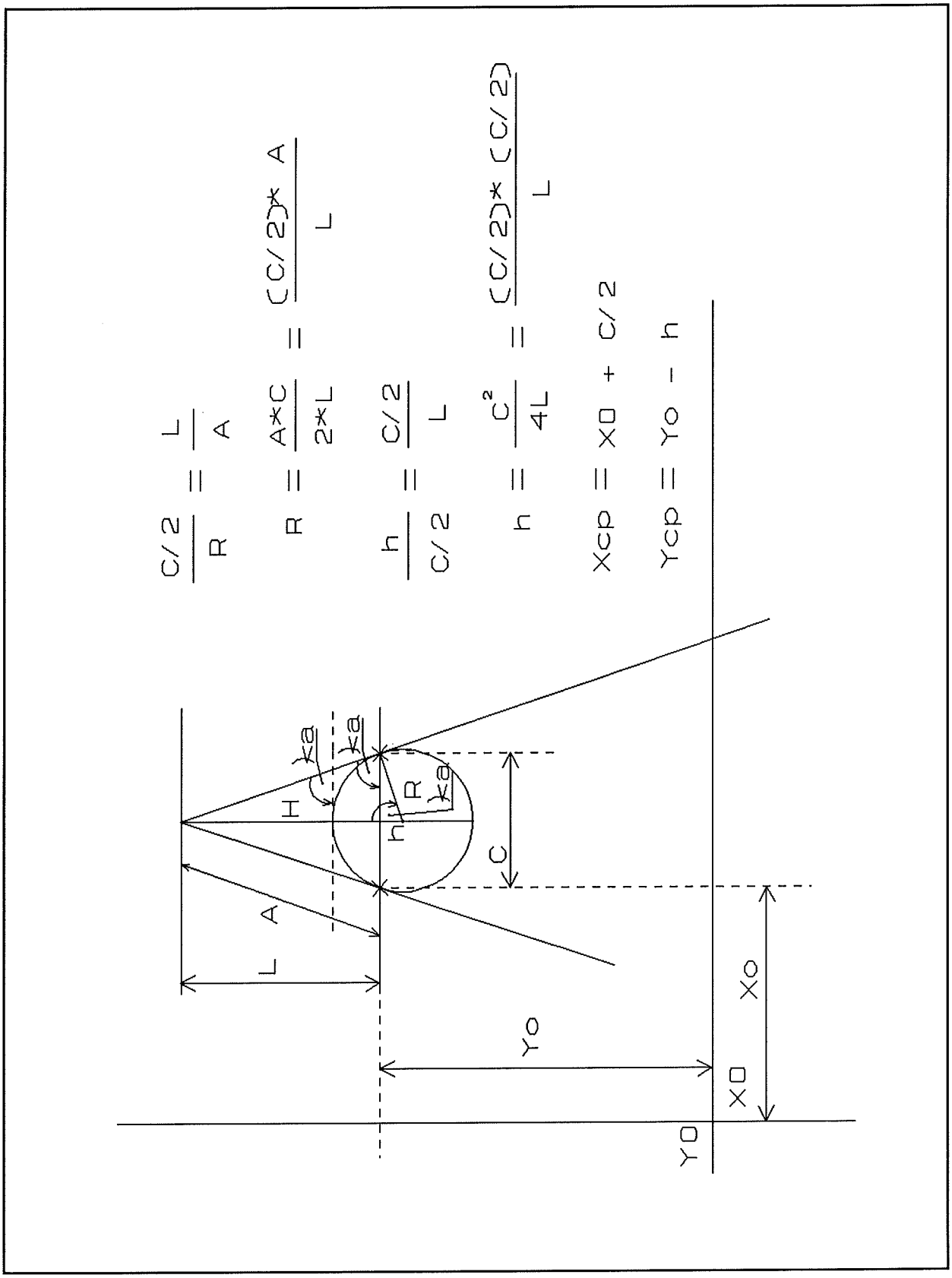
RE: Basic Program - Cubic Spline Interpolation

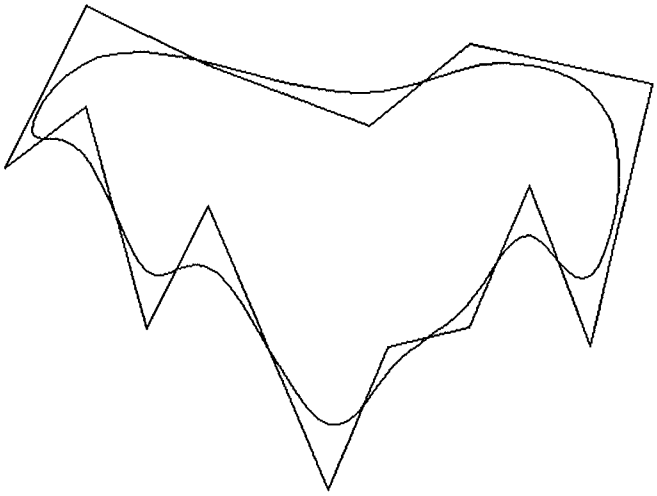
```
10   'save "spline.bas
19   '
20   CLS:SCREEN 2
21   '
30   'LIST THE SPLINE POINTS
40   '************************
50   '
60   DIM X(1,100)
70   DIM Y(1,100)
80   '
90   DIM XPT(12,1)
100  DIM YPT(12,1)
101  DIM PTSTOAD(12,1)
110  '
120  XPT(1,1)  =  5: YPT(1,1)  = 20: PTSTOAD(1,1)  =  15
130  XPT(2,1)  = 10: YPT(2,1)  = 10: PTSTOAD(2,1)  =  25
140  XPT(3,1)  = 15: YPT(3,1)  = 30: PTSTOAD(3,1)  =  15
150  XPT(4,1)  = 20: YPT(4,1)  = 10: PTSTOAD(4,1)  =  32
160  XPT(5,1)  = 25: YPT(5,1)  = 20: PTSTOAD(5,1)  =  43
170  XPT(6,1)  = 30: YPT(6,1)  = 10: PTSTOAD(6,1)  =  14
180  XPT(7,1)  = 35: YPT(7,1)  = 30: PTSTOAD(7,1)  =  32
190  XPT(8,1)  = 40: YPT(8,1)  = 10: PTSTOAD(8,1)  = 110
200  XPT(9,1)  = 45: YPT(9,1)  = 20: PTSTOAD(9,1)  =  25
205  XPT(10,1) = 50: YPT(10,1) = 10: PTSTOAD(10,1) =  35
206  XPT(11,1) = 55: YPT(11,1) = 30: PTSTOAD(11,1) =  64
207  XPT(12,1) =  5: YPT(12,1) = 30: PTSTOAD(12,1) =  64
209  STARTAT = 1
210  ENDAT = 10
211  XSCALE = 10: YSCALE = 5
219  '
220  'INITILIZE THE VARIABLES
230  '************************
240  '
250  XPTR = STARTAT: YPTR = STARTAT
255  OFSET = 0
260  LASTXDIFF = XPT(XPTR+1,1) - XPT(XPTR,1)
270  LASTYDIFF = YPT(YPTR+1,1) - YPT(YPTR,1)
275  PTSTOADD  = PTSTOAD(XPTR,1)
280  STEPSIZE  = 1 / (PTSTOADD + 1)
290  FLG1 = 0
300  '
330  'CALCULATE THE SEGMENT
335  '**********************
400  '
401  FOR X1 = STARTAT TO ENDAT
402  OFSET = 0
403  LASTXDIFF = XDIFF
404  LASTYDIFF = YDIFF
```

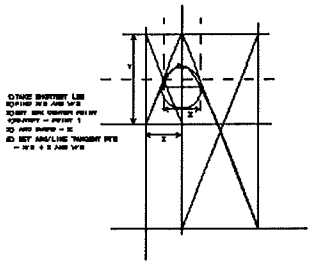
```
405  PTSTOADD = PTSTOAD(XPTR,1)
406  STEPSIZE = 1 / (PTSTOADD + 1)
410  GOSUB 1210           ;GET THE SEGMENT PTS
411  LOCATE 10,1:PRINT XPT(XPTR,1),YPT(YPTR,1)
412  A = XPT(XPTR-1,1)*XSCALE:B = YPT(YPTR-1,1)*YSCALE
415  PSET (A,B)
418  '
420  FOR X2 = 1 TO PTSTOADD
421  GOSUB 1010           'GET THE INTERP COEFFS
425  GOSUB 1300           'INTERP ALONG THE SEGMENT
430  A = NEWXPT*XSCALE:B = NEWYPT*YSCALE
432  PSET (A,B)
437  NEXT X2
438  '
440  XPTR = XPTR+1: YPTR = YPTR+1
445  IF FLG1 <> 0 GOTO 450
446  FLG1 = 11
447  GOTO 401
450  NEXT X1
451  '
460  'LOCATE 10,1:PRINT XPT(XPTR,1),YPT(YPTR,1)
461  A = XPT(XPTR-1,1)*XSCALE:B = YPT(YPTR-1,1)*YSCALE
462  PSET (A,B)
500  END
1000 '
1010 'CALCULATE THE INTERPOLATION COEFFICIENTS
1020 '*****************************************
1030 '
1040 OFSET  = OFSET + STEPSIZE
1050 OFSET2 = OFSET^2
1060 OFSET3 = OFSET^3
1061 '
1070 Z4 = OFSET3 - OFSET2
1080 Z3 = Z4 - OFSET2 + OFSET
1090 Z2 = -Z4 - Z3 + OFSET
1100 Z1 = 1 - Z2
1101 '
1110 RETURN
1200 '
1210 'CALCULATE THE SLOPE FOR EACH SEGMENT
1220 '************************************
1230 '
1240 LSTPTR = XPTR - 1
1250 NXTPTR = XPTR + 1
1260 MIDPTR = XPTR
1270 '
1280 XDIFF = (XPT(NXTPTR,1) - XPT(LSTPTR,1)) / 2
1290 YDIFF = (YPT(NXTPTR,1) - YPT(LSTPTR,1)) / 2
1291 '
1295 RETURN
1300 '
```

