Bosch CC100 Control

WADKIN
PARAMETRIC PROGRAMMING
MANUAL

CONTENTS

INTRODUCTION

Ever since you started programming the Bosch CC 100M control on your Wadkin router, you have been using Parametric Programmes. You probably didn't know it at the time! But if you remember using VX & VY values (and you should) then you may have thought these were part of normal N.C. programmes. Well, in fact, they are used as INPUT VARIABLES for the G820 offset cycle [more about that in Section 4].

The key word in the last sentence was VARIABLES (also known as PARAMETERS - hence PARAMETRIC PROGRAMMING). These variables can be used in a number of situations for a number of reasons:- calculating intersection points on shapes, variable depths, feed-rates etc, transfer of information to and from tool and zero shift tables and so on.

The way we use these variables for various applications is explained, in detail, in this manual. We hope you will benefit from the information (which accompanies the Wadkin Parametric Programming Course).

Once you have been on this course we trust that you will have sufficient knowledge and information to program your Wadkin CNC router and make it even more cost-effective than before.

GUIDANCE FOR USE OF THIS MANUAL

Due to the nature and flexibility of parametric programmes, the following points should be noted:-

1. The examples used are rather fictitious and are created as such to explain particular functions.

2. Examples should be followed through logically and with attention to detail.

3. The examples are NOT written to run on a specific machine. Hence, some of the codes required to run on your own machine may not be present for reasons of clarity. It is expected that you will be competent enough with the standard N.C. programming to know where these codes have been omitted.

4. Feedrates and depths of cut (unless otherwise stated) are arbitrary and do not relate to any particular product/material.

5. The VX and VY figures used are also arbitrary.

6. This manual is used as a basis for the 'Wadkin Parametric Programming Course' and not as a substitute for the course.

7. Please note that none of the drawings are to scale. They are only used as diagrammatic representations.

8. Although there are 125 variables available (V1-V99 and VA-VZ) we recommend that you use only V30 - V80 for your programmes. This is because V1 - V30, V81 - V99 and VA - VZ can, in some instances, be used by either Bosch or Wadkin internal cycles and could therefore overwrite values you have used.

1. BASIC PARAMETRIC PROGRAMMES.

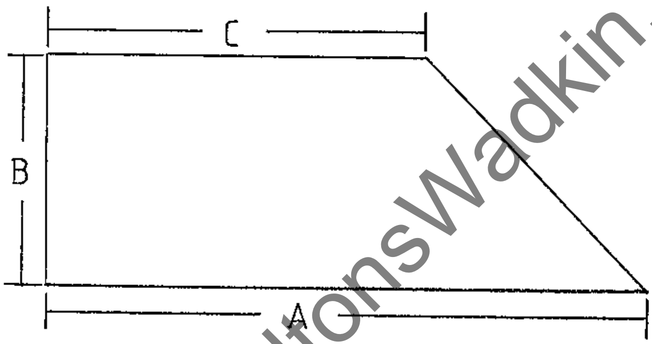
In Appendix 1 (BASIC TRIGONOMETRY AND ALGEBRA EXAMPLES) we find simple examples of problems which have been solved using trigonometry and algebra. In engineering and woodworking it is often necessary to perform calculations to determine dimensions for components which are to be machined on computer controlled machinery.

Parts of a similar shape, but different sizes, occur frequently. In these cases the aforementioned calculations become tedious as they have to be repeated, but with different dimensions, for every component.

This is where we see at least one of the benefits of PARAMETRIC PROGRAMMES. We can create programmes, but instead of using real numbers, we can use variables.

Consider example 1.

Example 1



We have to manufacture 3 components of the shape shown above, but with differing sizes:-

	A	B	C
COMPONENT 1.	50	25	40
COMPONENT 2.	90	30	60
COMPONENT 3.	45	18	32

Consider the following programmes using the normal CNC format to produce the components.

	COMPONENT 1	COMPONENT 2	COMPONENT 3
N1	G90	G90	G90
N2	G53	G53	G53
N3	G0 Z0 T00	G0 Z0 T00	G0 Z0 T00
N4	VX=200 VY=200	VX=200 VY=200	VX=200 VY=200
N5	G820	G820	G820
N6	G0 X-20 Y-20 Z5 T01	G0 X-20 Y-20 Z5 T01	G0 X-20 Y-20 Z5 T0
N7	G42 T01	G42 T01	G42 T01
N8	G1 X-10 Y0 F6000	G1 X-10 Y0 F6000	G1 X-10 Y0 F6000
N9	G1 Z-10 F1500	G1 Z-10 F1500	G1 Z-10 F1500
*N10	G1 X50 F4000	G1 X90 F4000	G1 X45 F4000
*N11	X40 Y25	X60 Y30	X32 Y18
N12	X0	X0	X0
N13	Y-10	Y-10	Y-10
N14	G0 Z5	G0 Z5	G0 Z5
N15	G40 X-20 Y-20	G40 X-20 Y-20	G40 X-20 Y-20
N16	G53	G53	G53
N17	G0 X650 Y800 Z0 T00	G0 X650 Y800 Z0 T00	G0 X650 Y800 Z0 T00
N18	M30	M30	M30

Here we have had to produce 3 programmes which are virtually identical. The only differences are in lines N10 and N11.

