factor =",
0198                        delay_factor );
0199   }                                          $ end of fucntion
0200
```

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

```
0201
0202   graphics();                                  $ open graphics window
0203   deflight();                                  $ apply default lighting
0204   vvector( 2.5, 2.5, 1.0 );                    $ set view vector
0205   setup();                                     $ call setup function
0206   assemble();                                  $ do initial assembly
0207   echo_wire_frame( 0 );                        $ switch off wire frame
0208   echo_dsp( 1 );                               $ switch on surfaces
0209   rpnt();                                      $ repaint to show graphics
0210   zoom();                                      $ zoom to fit graphics area
0211
0212   menu fbc                                     $ start menu definition
0213   {
0214     button Reset
0215     { setup();                                 $ call setup functio
0216       rpnt();                                  $ repaint (clearing screen)
0217     }
0218     button Assemble
0219     { assemble();                              $ do the assembly
0220       rpnt( 0 );                               $ repaint without clearing
0221     }
0222     submenu Background
0223     { button Black
0224       { background( 0, 0, 0 );                 $ black background
0225         rpnt(1);                               $ clear and repaint
0226       }
0227       button Grey (0.33)
0228       { background( 0.33, 0.33, 0.33 );        $ dark grey background
0229         rpnt(1);                               $ clear and repaint
0230       }
0231       button Grey (0.67)
0232       { background( 0.67, 0.67, 0.67 );        $ light grey background
0233         rpnt(1);                               $ clear and repaint
0234       }
0235       button White
0236       { background( 1, 1, 1 );                 $ white background
0237         rpnt(1);                               $ clear and repaint
0238       }
0239     }
0240     submenu Views>
0241     { button Initial
0242       { vvector( 2.5, 2.5, 1.0 );              $ view initially used
0243         rpnt(1);                               $ clear and repaint
0244       }
0245       button Isometric
0246       { vvector( 1.0, 1.0, 1.0 );              $ isometric view
0247         rpnt(1);                               $ clear and repaint
0248       }
0249     }
0250     submenu Echo>
```

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

```
0251     { button Wireframe ON
0252       { echo_wire_frame(1);                  $ switch on wireframe
0253       }
0254       button Wireframe OFF
0255       { echo_wire_frame(0);                  $ switch off wireframe
0256       }
0257       button Surfaces (dsp) ON
0258       { echo_dsp(1);                         $ switch on dsp surfaces
0259       }
0260       button Surfaces (dsp) OFF
0261       { echo_dsp(0);                         $ switch off dsp surfaces
0262       }
0263       button Track (qq) ON
0264       { echo_qq(1);                          $ switch on track points
0265       }
0266       button Track (qq) OFF
0267       { echo_qq(0);                          $ switch off track points
0268       }
0269     }
0270     button Cycle(0)
0271     { cycle( 0 );                            $ call cycle function
0272     }
0273     button Cycle(1)
0274     { cycle( 1 );                            $ call cycle function
0275     }
0276     submenu Delay>
0277     { button 0
0278       { delay_factor = 0;                    $ no delay
0279       }
0280       button 1
0281       { delay_factor = 1;                    $ set delay_factor
0282       }
0283       button 2
0284       { delay_factor = 2;                    $ set delay_factor
0285       }
0286       button 5
0287       { delay_factor = 5;                    $ set delay_factor
0288       }
0289       button 10
0290       { delay_factor = 10;                   $ set delay_factor
0291       }
0292       button 20
0293       { delay_factor = 20;                   $ set delay_factor
0294       }
0295       button 50
0296       { delay_factor = 50;                   $ set delay_factor
0297       }
0298       button 100
0299       { delay_factor = 100;                  $ set delay_factor
0300       }
```

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

```
0301     }
0302   }
0303
0304   remmenu();                                    $ remove any existing menu
0305   addmenu( fbc );                               $ put up new menu
0306
0307   $ End of file
0308
```

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