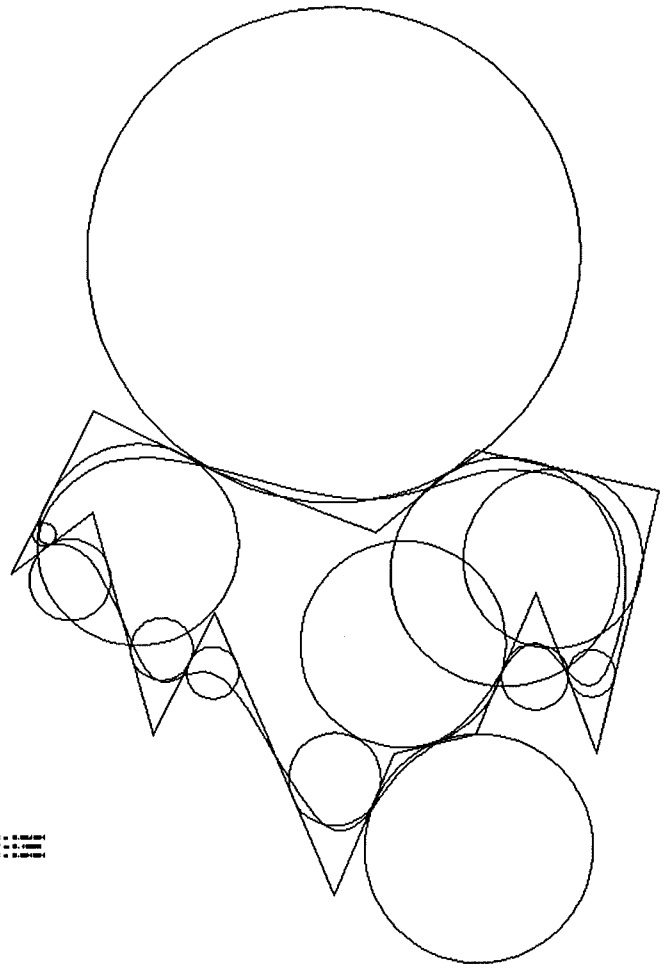
```
1310 'INTERPOLATE ALONG THE SEGMENT
1320 '*****************************
1330 '
1350 NEWXPT = Z1*XPT(LSTPTR,1) + Z2*XPT(MIDPTR,1) + Z3*LASTXDIFF +
             Z4*XDIFF
1360 NEWYPT = Z1*YPT(LSTPTR,1) + Z2*YPT(MIDPTR,1) + Z3*LASTYDIFF +
             Z4*YDIFF
1370 RETURN
```

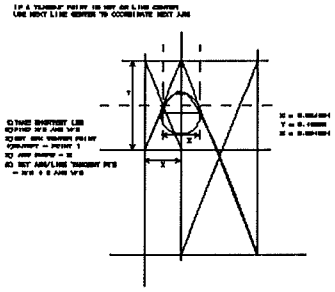
The following 4 pages show examples of the purpose behind the cubic splines. Sometimes refered to as curve-fitting, the intent of the cubic spline is to form a continuous curved path through a given set of points, anticipating a change in direction.

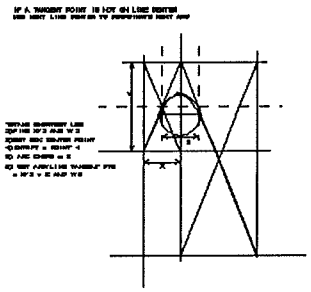

C/2 / R = L / A
R = A*C / 2*L = (C/2)* A / L
h / C/2 = C/2 / L
h = C² / 4L = (C/2)* (C/2) / L
Xcp = XO + C/2
Ycp = YO - h
H
h
R
C
A
L
YO
XO
XO
YO

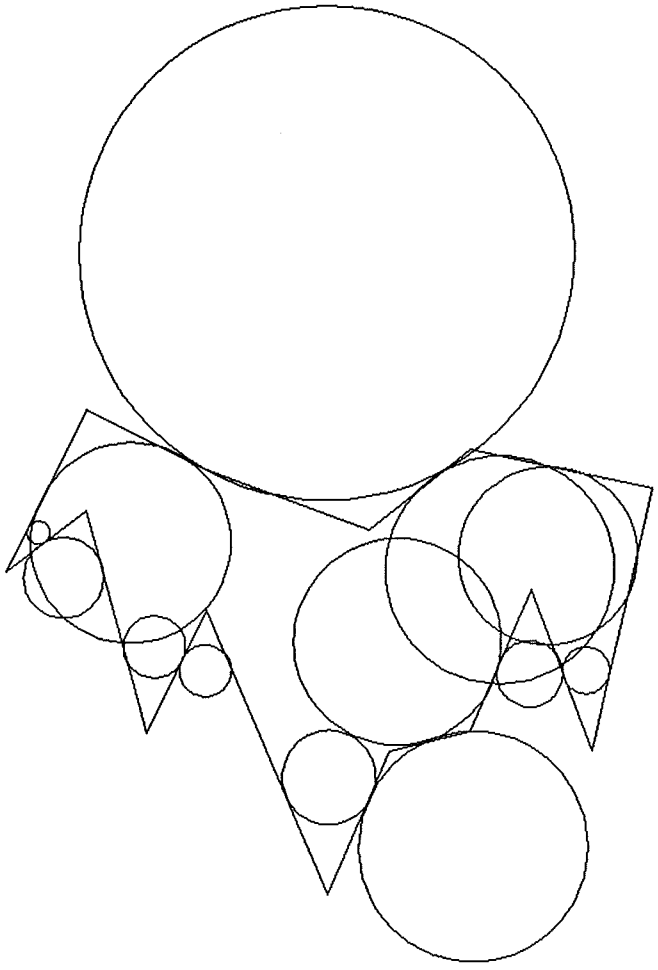












"C" Code Cubic Spline Algorithm

```
/*****************************************************************
```

pt_smooth(&contour_group, start_index, stop_index, num_add)

Purpose:

To generate smoothing points for piece contour

Inputs:

Pointer to contour point group
Index at which to start smoothing
Index at which to stop smoothing
Number of points to add on each segment

Returns:

next_curve_start if OK.
stop_index if ERROR

Description:

Allocate memory for # of points to be generated
If NO MEMORY AVAILABLE return (stop_index)

Copy all points up to and including the start_index to the new array

Initialize

last x - difference = x[start_index+1] - x[start_index]

Initialize

last y - difference = y[start_index+1] - y[start_index]
step_size = 1.0 / (# of points to add + 1)
offset = 0.