A way to overcome this is by using parametric programming. Consider the following:-

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200
N5     G820
N+5    V1=?       V2=?      V3=?
N6     G0   X-20 Y-20 Z5 T01
N7     G42 T01
N8     G1   X-10 Y0   F6000
N9     G1   Z-10 F1500
N10    G1   F4000
N+10        X=V1
N11         X=V3  Y=V2
N12         X0
N13               Y-10
N14    G0   Z5
N15    G40 X-20 Y-20
N16    G53
N17    G0   X650 Y800 Z0 T00
N18    M30
```

This programme has one fundamental difference to the previous 'N.C.' programmes.

Line N+5 contains variables V1, V2 and V3. Into these variable stores we can 'load' a numerical value. This value can relate to the sizes referred to on the component drawing, ie. for component 1, V1=50, V2=25 and V3=40.

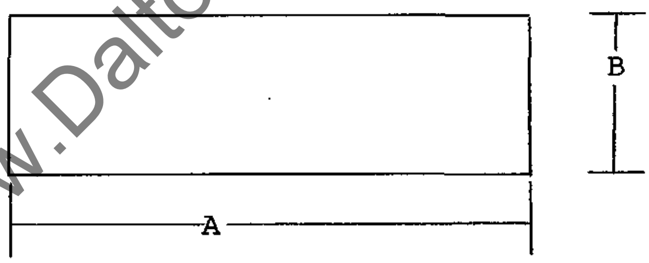
When the programme is running, as it gets to line N+5, the variable stores V1, V2 and V3 will be 'loaded' with the values 50, 25 and 40 respectively. As it gets to line N+10, the X axis will move to the current numerical value of V1 - hence the axis will travel to X50. The same happens in line N11 - the X and Y axes will travel to 40 and 25 respectively Therefore, running the programme completely will produce component 1.

Now, if we change the values of V1, V2 and V3 to, say 90, 30 and 60 respectively, and we were to run the programme again, the component produced would be as per component 2.

So we can see that by simply changing the parameter values we get the same shape, but with different sizes.

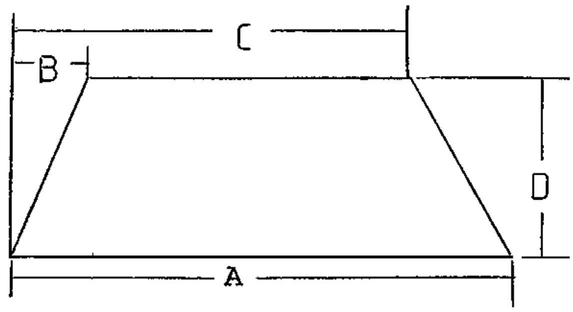
Try to write a parametric programme (similar to the previous example) for the following shapes. [Suggested answers are in Appendix 3.].

Problem 1



We need to produce 4 components as per the shape above. These are the sizes:-

	A	B
COMPONENT 1	120	40
COMPONENT 2	180	80
COMPONENT 3	100	50
COMPONENT 4	90	60



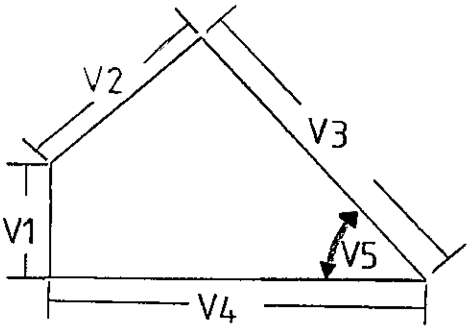
Here, we need to produce 3 components. These are the sizes:-

	A	B	C	D
COMPONENT 1	100	25	75	50
COMPONENT 2	200	50	150	100
COMPONENT 3	300	75	225	150

2. MORE COMPLEX EXAMPLES

Let us now take this concept one step further. Consider the following example.

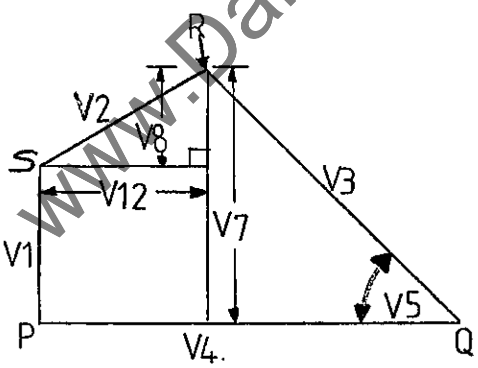
Example 2



The object, here, is to formulate a programme to machine the above shape of component. Notice, though , that we do not have fixed sizes, we have variables.

As we use 'CARTESIAN' coordinates - that is intersection points are determined using an X and Y 'grid' - we need to find all the intersection points with respect to a datum point. Let us use the bottom left hand corner in this case.

To determine all the intersection points for this shape, we need to do some arithmetic and trigonometry. Let us split up the shape into RIGHT ANGLED TRIANGLES.



Points P, Q and S are easy to determine in terms of distance from the datum point. Point R, however needs to be calculated.

To determine V7:- [* = multiplication sign on CNC control]

SIN V5 = $\frac{V7}{V3}$ therefore V7 = V3 * SIN V5

To perform this calculation using the CNC control, we have to break it down into single steps; ie:-

