Trio Motion Technology

BREATHING LIFE INTO MACHINES

Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292333
Fax: +44 (0)1684 297929

Web: www.triomotion.com

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
Ph: +1 412.968.9744
Fx: +1 412.968.9746

Doc No.: TN20-38
Version: 1.0
Date: 26th Sept 2001
Subject: Frame 5 Application Example

Application Note

Requirement

A machine is designed to apply adhesive in a pattern to a work-piece at intervals around its circumference. (see fig 1.) The glue nozzle is attached to a x – y positioning mechanism and the required profile is defined as a series of straight lines and curves merged together to make a continuous shape. This shape must be applied to each "pole" in turn BUT the work-piece is FIXED. i.e. it does not rotate.

It therefore follows that the "rotation" must be done in software.

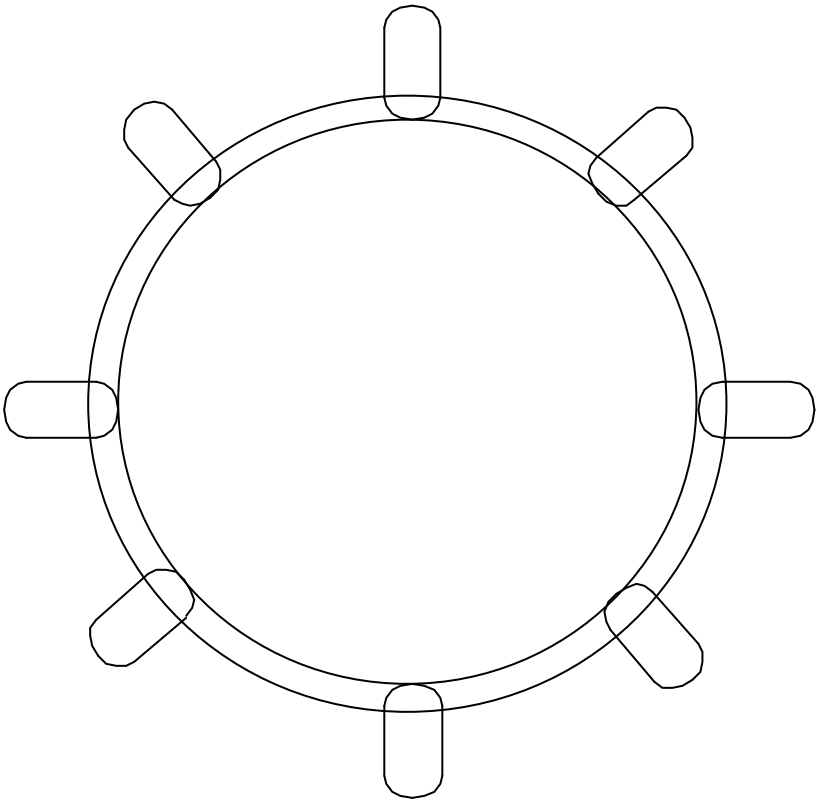
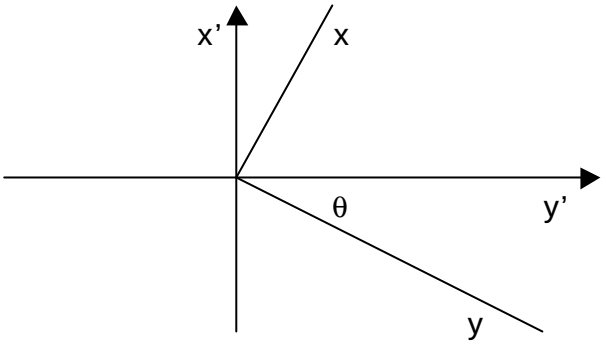


Figure 1 – 8 pole armature

FRAME Transform #5

In order to rotate the shape, a Frame Transform has been implemented in the system software that uses a 2x2 matrix to process the x / y co-ordinates supplied by the profile generator. This

Frame allows axes 0 and 1 to be “turned” through an angle so that command inputs to x, y (along the required plane) are transformed to the fixed axes x’ and y’. The coefficients of the 2 x 2 matrix can be derived from the required rotation angle of the operating plane.



The use of this transform allows the shape to be programmed as a sequence of moves in the horizontal plane, which can easily be turned through the required angle in order to repeat the pattern around the work-piece.

Calcualting the Matrix Coefficients.

For the frame to work, 2 sets of matrix coefficients must be entered, one for the forward transform and the second for the inverse. The transform calculates x and y according to the following:

```
(x’, y’) = (x, y) * ÉTABLE(0), TABLE(1)»
                    ÈTABLE(2), TABLE(3)¼
```

The inverse transform is calculated thus:

```
(x, y) = (x’, y’) * ÉTABLE(4), TABLE(5)»
                    ÈTABLE(6), TABLE(7)¼
```

For a given angle θ, the table entries are:

TABLE(0, COS(theta))	det = (TABLE(0) * TABLE(3)) - (TABLE(2) * TABLE(1))
TABLE(1, -SIN(theta))	TABLE(4, TABLE(3) / det)
TABLE(2, SIN(theta))	TABLE(5, -TABLE(1) / det)
TABLE(3, COS(theta))	TABLE(6, -TABLE(2) / det)
	TABLE(7, TABLE(0) / det)

Note: TABLE locations 0 to 7 are reserved for this function. User programs MUST NOT write to this area in the TABLE memory.

Frame 5 in Operation

The frame applies to axes 0 and 1 only. The axes should be datumed in FRAME=0. Once this is done, then the frame can be set to 5 and move commands directed at either axis or at both axes together in the usual way. However the actual movement of x’ and y’ (the real axes) will be according to the transform.