Do for step = 1 to # of points to insert for each point pair

```
{
    offset += step_size
    offset2 = offset * offset
    offset3 = offset * offset2
    z4 [step] = offset3 - offset2
    z3 [step] = z4 [step] - offset2 + offset
    z2 [step] = - z4 [step] - z3 [step] + offset
    z1 [step]  = 1. - z2[step]
}
```

```
newpt = start_index + 1

Do for each point trio
      (midpt from start_index + 1 to stop_index - 1)

{
     xdiff = (x[midpt +1] - x[midpt -1]) / 2.
     ydiff = (y[midpt +1] - y[midpt -1]) / 2.
}

Do for step = 1 to  # of pts to insert

{
     point_type [newpt] = SMOOTHING
     grade_rule [newpt] = 0

     xnew [newpt] = z1 [step] * x [midpt - 1] +
          z2 [step] * x [midpt] +
          z3 [step] * last_xdiff +
          z4 [step] * xdiff

     ynew[newpt]  = z1 [step] * y [midpt - 1] +
          z2 [step] * y [midpt] +
          z3 [step] * last_ydiff +
          z4 [step] * ydiff

     newpt++
}

Copy last point at midpt to new array

{
     newpt ++
     last_xdiff = xdiff
     last_ydiff = ydiff
}

handle last segment specially

{
     xdiff = ( x [stop_index] - x[stop_index -1])
     ydiff = ( y [stop_index] - y[stop_index -1])
}

Do for step = 1 to  # of pts to insert

{
     point_type [newpt] = SMOOTHING
     grade_rule [newpt] = 0

     xnew [newpt] = z1 [step] * x [stop_index-1] +
               z2 [step] * x [stop_index] +
               z3 [step] * last_xdiff +
```

```
                    z4 [step] * xdiff

          ynew [newpt] = z1 [step] * y [stop_index-1] +
                    z2 [step] * y [stop_index]   +
                    z3 [step] * last_ydiff +
                    z4 [step] * ydiff

          newpt ++
     }

     next_curve_start = newpt
     Copy points from stop_index to # of points to new array
     Update # of points in contour point group
     Update point array address in contour point group
     Release old point array memory
     return (next_curve_start)

*****************************************************************/

#include  <math.h>       /* types for math functions          */
#include  "mem.def"      /* memory functions definitions      */
#include  "global.inc"   /* global symbols definitions        */
#include  "data.inc"     /* Cuttex data types and symbols     */
#include  "pc_data.str"  /* structure def. for point data     */

/* max # of points to add per segment       */

#define        MAX_POINTS_TO_ADD   5

POINT_INDEX pt_smooth(contour_ptr,start_index,stop_index,num_add)

/* ptr to contour point group data          */

CONTOUR_GROUP_PTR   contour_ptr;

/* index at which to start smoothing        */

POINT_INDEX         start_index;

/* index at which to stop smoothing         */

POINT_INDEX         stop_index;

NUMBER_POINTS       num_add;

/* number of points to add on each segment */

{
CONTOUR_POINT_PTR data_ptr;          /* pnt to new data array   */
CONTOUR_POINT_PTR point_array;       /* pnt to contour dat array*/
CONTOUR_POINT_PTR point1_ptr;        /* 1st pt in point trio    */
CONTOUR_POINT_PTR midpt_ptr;         /* ptr to midpt in trio    */
CONTOUR_POINT_PTR point2_ptr;        /* ptr to 2nd. pt in trio  */
```

```
float     x_diff;                /* x-difference between 2 points */
float     y_diff;                /* y-difference between 2 points */
float     last_xdiff;            /* last x-diff  between 2 points */
float     last_ydiff;            /* last y-diff  between 2 points */
float     step_size;             /* size of each step             */
float     offset;                /*  offset  for  point  in  process
*/float   offset2;               /* offset square                 */
float     offset3;               /* offset cube                   */

int       step;                  /* current interval index        */

POINT_INDEX    index;            /* array index                   */
POINT_INDEX    midpt;            /* index of midpoint for pnt trio*/
POINT_INDEX next_curve_start;       /*  index  where  next  curve
starts */
POINT_INDEX    newpt;            /* array index for new points    */
NUMBER_POINTS number_new_points;  /* # of new generated pnts       */

float z1 [MAX_POINTS_TO_ADD+1]; /* smoothing formula coefficient */
float z2 [MAX_POINTS_TO_ADD+1]; /* smoothing formula coefficient */
float z3 [MAX_POINTS_TO_ADD+1]; /* smoothing formula coefficient */
float z4 [MAX_POINTS_TO_ADD+1]; /* smoothing formula coefficient */

NUMBER_POINTS number_to_allocate;  /* # points to allocate   */

/*********************************************************************/

/* force number of pts to insert to be <= to MAX_POINTS_TO_ADD  */

  if ( num_add > MAX_POINTS_TO_ADD ) num_add = MAX_POINTS_TO_ADD;

/* allocate and initialize memory for # of pnts to be generated */

  number_new_points  = num_add * ( stop_index - start_index );

  number_to_allocate =
               contour_ptr->number_points + number_new_points;

  if ((data_ptr = (CONTOUR_POINT_PTR)
      calloc(number_to_allocate, sizeof(CONTOUR_POINT)))==NULL_PTR)

     {
     log_error("MEMORY ALLOCATION ERROR: for point smoothing!\n");
     return (stop_index);
     }

  point_array = contour_ptr->point_data;

/* copy all points up to and including start_index to new array */

  for (index = 0; index <= start_index; index ++)
     memcpy(&data_ptr[index],&point_array[index],sizeof(CONTOUR_POINT));
```

```
/* initialize last x and y- differences to point[1] - point[0]  */

 last_xdiff =
    point_array[start_index +1].x -point_array[start_index].x;

 last_ydiff =
    point_array[start_index +1].y - point_array[start_index].y;

/* calculate interpolation coeficients  */

/*  ..  calculation  ..  */

  step_size = 1.0 / (num_add + 1.0);
  offset    = 0.0;

  for (step = 1; step <= num_add; step++)

       {
          offset   += step_size;
          offset2   = offset * offset;
          offset3   = offset * offset2;
          z4 [step] = offset3 - offset2;
          z3 [step] = z4 [step] - offset2 +  offset ;
          z2 [step] = -z4 [step] - z3 [step] + offset;
          z1 [step] = 1. - z2 [step];
       }

  newpt = start_index + 1;    /* set index for new added pnts    */

/* For each segment */

  for (midpt = start_index + 1; midpt < stop_index; midpt ++)

     {
          point1_ptr = &point_array[midpt -1];
          point2_ptr = &point_array[midpt +1];
          midpt_ptr = &point_array[midpt];

/*  ..  average previous and next diffs  ..  */

          x_diff = ( point2_ptr->x - point1_ptr->x ) / 2.0;
          y_diff = ( point2_ptr->y - point1_ptr->y ) / 2.0;

/*  ..  interpolate along the segment  ..  */

     for( step = 1; step <= num_add; step ++ )

     {
          /*  ..  adding new smoothing points  ..  */

          data_ptr[newpt].point_type = SMOOTHING;
          data_ptr[newpt].grade_rule = 0;
          data_ptr[newpt].x = z1 [step] * point1_ptr->x +
```

```
                z2 [step] * midpt_ptr->x  +
                z3 [step] * last_xdiff    +
                z4 [step] * x_diff;

            data_ptr[newpt].y = z1 [step] * point1_ptr->y +
                z2 [step] * midpt_ptr->y  +
                z3 [step] * last_ydiff    +
                z4[step] * y_diff;

            newpt ++;
        }                       /* endfor      */

/* copy midpoint to new point array */

/*  ..  algorithm does not move original  ..  */

/*  ..  points, only adds smoothing  ..  */

     memcpy(&data_ptr[newpt],midpt_ptr,sizeof(CONTOUR_POINT) );
     newpt ++;
     last_xdiff = x_diff;
     last_ydiff = y_diff;

       }                                /* endfor */

/* special handling for last segment */

/*  ..  no next segment for difference  ..  */

x_diff = point_array[stop_index].x - point_array[stop_index -1].x;
y_diff = point_array[stop_index].y - point_array[stop_index -1].y;

/*  ..  interpolate along final segment  ..  */

  for (step = 1; step <= num_add; step ++)
       {
          data_ptr[newpt].point_type = SMOOTHING;
          data_ptr[newpt].grade_rule = 0;

          data_ptr[newpt].x =
               z1 [step] * point_array[stop_index -1].x  +
               z2 [step] * point_array[stop_index].x     +
               z3 [step] * last_xdiff                     +
               z4 [step] * x_diff;

          data_ptr[newpt].y =
               z1 [step] * point_array[stop_index -1].y  +
               z2 [step] * point_array[stop_index].y     +
               z3 [step] * last_ydiff                     +
               z4 [step] * y_diff;

          newpt ++;
       }
```

```
/* set start index in new array */

  next_curve_start = newpt;

/* copy points from stop_index to end */

/*  ..  of contour to new array  ..  */

for (index = stop_index;index < contour_ptr->number_points;index++)
     {
memcpy(&data_ptr[newpt],&point_array[index],sizeof(CONTOUR_POINT));
          newpt ++;
     }

/* update # of points in contour */

  contour_ptr->number_points = newpt;

/* setup new array in contour defined */

  contour_ptr->point_data = data_ptr;
  free( point_array );        /* release old point array memory */
  point_array = NULL_PTR;

/* return start of next curve in the */

/*  ..  new point array  ..  */

  return (next_curve_start);

}
```

RE: Cubic Spline 8051 Assembly Code

```
;SPLINE FORMAT

;     1) LOAD VARIABLES IN X AND Y TABLES .... X,Y,NUM_OF_INTERP_PTS

;     2) CALL THE SPLINE MOVE WITH .... A) X and Y TABLE POINTERS
;                                       B) VELOCITY

;                    **********************************

;REGISTER LABLES

VAL1                EQU     16H
VAL2                EQU     17H
VAL3                EQU     18H
VAL4                EQU     19H
CHAR                EQU     1AH

STR11               EQU     1BH
STR12               EQU     1CH
STR13               EQU     1DH
STR14               EQU     1EH
SCHAR               EQU     1FH

FLAG1               EQU     20H
X_SLOP_NEG          BIT     FLAG1.0
Y_SLOP_NEG          BIT     FLAG1.1

STR21               EQU     22H
STR22               EQU     23H
STR23               EQU     24H
STR24               EQU     25H

SIZE_OF_TABLEL      EQU     26H
SIZE_OF_TABLEH      EQU     27H

XLST_LPTR                   EQU     28H
;XMID_PTR-1
XLST_HPTR           EQU     29H

XMID_LPTR           EQU     2AH                     ;XMID_PTR
XMID_HPTR           EQU     2BH

XNXT_LPTR           EQU     2CH                     ;XMID_PTR+1
XNXT_HPTR           EQU     2DH

YLST_LPTR           EQU     2EH                     ;YMID_PTR-1
YLST_HPTR           EQU     2FH

YMID_LPTR           EQU     30H                     ;YMID_PTR
YMID_HPTR           EQU     31H

YNXT_LPTR           EQU     32H                     ;YMID_PTR+1
YNXT_HPTR           EQU     33H

NUM_OF_PTSL         EQU     34H
NUM_OF_PTSH         EQU     35H

VEL1                EQU     36H
VEL2                EQU     37H
VEL3                EQU     38H
VEL4                EQU     39H
```

```
OFSET11                 EQU     36H             ;FRACTION
OFSET12                 EQU     37H             ;FRACTION
OFSET13                 EQU     38H             ;INTEGER
OFSET14                 EQU     39H             ;INTEGER
X_VEL1                  EQU     3AH
X_VEL2                  EQU     3BH
X_VEL3                  EQU     3CH
X_VEL4                  EQU     3DH

OFSET21                 EQU     3AH             ;FRACTION
OFSET22                 EQU     3BH             ;FRACTION
OFSET23                 EQU     3CH             ;INTEGER
OFSET24                 EQU     3DH             ;INTEGER

Y_VEL1                  EQU     3EH
Y_VEL2                  EQU     3FH
Y_VEL3                  EQU     40H
Y_VEL4                  EQU     41H

OFSET31                 EQU     3EH                     ;FRACTION
OFSET32                 EQU     3FH                     ;FRACTION
OFSET33                 EQU     40H                     ;INTEGER
OFSET34                 EQU     41H                     ;INTEGER

Z4_1                    EQU     42H
Z4_2                    EQU     43H
Z4_3                    EQU     44H
Z4_4                    EQU     45H
Z4_5                    EQU     46H

Z3_1                    EQU     47H
Z3_2                    EQU     48H
Z3_3                    EQU     49H
Z3_4                    EQU     4AH
Z3_5                    EQU     4BH

Z2_1                    EQU     4CH
Z2_2                    EQU     4DH
Z2_3                    EQU     4EH
Z2_4                    EQU     4FH
Z2_5                    EQU     50H

Z1_1                    EQU     51H
Z1_2                    EQU     52H
Z1_3                    EQU     53H
Z1_4                    EQU     54H
Z1_5                    EQU     55H

X_DIFF1                 EQU     56H
X_DIFF2                 EQU     57H
X_DIFF3                 EQU     58H
X_DIFF4                 EQU     59H
X_DIFF5                 EQU     59H

Y_DIFF1                 EQU     5AH
Y_DIFF2                 EQU     5BH
Y_DIFF3                 EQU     5CH
Y_DIFF4                 EQU     5DH
Y_DIFF5                 EQU     59H

LAST_X_DIFF1            EQU     5EH
LAST_X_DIFF2            EQU     5FH
LAST_X_DIFF3            EQU     60H
LAST_X_DIFF4            EQU     61H
LAST_X_DIFF5            EQU     61H

LAST_Y_DIFF1            EQU     62H
LAST_Y_DIFF2            EQU     63H
LAST_Y_DIFF3            EQU     64H
LAST_Y_DIFF4            EQU     65H
LAST_Y_DIFF5            EQU     65H
```

```
STEP_SIZE1          EQU     66H                         ;FRACTION
STEP_SIZE2          EQU     67H                         ;FRACTION
STEP_SIZE3          EQU     68H                         ;INTEGER
STEP_SIZE4          EQU     69H                         ;INTEGER

X_POS1              EQU     6AH
X_POS2              EQU     6BH
X_POS3              EQU     6CH
X_POS4              EQU     6DH

Y_POS1              EQU     6EH
Y_POS2              EQU     6FH
Y_POS3              EQU     70H
Y_POS4              EQU     71H

X_LPOS1             EQU     72H
X_LPOS2             EQU     73H
X_LPOS3             EQU     74H
X_LPOS4             EQU     75H

X_INC1              EQU     72H
X_INC2              EQU     73H
X_INC3              EQU     74H
X_INC4              EQU     75H

Y_LPOS1             EQU     76H
Y_LPOS2             EQU     77H
Y_LPOS3             EQU     78H
Y_LPOS4             EQU     79H

Y_INC1              EQU     76H
Y_INC2              EQU     77H
Y_INC3              EQU     78H
Y_INC4              EQU     79H

AXIS_LPTR           EQU     7AH
AXIS_HPTR           EQU     7BH

;                           ------------------------------

X_TABLE:            DW      X1P1,X1P2,X1P3,X1P4,0000,0000   ;1st X point pos.
                    DW      X2P1,X2P2,X2P3,X2P4,PTSL,PTSH
                    DW      X3P1,X3P2,X3P3,X3P4,PTSL,PTSH
                    DW      X4P1,X4P2,X4P3,X4P4,PTSL,PTSH
                    DW      etc. ....

Y_TABLE:            DW      Y1P1,Y1P2,Y1P3,Y1P4,0000,0000   ;1st Y point pos.
                    DW      Y2P1,Y2P2,Y2P3,Y2P4,PTSL,PTSH
                    DW      Y3P1,Y3P2,Y3P3,Y3P4,PTSL,PTSH
                    DW      Y4P1,Y4P2,Y4P3,Y4P4,PTSL,PTSH
                    DW      etc. ....

;                       ******************************
;                       Setup the X axis table pointers

SPLINE:             MOV     DPTR,#X_TABLE
                    MOV     A,DPL
                    ADD     A,#06
                    MOV     XMID_LPTR,A
                    MOV     A,DPH
                    ADDC    A,#00
                    MOV     XMID_HPTR,A

;                           ------------------------------      8us
;                       Setup the Y axis table pointers

                    MOV     DPTR,#Y_TABLE
                    MOV     A,DPL
                    ADD     A,#06
                    MOV     YMID_LPTR,A
```

```
                MOV     A,DPH
                ADDC    A,#00
                MOV     YMID_HPTR,A

;                       ----------------------------        16us
;               clear the offset registers

                CLR     OFSET11
                CLR     OFSET12
                CLR     OFSET13
                CLR     OFSET14

;                       ----------------------------        20us
;       Get the X difference value, and put into the difference register

                MOV     PSW,#00
                MOV     DPTR,#X_TABLE
                MOVX    A,@DPTR
                MOV     LAST_X_DIFF1,A

                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_X_DIFF2,A

                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_X_DIFF3,A

                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_X_DIFF4,A

                MOV     DPTR,#X_TABLE+6
                MOVX    A,@DPTR
                SUBB    A,LAST_X_DIFF1

                MOV     VAL1,A
                INC     DPTR
                MOVX    A,@DPTR

                SUBB    A,LAST_X_DIFF2
                MOV     VAL2,A
                INC     DPTR

                MOVX    A,@DPTR
                SUBB    A,LAST_X_DIFF3
                MOV     VAL3,A

                INC     DPTR
                MOVX    A,@DPTR
                SUBB    A,LAST_X_DIFF4

                MOV     VAL4,A

                CALL    GET_LOG

                MOV     LAST_X_DIFF1,VAL1
                MOV     LAST_X_DIFF2,VAL2
                MOV     LAST_X_DIFF3,VAL3
                MOV     LAST_X_DIFF4,VAL4
                MOV     LAST_X_DIFF5,CHAR

;                       ----------------------------        125us
;       Get the Y difference value, and put into the difference register

                MOV     PSW,#00
                MOV     DPTR,#Y_TABLE
                MOVX    A,@DPTR
                MOV     LAST_Y_DIFF1,A
                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_Y_DIFF2,A
```

```
                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_Y_DIFF3,A
                INC     DPTR
                MOVX    A,@DPTR
                MOV     LAST_Y_DIFF4,A

                MOV     DPTR,#Y_TABLE+6
                MOVX    A,@DPTR
                SUBB    A,LAST_Y_DIFF1
                MOV     VAL1,A

                INC     DPTR
                MOVX    A,@DPTR
                SUBB    A,LAST_Y_DIFF2
                MOV     VAL2,A

                INC     DPTR
                MOVX    A,@DPTR
                SUBB    A,LAST_Y_DIFF3
                MOV     VAL3,A

                INC     DPTR
                MOVX    A,@DPTR
                SUBB    A,LAST_Y_DIFF4
                MOV     VAL4,A

                CALL    GET_LOG

                MOV     LAST_X_DIFF1,VAL1
                MOV     LAST_X_DIFF2,VAL2
                MOV     LAST_X_DIFF3,VAL3
                MOV     LAST_X_DIFF4,VAL4
                MOV     LAST_X_DIFF5,CHAR
;                       ----------------------------        230us

CHK_R2:         MOV     R2,#SIZE_OF_TABLEL
                CJNE    R2,#00,CHK_R3
                INC     R2

CHK_R3:         MOV     R3,#SIZE_OF_TABLEH
                CJNE    R3,#00,BGN_THE_CALCS
                INC     R3

                JMP     BGN_THE_CALCS

;                       ********************************* 240us
;               Calculate succeeding segments, clear the offset registers

NXT_SEGMENT:    CLR     OFSET11
                CLR     OFSET12
                CLR     OFSET13
                CLR     OFSET14

;               Transfer the X/Y differences to the last difference regs.

                MOV     LAST_X_DIFF1,X_DIFF1
                MOV     LAST_X_DIFF2,X_DIFF2
                MOV     LAST_X_DIFF3,X_DIFF3
                MOV     LAST_X_DIFF4,X_DIFF4
                MOV     LAST_X_DIFF5,X_DIFF5

                MOV     LAST_Y_DIFF1,Y_DIFF1
                MOV     LAST_Y_DIFF2,Y_DIFF2
                MOV     LAST_Y_DIFF3,Y_DIFF3
                MOV     LAST_Y_DIFF4,Y_DIFF4
                MOV     LAST_Y_DIFF5,Y_DIFF5
```

```
;                                   ******************************      24us
;                                     on new segmnt / from start

BGN_THE_CALCS:              ;   24us    /   240us

;STEPSIZE = 1 / (PTSTOADD + 1)          FOR THIS MATH ....
;                                       PTSTOADD = PTSTOADD * 2^16   1 = 1 * 2^16

GET_STEP_SIZE:              MOV     VAL1,#00
                            MOV     VAL2,#00

                            MOV     A,NUM_OF_PTSL
                            ADD     A,#01
                            MOV     VAL3,A

                            MOV     A,NUM_OF_PTSH
                            ADDC    A,#00
                            MOV     VAL4,A

                            CALL    GET_LOG

                            MOV     PSW,#00
                            MOV     A,#00