While frame 5 is set, DPOS represents the positions of x and y and MPOS shows the positions of the real axes x’ and y’. FE shows the following error in the usual way but the output of the transform (i.e. the demand positions of the real axes) is hidden.

If the axes need to be re-positioned according to the real axes, the frame can be turned off simply by setting FRAME=0. When this is done, the DPOS values will change to be the same as the MPOS positions, i.e. they become the positions in the x' / y' plane. The axes can then be moved to a new starting position and the frame set back to 5, perhaps with a new angle set.

Warning: Changing the matrix coefficients while the axes are in motion will result in unpredicable movement and could cause damage.

Example 1: Initialisation of Matrix Coefficients

```
x_axis = 0
y_axis = 1

FRAME = 0
WA(10)
DEFPOS(0, 0)

theta_degrees = 45  ‘ Set required angle in degrees
theta = theta_degrees * (2*PI/360)  ‘ Convert to radians
GOSUB calc_matrix
FRAME = 5

STOP

'================================================
' Calculate the matrix parameters for FRAME 5
' Transform (x, y) * (TABLE(0), TABLE(1) )
'                    (TABLE(2), TABLE(3) )
'
' Inverse Transform:
'         (x', y') * (TABLE(4), TABLE(5) )
'                    (TABLE(6), TABLE(7) )
'================================================
calc_matrix:
  ' forward transform
  TABLE(0, COS(theta))
  TABLE(1, -SIN(theta))
  TABLE(2, SIN(theta))
  TABLE(3, COS(theta))

  ' inverse transform
  det = (TABLE(0) * TABLE(3)) - (TABLE(2) * TABLE(1))
  TABLE(4, TABLE(3) / det)
  TABLE(5, -TABLE(1) / det)
  TABLE(6, -TABLE(2) / det)
  TABLE(7, TABLE(0) / det)
RETURN
```

Example 2: 8 pole sequential program

```
start:
  GOSUB inital

  '**** DATUM THE AXES ****
  GOSUB dat
```

```
  '**** MOVE TO HOME POSITION
  BASE(0)
  SPEED=trav_spd
  ACCEL=trav_acc

  MOVEABS(x_home,y_home)
  WAIT IDLE
  DEFPOS(0,0)
  GOSUB do_profile
STOP

'============================
'Do 8 Pole Profile
'============================

do_profile:
  BASE(0)
  MERGE=ON
  index=0
  ' Set Traversing speed
  SPEED=trav_spd
  ACCEL=trav_acc
  ' move to first position (at 360 degrees)
  MOVE(x_st,y_st)
  WAIT IDLE
  ' Go round in 45 degree steps
  FOR angle=0 TO 315 STEP 45
    theta=angle * (2*PI/360)
    GOSUB do_shape
    ' Frame zero allows next move to be on original x/y plane
    FRAME=0
    ' Move to next starting position
    MOVE(VR(10+index),VR(11+index))
    WAIT IDLE
    index=index+2
  NEXT angle
  MOVEABS(0, 0)
RETURN

'============================
'Shape sub-routine
'============================
do_shape:
  GOSUB calc_matrix
  FRAME=5
  len=VR(length)
  wid=VR(width)
  cr=VR(width)/2
  BASE(0)
  MOVE(wid/2,0)
  WAIT IDLE
  SPEED=profile_spd
  ACCEL=profile_acc
  OP(disp_sol,ON)
    MOVE(0,-((len/2)-cr))
    MOVECIRC(-wid,0,-wid/2,0,1)
    MOVE(0,((len)-2*cr))
    MOVECIRC(wid,0,wid/2,0,1)
    MOVE(0,-((len/2)-cr))
  WAIT IDLE
```

```
  OP(disp_sol,OFF)
  SPEED=trav_spd
  ACCEL=trav_acc
  MOVE(-(wid/2),0)
  WAIT IDLE
RETURN


'============================
'Intialisation
'============================
inital:
  '**axis positions**
  x_home=0
  y_home=200
  x_st=100
  y_st=-15
  rad=85

  '**System settings**
  profile_acc=3000
  profile_spd=300
  trav_spd=100
  trav_acc=1000

  '****Variable Settings****
  length=13
  width=14

  step_angle=45*2*PI/360
  ' coordinates for 45 degrees
  VR(10)=rad*SIN(step_angle)
  VR(11)=-rad*(1-COS(step_angle))
  ' coordinates for 90 degrees
  VR(12)=rad*(1-COS(step_angle))
  VR(13)=-rad*SIN(step_angle)
  ' coordinates for 135 degrees
  VR(14)=-rad*(1-COS(step_angle))
  VR(15)=-rad*SIN(step_angle)
  ' coordinates for 180 degrees
  VR(16)=-rad*SIN(step_angle)
  VR(17)=-rad*(1-COS(step_angle))
  ' coordinates for 225 degrees
  VR(18)=-rad*SIN(step_angle)
  VR(19)=rad*(1-COS(step_angle))
  ' coordinates for 270 degrees
  VR(20)=-rad*(1-COS(step_angle))
  VR(21)=rad*SIN(step_angle)
  ' coordinates for 315 degrees
  VR(22)=rad*(1-COS(step_angle))
  VR(23)=rad*SIN(step_angle)
RETURN

'============================
'DATUM ROUTINE
'============================
dat:
  BASE(0)
  DATUM_IN=0
  SPEED=30
```

```
  CREEP=3
  ACCEL=500
  DATUM(4)
  WAIT IDLE

  BASE(1)
  DATUM_IN=1
  SPEED=30
  CREEP=3
  ACCEL=500
  DATUM(4)
  WAIT IDLE
RETURN
```