                            SUBB    A,VAL1
                            MOV     VAL1,A

                            MOV     A,#00
                            SUBB    A,VAL2
                            MOV     VAL2,A

                            MOV     A,#07               ;CORRECTION FACTOR
                            SUBB    A,VAL3
                            MOV     VAL3,A

                            MOV     A,#00
                            SUBB    A,VAL4
                            MOV     VAL4,A

                            MOV     A,#16
                            SUBB    A,CHAR
                            MOV     CHAR,A

                            CALL    ANTI_LOG

                             MOV    STEP_SIZE1,VAL1     ;FRACTION
                             MOV    STEP_SIZE2,VAL2     ;FRACTION
                             MOV    STEP_SIZE3,VAL3     ;INTEGER
                             MOV    STEP_SIZE4,VAL4     ;INTEGER

;                                   ----------------------------    156us / 372us
;                       If the num of points to add = 0 ... abort the move

CHK_R4:                     MOV     R4,#NUM_OF_PTSL
                            CJNE    R4,#00,CHK_R5
                            INC     R4

CHK_R5:                     MOV     R5,#NUM_OF_PTSH
                            CJNE    R5,#00,GT_SEG_PTRS
                            INC     R5

;                                   ----------------------------    164us / 380us
;Get the new Trio LAST and NEXT segment pointers.  Setup the X axis table pointers.

GT_SEG_PTRS:                MOV     PSW,#00
                            MOV     A,XMID_LPTR
                            SUBB    A,#06
                            MOV     XLST_LPTR,A
                            MOV     A,XMID_HPTR
                            SUBB    A,#00
                            MOV     XLST_HPTR,A
```

```
                    MOV     A,XMID_LPTR
                    ADD     A,#06
                    MOV     XNXT_LPTR,A
                    MOV     A,XMID_LPTR
                    ADDC    A,#00
                    MOV     XNXT_HPTR,A

;                           ----------------------------        177us / 393us
;                       Setup the Y axis table pointers

                    MOV     PSW,#00
                    MOV     A,YMID_LPTR
                    SUBB    A,#06
                    MOV     YLST_LPTR,A
                    MOV     A,YMID_HPTR
                    SUBB    A,#00
                    MOV     YLST_HPTR,A

                    MOV     A,YMID_LPTR
                    ADD     A,#06
                    MOV     YNXT_LPTR,A
                    MOV     A,YMID_LPTR
                    ADDC    A,#00
                    MOV     YNXT_HPTR,A

;                           ----------------------------        190us / 406us
;                       Calculate the new X segment Slope

                    MOV     DPL,XLST_LPTR
                    MOV     DPH,XLST_HPTR
                    MOVX    A,@DPTR
                    MOV     X_DIFF1,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    MOV     X_DIFF2,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    MOV     X_DIFF3,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    MOV     X_DIFF4,A

                    MOV     PSW,#00
                    MOV     DPL,XNXT_LPTR
                    MOV     DPH,XNXT_HPTR
                    MOVX    A,@DPTR
                    SUBB    A,X_DIFF1
                    MOV     X_DIFF1,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    SUBB    A,X_DIFF2
                    MOV     X_DIFF2,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    SUBB    A,X_DIFF3
                    MOV     X_DIFF3,A
                    INC     DPTR
                    MOVX    A,@DPTR
                    SUBB    A,X_DIFF4
                    MOV     X_DIFF4,A

                    JNB     ACC.7,X_SLOPE_OK    ;Negative slope ?

                    MOV     R4,X_DIFF1          ;Yes

                    CALL    CONVERT_ANS         ;convert to + val

                    SETB    X_SLOP_NEG

X_SLOPE_OK:         MOV     R4,X_DIFF4              ;Divide a signed ans by 2
                    CALL    DIV_BY_2
                    CALL    GET_LOG
```

```
                        MOV     X_DIFF1,VAL1
                        MOV     X_DIFF2,VAL2
                        MOV     X_DIFF3,VAL3
                        MOV     X_DIFF4,VAL4
                        MOV     X_DIFF5,CHAR

;                               ----------------------------        348us  /  564us
;                       Calculate the new Y segment Slope

                        MOV     DPL,YLST_LPTR
                        MOV     DPH,YLST_HPTR
                        MOVX    A,@DPTR
                        MOV     Y_DIFF1,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     Y_DIFF2,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     Y_DIFF3,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     Y_DIFF4,A

                        MOV     PSW,#00
                        MOV     DPL,YNXT_LPTR
                        MOV     DPH,YNXT_HPTR
                        MOVX    A,@DPTR
                        SUBB    A,Y_DIFF1
                        MOV     Y_DIFF1,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        SUBB    A,Y_DIFF2
                        MOV     Y_DIFF2,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        SUBB    A,Y_DIFF3
                        MOV     Y_DIFF3,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        SUBB    A,Y_DIFF4
                        MOV     Y_DIFF4,A

                        JNB     ACC.7,Y_SLOPE_OK        ;Negative slope ?

                        MOV     R4,Y_DIFF1              ;Yes -

                        CALL    CONVERT_ANS             ;convert to a signed + val

                        SETB    Y_SLOP_NEG

Y_SLOPE_OK:             MOV     R4,Y_DIFF1              ;Divide the ans by 2
                        CALL    DIV_BY_2

                        CALL    GET_LOG

                        MOV     Y_DIFF1,VAL1
                        MOV     Y_DIFF2,VAL2
                        MOV     Y_DIFF3,VAL3
                        MOV     Y_DIFF4,VAL4
                        MOV     Y_DIFF5,CHAR

;                           ****************************        506us  /  722us
;                         MOVE TO A GIVEN X and Y POINT

                        CALL    LOAD_AND_MOVE

;                           ****************************
;               Calc the Interpolation coefficients - OFSET = OFSET + STEPSIZE

PLOT_NXT_PNT:           MOV     A,STEP_SIZE1            ;FRACTION
                        ADD     A,OFSET11
                        MOV     OFSET11,A
```

```
                MOV     VAL1,A

                MOV     A,STEP_SIZE2            ;FRACTION
                ADDC    A,OFSET12
                MOV     OFSET12,A
                MOV     VAL2,A

                MOV     A,STEP_SIZE3                    ;INTEGER
                ADDC    A,OFSET13
                MOV     OFSET13,A
                MOV     VAL3,A

                MOV     A,STEP_SIZE4            ;INTEGER
                ADDC    A,OFSET14
                MOV     OFSET14,A
                MOV     VAL4,A

                CALL    GET_LOG

                MOV     STR11,VAL1
                MOV     STR12,VAL2
                MOV     STR13,VAL3
                MOV     STR14,VAL4
                MOV     SCHAR,CHAR

;                       ----------------------------    / 76us
;               OFSET2 = OFSET^2

                MOV     A,VAL1                  ;Val^2
                ADD     A,VAL1
                MOV     VAL1,A

                MOV     A,VAL2
                ADDC    A,VAL2
                MOV     VAL2,A

                MOV     A,VAL3
                ADDC    A,VAL3
                MOV     VAL3,A

                MOV     A,VAL4
                ADDC    A,VAL4
                MOV     VAL4,A

                MOV     A,CHAR
                ADDC    A,CHAR
                MOV     CHAR,A

;                       ----------------------------    / 91us
                MOV     A,STR11                 ;VAL^3
                ADD     A,VAL1
                MOV     STR11,A

                MOV     A,STR12
                ADDC    A,VAL2
                MOV     STR12,A

                MOV     A,STR13
                ADDC    A,VAL3
                MOV     STR13,A

                MOV     A,STR14
                ADDC    A,VAL4
                MOV     STR14,A

                MOV     A,SCHAR
                ADDC    A,CHAR
                MOV     SCHAR,A

;                       ----------------------------    / 106us

                CALL    ANTI_LOG                ;GET AND STOR OFSET^2
```

```
                        MOV     OFSET21,VAL1
                        MOV     OFSET22,VAL2
                        MOV     OFSET23,VAL3
                        MOV     OFSET24,VAL4

;                               ----------------------------        /  164us
;                             OFSET3 = OFSET^3

                        MOV     VAL1,STR11              ;RECOVER OFSET^3
                        MOV     VAL2,STR12
                        MOV     VAL3,STR13
                        MOV     VAL4,STR14
                        MOV     CHAR,SCHAR

                        CALL    ANTI_LOG

                        MOV     OFSET31,VAL1
                        MOV     OFSET32,VAL2
                        MOV     OFSET33,VAL3
                        MOV     OFSET34,VAL4            ;10 + 50 + 8

;                               ----------------------------        /  232us

;                   .... Z4 is always negative .... Z1-Z3 are always positive
;                                 OFSET1x > OFSET2x > OFSET3x
;                                   .5    >   .25   >  .125

;                               ----------------------------
;Z4 = OFSET3 - OFSET2
;Z4 = (OFS1 * OFS1 * OFS1) - (OFS1 * OFS1) = .125 - .25 =  - .125
;Z4 =  .125  - .25 = - .125    --> STR1x

                        MOV     PSW,#00                 ;Z4 = - ABS (OFSET2 - OFSET3)
                        MOV     A,OFSET21
                        SUBB    A,OFSET31
                        MOV     VAL1,A
                        MOV     STR11,A

                        MOV     A,OFSET22
                        SUBB    A,OFSET32
                        MOV     VAL2,A
                        MOV     STR12,A

                        MOV     A,OFSET23
                        SUBB    A,OFSET33
                        MOV     VAL3,A
                        MOV     STR13,A

                        MOV     A,OFSET24
                        SUBB    A,OFSET34
                        MOV     VAL4,A
                        MOV     STR14,A

                        CALL    GET_LOG

                        MOV     Z4_1,VAL1
                        MOV     Z4_2,VAL2
                        MOV     Z4_3,VAL3
                        MOV     Z4_4,VAL4
                        MOV     Z4_5,CHAR

;                               ----------------------------        /  309us
;Z3 = Z4 - OFSET2 + OFSET
;Z3 = .5 - .125 - .25 =  .125
;  = (OFSET3 - OFSET2) - OFSET2 + OFSET1
;  =     (-STR11-4)    - OFSET2 + OFSET1
;  =     + OFSET1  - (STR1)  - OFSET2      --> STR2x

                        MOV     PSW,#00
                        MOV     A,OFSET11
                        SUBB    A,STR11
                        MOV     Z3_1,A
```

```
                        MOV     A,OFSET12
                        SUBB    A,STR12
                        MOV     Z3_2,A

                        MOV     A,OFSET13
                        SUBB    A,STR13
                        MOV     Z3_3,A

                        MOV     A,OFSET14
                        SUBB    A,STR14
                        MOV     Z3_4,A

;                               ----------------------------    / 322us

                        MOV     PSW,#00
                        MOV     A,Z3_1
                        SUBB    A,OFSET21
                        MOV     VAL1,A
                        MOV     STR21,A

                        MOV     A,Z3_2
                        SUBB    A,OFSET22
                        MOV     VAL2,A
                        MOV     STR22,A

                        MOV     A,Z3_3
                        SUBB    A,OFSET23
                        MOV     VAL3,A
                        MOV     STR23,A

                        MOV     A,Z3_4
                        SUBB    A,OFSET24
                        MOV     VAL4,A
                        MOV     STR24,A

                        CALL    GET_LOG

                        MOV     Z3_1,VAL1
                        MOV     Z3_2,VAL2
                        MOV     Z3_3,VAL3
                        MOV     Z3_4,VAL4
                        MOV     Z3_5,CHAR

;                               ----------------------------    / 399us
;Z2 = -Z4 - Z3 + OFSET
;Z2 = .125 + .5 - .125 = .5
;   = - (OFSET3 - OFSET2) - ((OFSET3 - OFSET2) - OFSET2 + OFSET)
;                                                                   + OFSET
;   =                    - (-STR1) - (STR2) + OFSET1
;   =                    + (STR1) - (STR2) + OFSET1
;   =                    + (STR1) + OFSET1 - (STR2)      --->  STR2x

                        MOV     A,STR11
                        ADD     A,OFSET11
                        MOV     Z2_1,A

                        MOV     A,STR22
                        ADDC    A,OFSET12
                        MOV     Z2_2,A

                        MOV     A,STR23
                        ADDC    A,OFSET13
                        MOV     Z2_3,A

                        MOV     A,STR24
                        ADDC    A,OFSET14
                        MOV     Z2_4,A

;                               ----------------------------    / 411us
```

```
                        MOV     PSW,#00
                        MOV     A,Z2_1
                        SUBB    A,STR21
                        MOV     Z2_1,A
                        MOV     STR11,A

                        MOV     A,Z2_2
                        SUBB    A,STR22
                        MOV     Z2_2,A
                        MOV     STR12,A

                        MOV     A,Z2_3
                        SUBB    A,STR23
                        MOV     Z2_3,A
                        MOV     STR13,A

                        MOV     A,Z2_4
                        SUBB    A,STR24
                        MOV     Z2_4,A
                        MOV     STR14,A

                        CALL    GET_LOG

                        MOV     Z2_1,VAL1
                        MOV     Z2_2,VAL2
                        MOV     Z2_3,VAL3
                        MOV     Z2_4,VAL4
                        MOV     Z2_5,CHAR

;                               ----------------------------        /  488us
;Z1 = 1 - Z2
;Z1 = 1 - .5 =  .5
                        MOV     PSW,#00
                        MOV     A,#00
                        SUBB    A,STR11
                        MOV     VAL1,A

                        MOV     A,#00
                        SUBB    A,STR12
                        MOV     VAL2,A

                        MOV     A,#00
                        SUBB    A,STR13
                        MOV     VAL3,A

                        MOV     A,#01
                        SUBB    A,STR14
                        MOV     VAL4,A

                        CALL    GET_LOG

                        MOV     Z1_1,VAL1
                        MOV     Z1_2,VAL2
                        MOV     Z1_3,VAL3
                        MOV     Z1_4,VAL4
                        MOV     Z1_5,CHAR

;                               ----------------------------        /  561us
;                       INTERPOLATE ALONG THE SEGMENT X/Y POINTS ....
;       NEWXPT = Z1 * XPT(LSTPTR,1) + 2 * XPT(MIDPTR,1) + Z3 * LASTXDIFF + Z4 * XDIFF
;                   AFTER THIS ROUTINE RECOVER OFSET DECIMAL PT POSITION

;                               ----------------------------

                        MOV     DPL,XLST_LPTR
                        MOV     DPH,XLST_HPTR
                        MOVX    A,@DPTR
                        MOV     VAL1,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     VAL2,A
                        INC     DPTR
```

```
        MOVX    A,@DPTR
        MOV     VAL3,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     VAL4,A

        CALL    GET_LOG

        MOV     A,VAL1
        ADD     A,Z1_1
        MOV     VAL1,A
        MOV     A,VAL2
        ADDC    A,Z1_2
        MOV     VAL2,A
        MOV     A,VAL3
        ADDC    A,Z1_3
        MOV     VAL3,A
        MOV     A,VAL4
        ADDC    A,Z1_4
        MOV     VAL4,A
        MOV     A,CHAR
        ADDC    A,Z1_5
        MOV     CHAR,A

        CALL    ANTI_LOG

        MOV     X_POS1,VAL1
        MOV     X_POS2,VAL2
        MOV     X_POS3,VAL3
        MOV     X_POS4,VAL4

;               ----------------------------    / 706us

        MOV     DPL,XMID_LPTR
        MOV     DPH,XMID_HPTR
        MOVX    A,@DPTR
        MOV     VAL1,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     VAL2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     VAL3,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     VAL4,A

        CALL    GET_LOG

        MOV     A,VAL1
        ADD     A,Z2_1
        MOV     VAL1,A
        MOV     A,VAL2
        ADDC    A,Z2_2
        MOV     VAL2,A
        MOV     A,VAL3
        ADDC    A,Z2_3
        MOV     VAL3,A
        MOV     A,VAL4
        ADDC    A,Z2_4
        MOV     VAL4,A
        MOV     A,CHAR
        ADDC    A,Z2_5
        MOV     CHAR,A

        CALL    ANTI_LOG

        MOV     A,VAL1
        ADD     A,X_POS1
        MOV     X_POS1,A
        MOV     A,VAL2
        ADDC    A,X_POS2
```

```
                MOV     X_POS2,A
                MOV     A,VAL3
                ADDC    A,X_POS3
                MOV     X_POS3,A
                MOV     A,VAL4
                ADDC    A,X_POS4
                MOV     X_POS4,A

;                       ----------------------------        / 855us

                MOV     A,LAST_X_DIFF1
                ADD     A,Z3_1
                MOV     VAL1,A
                MOV     A,LAST_X_DIFF2
                ADDC    A,Z3_2
                MOV     VAL2,A
                MOV     A,LAST_X_DIFF3
                ADDC    A,Z3_3
                MOV     VAL3,A
                MOV     A,LAST_X_DIFF4
                ADDC    A,Z3_4
                MOV     VAL4,A
                MOV     A,LAST_X_DIFF5
                ADDC    A,Z3_5
                MOV     CHAR,A

                CALL    ANTI_LOG

                MOV     A,VAL1
                ADD     A,X_POS1
                MOV     X_POS1,A
                MOV     A,VAL2
                ADDC    A,X_POS2
                MOV     X_POS2,A
                MOV     A,VAL3
                ADDC    A,X_POS3
                MOV     X_POS3,A
                MOV     A,VAL4
                ADDC    A,X_POS4
                MOV     X_POS4,A

;                       ----------------------------        / 932us

                MOV     A,X_DIFF1
                ADD     A,Z4_1
                MOV     VAL1,A
                MOV     A,X_DIFF2
                ADDC    A,Z4_2
                MOV     VAL2,A
                MOV     A,X_DIFF3
                ADDC    A,Z4_3
                MOV     VAL3,A
                MOV     A,X_DIFF4
                ADDC    A,Z4_4
                MOV     VAL4,A
                MOV     A,X_DIFF5
                ADDC    A,Z4_5
                MOV     CHAR,A

                CALL    ANTI_LOG

                MOV     A,VAL1
                ADD     A,X_POS1
                MOV     VAL1,A
                MOV     A,VAL2
                ADDC    A,X_POS2
                MOV     VAL2,A
                MOV     A,VAL3
                ADDC    A,X_POS3
                MOV     VAL3,A
                MOV     A,VAL4
                ADDC    A,X_POS4
```

```
                        MOV     VAL4,A
;                               ----------------------------    / 1009us
;               AT THIS POINT RECOVER OFSET DECIMAL PT (DIV BY 2^24)
                        CALL    GET_LOG

                        MOV     PSW,#00
                        MOV     A,VAL1
                        SUBB    A,#00
                        MOV     VAL1,A
                        MOV     A,VAL2
                        SUBB    A,#00
                        MOV     VAL2,A
                        MOV     A,VAL3          ;CORRECTION FACTOR
                        SUBB    A,#07
                        MOV     VAL3,A
                        MOV     A,VAL4
                        SUBB    A,#00
                        MOV     VAL4,A
                        MOV     A,CHAR
                        SUBB    A,#16
                        MOV     CHAR,A

                        CALL    ANTI_LOG

                        MOV     X_POS1,VAL1
                        MOV     X_POS2,VAL2
                        MOV     X_POS3,VAL3
                        MOV     X_POS4,VAL4
;                               ----------------------------    / 1133us
;       NEWYPT = Z1 * YPT(LSTPTR,1) + Z2 * YPT(MIDPTR,1) + Z3 * LASTYDIFF + Z4 * YDIFF

                        MOV     DPL,YLST_LPTR
                        MOV     DPH,YLST_HPTR
                        MOVX    A,@DPTR
                        MOV     VAL1,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     VAL2,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     VAL3,A
                        INC     DPTR
                        MOVX    A,@DPTR
                        MOV     VAL4,A

                        CALL    GET_LOG

                        MOV     A,VAL1
                        ADD     A,Z1_1
                        MOV     VAL1,A
                        MOV     A,VAL2
                        ADDC    A,Z1_2
                        MOV     VAL2,A
                        MOV     A,VAL3
                        ADDC    A,Z1_3
                        MOV     VAL3,A
                        MOV     A,VAL4
                        ADDC    A,Z1_4
                        MOV     VAL4,A
                        MOV     A,CHAR
                        ADDC    A,Z1_5
                        MOV     CHAR,A

                        CALL    ANTI_LOG
```

```
                MOV     Y_POS1,VAL1
                MOV     Y_POS2,VAL2
                MOV     Y_POS3,VAL3
                MOV     Y_POS4,VAL4
;                       ------------------------------          / 1278us

                MOV     DPL,YMID_LPTR
                MOV     DPH,YMID_HPTR
                MOVX    A,@DPTR
                MOV     VAL1,A
                INC     DPTR
                MOVX    A,@DPTR
                MOV     VAL2,A
                INC     DPTR
                MOVX    A,@DPTR
                MOV     VAL3,A
                INC     DPTR
                MOVX    A,@DPTR
                MOV     VAL4,A

                CALL    GET_LOG

                MOV     A,VAL1
                ADD     A,Z2_1
                MOV     VAL1,A
                MOV     A,VAL2
                ADDC    A,Z2_2
                MOV     VAL2,A
                MOV     A,VAL3
                ADDC    A,Z2_3
                MOV     VAL3,A
                MOV     A,VAL4
                ADDC    A,Z2_4
                MOV     VAL4,A
                MOV     A,CHAR
                ADDC    A,Z2_5
                MOV     CHAR,A

                CALL    ANTI_LOG

                MOV     A,VAL1
                ADD     A,Y_POS1
                MOV     Y_POS1,A
                MOV     A,VAL2
                ADDC    A,Y_POS2
                MOV     Y_POS2,A
                MOV     A,VAL3
                ADDC    A,Y_POS3
                MOV     Y_POS3,A
                MOV     A,VAL4
                ADDC    A,Y_POS4
                MOV     Y_POS4,A
;                       ------------------------------          / 1427us

                MOV     A,LAST_Y_DIFF1
                ADD     A,Z3_1
                MOV     VAL1,A
                MOV     A,LAST_Y_DIFF2
                ADDC    A,Z3_2
                MOV     VAL2,A
                MOV     A,LAST_Y_DIFF3
                ADDC    A,Z3_3
                MOV     VAL3,A
                MOV     A,LAST_Y_DIFF4
                ADDC    A,Z3_4
                MOV     VAL4,A
                MOV     A,LAST_Y_DIFF5
                ADDC    A,Z3_5
                MOV     CHAR,A
```

```
                CALL    ANTI_LOG

                MOV     A,VAL1
                ADD     A,Y_POS1
                MOV     Y_POS1,A
                MOV     A,VAL2
                ADDC    A,Y_POS2
                MOV     Y_POS2,A
                MOV     A,VAL3
                ADDC    A,Y_POS3
                MOV     Y_POS3,A
                MOV     A,VAL4
                ADDC    A,Y_POS4
                MOV     Y_POS4,A

;                       ----------------------------    / 1504us

                MOV     A,Y_DIFF1
                ADD     A,Z4_1
                MOV     VAL1,A
                MOV     A,Y_DIFF2
                ADDC    A,Z4_2
                MOV     VAL2,A
                MOV     A,Y_DIFF3
                ADDC    A,Z4_3
                MOV     VAL3,A
                MOV     A,Y_DIFF4
                ADDC    A,Z4_4
                MOV     VAL4,A
                MOV     A,Y_DIFF5
                ADDC    A,Z4_5
                MOV     CHAR,A

                CALL    ANTI_LOG

                MOV     A,VAL1
                ADD     A,Y_POS1
                MOV     VAL1,A
                MOV     A,VAL2
                ADDC    A,Y_POS2
                MOV     VAL2,A
                MOV     A,VAL3
                ADDC    A,Y_POS3
                MOV     VAL3,A
                MOV     A,VAL4
                ADDC    A,Y_POS4
                MOV     VAL4,A

;                       ----------------------------    / 1581us
;       AT THIS POINT RECOVER OFSET DECIMAL PT (DIV BY 2^24)

                CALL    GET_LOG

                MOV     PSW,#00
                MOV     A,VAL1
                SUBB    A,#00
                MOV     VAL1,A
                MOV     A,VAL2
                SUBB    A,#00
                MOV     VAL2,A
                MOV     A,VAL3          ;CORRECTION FACTOR
                SUBB    A,#07
                MOV     VAL3,A
                MOV     A,VAL4
                SUBB    A,#00
                MOV     VAL4,A
                MOV     A,CHAR
                SUBB    A,#16
                MOV     CHAR,A

                CALL    ANTI_LOG
```

```
                        MOV     Y_POS1,VAL1
                        MOV     Y_POS2,VAL2
                        MOV     Y_POS3,VAL3
                        MOV     Y_POS4,VAL4
;                           ****************************      / 1705us
;               MOVE TO THE INTERPOLATED .... NEW_X_PT and NEW_Y_PT

                        CALL    LOAD_AND_MOVE

;                           ****************************
;                         To this point = 1705us + PLOT TIME

;                               ----------------------------

                        DJNZ    R4,OK_TO_PLOT
                        DJNZ    R5,OK_TO_PLOT

;                           Increment the X/Y table pointers

                        MOV     A,XMID_LPTR
                        ADD     A,#06
                        MOV     XMID_LPTR,A
                        MOV     A,XMID_HPTR
                        ADDC    A,#00
                        MOV     XMID_HPTR,A

                        MOV     A,YMID_LPTR
                        ADD     A,#06
                        MOV     YMID_LPTR,A
                        MOV     A,YMID_HPTR
                        ADDC    A,#00
                        MOV     YMID_HPTR,A

;                           Check if at the end of the table

                        DJNZ    R2,DO_NXT_SEGMNT
                        DJNZ    R3,DO_NXT_SEGMNT

;                           ****************************
;                         Move to the last GIVEN X and Y point

                        CALL    LOAD_AND_MOVE

;                           ****************************

                        JMP     EXT_SPLINE_ROUTINE

;                           ****************************

OK_TO_PLOT:             JMP     PLOT_NXT_PNT

DO_NXT_SEGMNT:          JMP     NXT_SEGMENT

;                           ****************************
;               SUB THE NEW POS FROM THE OLD POS - GET THE MOVE INCREMENTS
;               CALC THE INTERP VEL TO MAINTAIN CONSTANT SURFACE FEEDRATE
;                     CHECK THE STATUS - LOAD THE X/Y MOVE - POS AND VEL
;                       CHECK THE TRAJ COMP FLG - START THE X/Y MOVE

LOAD_AND_MOVE:          CALL    CALC_XY_VELS

LOAD_X_AXIS:            MOV     AXIS_LPTR,X_ADDR
                        CALL    LOAD_BUFF

                        MOV     AXIS_LPTR,Y_ADDR
                        CALL    LOAD_BUFF

                        MOV     AXIS_LPTR,X_ADDR        ;X AXIS

X_STAT:                 CALL    GET_LM628_STAT
                        JNB     ACC.2,X_STAT            ;Trajectory Complete ?
```

```
                        MOV     AXIS_LPTR,X_AXIS        ;Yes - Start buffered move
                        CALL    START_AXIS

                        MOV     AXIS_LPTR,Y_ADDR        ;X AXIS

Y_STAT:                 CALL    GET_LM628_STAT
                        JNB     ACC.2,Y_STAT            ;Trajectory Complete ?

                        MOV     AXIS_LPTR,Y_AXIS        ;Yes - Start buffered move
                        CALL    START_AXIS
                        RET

;                       **************************
;       Load any axis velocity, and position into its LM628 DSP controller

LOAD_BUFF:              MOV     DPL,AXIS_LPTR           ;Load the Axis lo-ptr address
                        MOV     DPH,#0FFH               ;Load the Axis hi-ptr address

                        MOV     A,#01FH                 ;Load the trajectory command
                        MOVX    @DPTR,A                 ;Send the command

                        CALL    LM628_BUSY              ;Check LM628 busy bit
                        JNC     SET_MODE                ;Carry is set if error
                        SETB    ES                      ;Set serial on if error
                        RET                             ;Error exit

SET_MODE:               INC     DPTR                    ;Base address + 1
                        MOV     A,#000H                 ;BYTE 1 = Pos. Mode
                        MOVX    @DPTR,A                 ;Send the command

                        MOV     A,#00001010B            ;Abs. Pos. and Vel. data
                                                        ;    to be sent
                        MOVX    @DPTR,A                 ;Send the command

                        CALL    LM628_BUSY              ;Check LM628 busy bit
                        JNC     SND_VEL43               ;Carry is set if error
                        SETB    ES                      ;Set serial on if error
                        RET                             ;Error exit

SND_VEL43:              MOV     A,STR14
                        MOVX    @DPTR,A                 ;SEND BYTE 4
                        MOV     A,STR13
                        MOVX    @DPTR,A                 ;SEND BYTE 3

                        CALL    LM628_BUSY              ;Check LM628 busy bit
                        JNC     SND_VEL21               ;Carry is set if error
                        SETB    ES                      ;Set serial on if error
                        RET                             ;Error exit

SND_VEL21:              MOV     A,STR12
                        MOVX    @DPTR,A                 ;SEND BYTE 2
                        MOV     A,STR11
                        MOVX    @DPTR,A                 ;SEND BYTE 1

                        CALL    LM628_BUSY              ;Check LM628 busy bit
                        JNC     SND_POS43               ;Carry is set if error
                        SETB    ES                      ;Set serial on if error
                        RET                             ;Error exit

SND_POS43:              MOV     A,VAL4
                        MOVX    @DPTR,A                 ;SEND BYTE 4
                        MOV     A,VAL3
                        MOVX    @DPTR,A                 ;SEND BYTE 3

                        CALL    LM628_BUSY              ;Check LM628 busy bit
                        JNC     SND_POS21               ;Carry is set if error
                        SETB    ES                      ;Set serial on if error
                        RET                             ;Error exit

SND_POS21:              MOV     A,VAL2
                        MOVX    @DPTR,A                 ;SEND BYTE 2
                        MOV     A,VAL1
```

```
                        MOVX    @DPTR,A             ;SEND BYTE 1

                        MOV     X_LPOS1,X_POS1      ;SAVE CURRENT POS
                        MOV     X_LPOS2,X_POS2      ;       IN LAST POS REG
                        MOV     X_LPOS3,X_POS3
                        MOV     X_LPOS4,X_POS4

                        MOV     Y_LPOS1,Y_POS1
                        MOV     Y_LPOS2,Y_POS2
                        MOV     Y_LPOS3,Y_POS3
                        MOV     Y_LPOS4,Y_POS4
                        RET

START_AXIS:             MOV     DPL,AXIS_LPTR       ;Load the Axis lo-ptr address
                        MOV     DPH,#0FFH           ;Load the Axis hi-ptr address

                        CALL    LM628_BUSY          ;Chk the busy bit flag
                        JNC     START_AXS           ;Jump if no error
                        SETB    ES                  ;Else - turn "on" Serial intp
                        POP     ACC                 ;Clear this call return
                        POP     ACC                 ;       address
                        JMP     ERROR_HANDLER       ;Jump to error routine

START_AXS:              MOV     A,#001H             ;Start the axis move
                        MOVX    @DPTR,A             ;Send the command
                        RET                         ;Rtn to the calling routine

;                           ****************************
;                         Check to see if the LM628 busy bit is = 0

LM628_BUSY:             PUSH    DPL                 ;Save DPL address value
                        PUSH    ACC                 ;Save the Acc reg
                        MOV     R0,#0FFH            ;load timeout for error chk
                        MOV     A,DPL               ;Mask off bit 1 to read the
                        ANL     A,#0FEH             ;       LM628 status register
                        MOV     DPL,A

CHK_BUSY:               MOVX    A,@DPTR             ;Get the LM628 status
                        ANL     A,#00000001B        ;Check busy bit
                        JZ      NOT_BUSY            ;If clear, exit
                        DJNZ    R0,CHK_BUSY         ;Loop untill timeout or not busy

                        SETB    C                   ;Set timeout error
                        JMP     BUSY_EXT            ;Error exit

NOT_BUSY:               CLR     C                   ;No error exit

BUSY_EXT:               POP     ACC                 ;Restore the Acc reg
                        POP     DPL                 ;Restore DPL address value
                        RET                         ;Return to calling routine

;                           ************************

GET_LM628_STAT:         MOV     DPL,AXIS_LPTR       ;Load the Axis low ptr
                        MOV     DPH,#0FFH           ;Load the Axis hi ptr
                        MOVX    A,@DPTR             ;Get the LM628 status
                        RET                         ;Return to calling routine

;                           ************************
;  Calculate the new X/Y vector velocities.  Use registers VEL1/4  X_VEL1/4  Y_VEL1/4  Y_INC1/4 for values.

CALC_XY_VELS:           MOV     PSW,#00
                        MOV     A,Y_POS1
                        SUBB    A,Y_LPOS1
                        MOV     VAL1,A

                        MOV     A,Y_POS2
                        SUBB    A,Y_LPOS2
                        MOV     VAL2,A
```

```
                MOV     A,Y_POS3
                SUBB    A,Y_LPOS3
                MOV     VAL3,A

                MOV     A,Y_POS4
                SUBB    A,Y_LPOS4
                MOV     VAL4,A

                MOV     A,VAL4          ;Get Y increment
                JNB     ACC.7,STRYINC   ;Jump if Yinc = +

                MOV     PSW,#0          ;If Yinc = -
                MOV     A,#0            ;Convert to a + value
                SUBB    A,VAL1
                MOV     VAL1,A
                MOV     A,#0
                SUBB    A,VAL2
                MOV     VAL2,A
                MOV     A,#0
                SUBB    A,VAL4
                MOV     VAL4,A
                MOV     A,#0
                SUBB    A,VAL4
                MOV     VAL4,A

;                       ------------------------

STRYINC:        MOV     Y_INC1,VAL1
                MOV     Y_INC2,VAL2
                MOV     Y_INC3,VAL3
                MOV     Y_INC4,VAL4

                CALL    GET_LOG         ;Get LOG of Yinc
                CALL    STORE_VAL       ;Put VAL1-4 -> STORE1-4
                CALL    AD_TO_STR1      ;Get (Yinc)^2 -> STORE1-4

;                       ------------------------

                MOV     PSW,#00
                MOV     A,X_POS1
                SUBB    A,X_LPOS1
                MOV     VAL1,A
                MOV     A,X_POS2
                SUBB    A,X_LPOS2
                MOV     VAL2,A
                MOV     A,X_POS3
                SUBB    A,X_LPOS3
                MOV     VAL3,A
                MOV     A,X_POS4
                SUBB    A,X_LPOS4
                MOV     VAL4,A

                MOV     A,VAL4          ;Get X increment
                JNB     ACC.7,STRXINC   ;Jump if Xinc = +

                MOV     PSW,#0          ;If Xinc = -
                MOV     A,#0            ;Convert to a + value
                SUBB    A,VAL1
                MOV     VAL1,A
                MOV     A,#0
                SUBB    A,VAL2
                MOV     VAL2,A
                MOV     A,#0
                SUBB    A,VAL4
                MOV     VAL4,A
                MOV     A,#0
                SUBB    A,VAL4
                MOV     VAL4,A

STRXINC:        MOV     X_INC1,VAL1
                MOV     X_INC2,VAL2
                MOV     X_INC3,VAL3
```

```
                MOV     X_INC4,VAL4

                CALL    GET_LOG             ;Get LOG of Xinc
;                       ------------------------

                MOV     A,VAL1              ;Get (Xinc)^2 -> VAL1-4
                ADD     A,VAL1
                MOV     VAL1,A

                MOV     A,VAL2
                ADDC    A,VAL2
                MOV     VAL2,A

                MOV     A,VAL3
                ADDC    A,VAL3
                MOV     VAL3,A

                MOV     A,VAL4
                ADDC    A,VAL4
                MOV     VAL4,A

                MOV     A,CHAR
                ADDC    A,CHAR
                MOV     CHAR,A
;                       ------------------------

                MOV     A,VAL1              ;Get (Xinc ^2 + Yinc^2)
                ADD     A,STR11
                MOV     VAL1,A

                MOV     A,VAL2
                ADDC    A,STR12
                MOV     VAL2,A

                MOV     A,VAL3
                ADDC    A,STR13
                MOV     VAL3,A

                MOV     A,VAL4
                ADDC    A,STR14
                MOV     VAL4,A

                MOV     A,CHAR
                ADDC    A,SCHAR
                MOV     CHAR,A
;                       ------------------------

                CALL    LOG_D2              ;Get SQRT(Xinc ^2 + Yinc^2)
                CALL    STORE_VAL           ;Store the radius of the
                                            ;Increm triangle -> STORE1-4
;                       ------------------------

                MOV     VAL1,VEL1           ;Get the Vel. requirement
                MOV     VAL2,VEL2
                MOV     VAL3,VEL3
                MOV     VAL4,VEL4

                CALL    GET_LOG             ;Get LOG of the velocity
;                       ------------------------

                MOV     PSW,#000H           ;Get VEL/RAD -> STORE1-4
                MOV     A,VAL1              ; = Log(vel) - Log(Rad)
                SUBB    A,STR11
                MOV     STR11,A

                MOV     A,VAL2
                SUBB    A,STR12
                MOV     STR12,A
```

```
                MOV     A,VAL3
                SUBB    A,STR13
                MOV     STR13,A

                MOV     A,VAL4
                SUBB    A,STR14
                MOV     STR14,A

                MOV     A,CHAR
                SUBB    A,SCHAR
                MOV     SCHAR,A

;                       ------------------------

                MOV     VAL1,X_INC1         ;Get the Xinc
                MOV     VAL2,X_INC2
                MOV     VAL3,X_INC3
                MOV     VAL4,X_INC4

                CALL    GET_LOG             ;Get LOG of the Xinc
                CALL    VR_xINC             ;Mult Vel.Ratio by XInc.
                CALL    ANTI_LOG            ;Get the X Velocity

                MOV     X_VEL1,VAL1         ;Store the X velocity
                MOV     X_VEL2,VAL2
                MOV     X_VEL3,VAL3
                MOV     X_VEL4,VAL4

;                       ------------------------

                MOV     VAL1,Y_INC1         ;Get the Yinc
                MOV     VAL2,Y_INC2
                MOV     VAL3,Y_INC3
                MOV     VAL4,Y_INC4

                CALL    GET_LOG             ;Get LOG of the Yinc
                CALL    VR_xINC             ;Mult Vel.Ratio by YInc.
                CALL    ANTI_LOG            ;Get the Y Velocity

                MOV     Y_VEL1,VAL1         ;Store the Y velocity
                MOV     Y_VEL2,VAL2
                MOV     Y_VEL3,VAL3
                MOV     Y_VEL4,VAL4
                RET

;                  ****************************

CONVERT_ANS:    MOV     PSW,#00
                MOV     A,#00
                SUBB    A,@R0
                MOV     @R0,A

                INC     R0
                MOV     A,#00
                SUBB    A,@R0
                MOV     @R0,A

                INC     R0
                MOV     A,#00
                SUBB    A,@R0
                MOV     @R0,A

                INC     R0
                MOV                         A,#00
                SUBB    A,@R0
                MOV     @R0,A
                RET

;                  ****************************
```

```
DIV_BY_2:       CLR     C
                MOV     A,@R0
                RRC     A
                MOV     @R0,A
                MOV     VAL1,A

                DEC     R0

                MOV     A,@R0
                RRC     A
                MOV     @R0,A
                MOV     VAL1,A

                DEC     R0

                MOV     A,@R0
                RRC     A
                MOV     @R0,A
                MOV     VAL1,A

                DEC     R0

                MOV     A,@R0
                RRC     A
                MOV     @R0,A
                MOV     VAL1,A
                RET
;                       ******************************
STORE_VAL:      MOV     STR11,VAL1
                MOV     STR12,VAL2
                MOV     STR13,VAL3
                MOV     STR14,VAL4
                MOV     SCHAR,CHAR
                RET
;                       ******************************
AD_TO_STR1:     MOV     A,VAL1
                ADD     A,STR11
                MOV     STR11,A

                MOV     A,VAL2
                ADDC    A,STR12
                MOV     STR12,A

                MOV     A,VAL3
                ADDC    A,STR13
                MOV     STR13,A

                MOV     A,VAL4
                ADDC    A,STR14
                MOV     STR14,A

                MOV     A,SCHAR
                ADDC    A,CHAR
                MOV     SCHAR,A
                RET
;                       ******************************
;               Divide the logrithm by 2 to get the squareroot of the number
LOG_D2:         CLR     C
                MOV     A,CHAR
                RRC     A
                MOV     CHAR,A

                MOV     A,VAL4
                RRC     A
                MOV     VAL4,A
```

```
                    MOV     A,VAL3
                    RRC     A
                    MOV     VAL3,A

                    MOV     A,VAL2
                    RRC     A
                    MOV     VAL2,A

                    MOV     A,VAL1
                    RRC     A
                    MOV     VAL1,A
                    RET

;                           ******************************
;       Add the VAL1-4 regs to the STORE1-4 regs.  Put the answer in the VAL1-4 regs.

VR_xINC:            MOV     A,VAL1
                    ADD     A,STR11
                    MOV     VAL1,A

                    MOV     A,VAL2
                    ADDC    A,STR12
                    MOV     VAL2,A

                    MOV     A,VAL3
                    ADDC    A,STR13
                    MOV     VAL3,A

                    MOV     A,VAL4
                    ADDC    A,STR14
                    MOV     VAL4,A

                    MOV     A,CHAR
                    ADDC    A,SCHAR
                    MOV     CHAR,A
                    RET

;                           **************************

GET_LOG:            ;The GET_LOG  routine is listed Appendix F
                    RET

ANTI_LOG:           ;The ANTI_LOG routine is listed Appendix F
                    RET

;                           **************************

                    END
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28