```
V6 = SIN V5
V7 = V3 * V6
```

[<u>NOTE</u>:- We cannot enter V7 = V3 * SIN V5 on one line as the control does not allow this]

To determine V8:-

```
V8 = V7 - V1
```

To determine V12:-

```
 V9 =  V8 * V8
V10 =  V2 * V2
V11 = V10 - V9
V12 = SQR V11
```

We now have the X and Y co-ordinates of point R. So, all the points can be shown thus:-

	X	Y
POINT P	0	0
POINT Q	V4	0
POINT R	V12	V7
POINT S	0	V1

Let us now write the complete programme incorporating the calculations we have just performed.

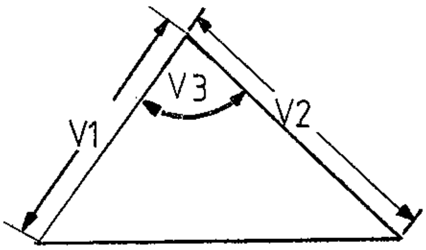
```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200
N5     G820
N6     V1=?    V2=?    V3=?    V4=?    V5=?    V13=0
N7     G22 P1
N8     G0  X-20 Y-20 Z5 T01
N9     G42 T01
N10    G1  X-10 Y0   F6000
N11    G1  Z-10 F1500
N12    G1  F4000
N13        X=V4
N14        X=V12  Y=V7
N15        X=V13  Y=V1
N16               Y-10
N17    G0  Z5
N18    G40 X-20 Y-20
N19    G53
N20    G0  X650 Y800 Z0 T00
N21    M30
N22    $1
N23    V6 = SIN V5
N24    V7 = V3 * V6
N25    V8 = V7 - V1
N26    V9 = V8 * V8
N27    V10 = V2 *V2
V28    V11 = V10 - V9
V29    V12 = SQR V11
V30    G99
```

<u>NOTE 1</u> - Notice that we have set the value of V13 to 0. In line N15 we have programmed the machine to go to X=V13 Y=V1. The control does not allow us to put the line in thus:- X0 Y=V1, as this would combine normal NC information and parametric information.

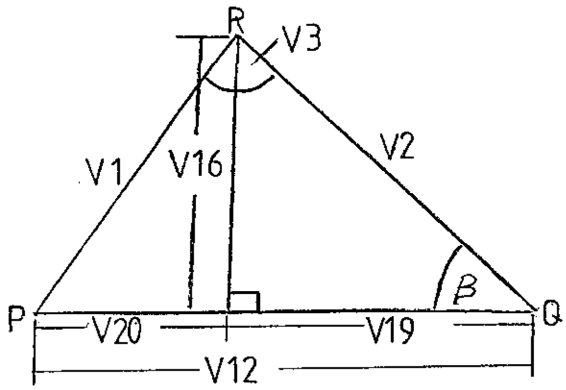
<u>NOTE 2</u> - The use of the sub-programme, $1, is purely for clarity and is not a necessity of a parametric programme.

Let us consider a further example.

<u>Example 3</u>



Again, from the above shape we have to calculate the X and Y co-ordinates.



To determine V12:-

Using the formula:-

$$A^2 = B^2 + C^2 - 2BC \cos A$$

V4 = V1 * V1
V5 = V2 * V2
V6 = COS V3
V7 = V1 * V2
V8 = V7 * 2
V9 = V8 * V6
V10 = V4 + V5
V11 = V10 - V9
V12 = SQR V11

Now, we need to determine Sin ß:-

Using the formula:-

$$\frac{a}{\text{Sin A}} = \frac{b}{\text{Sin B}}$$

$$\text{Sin } \beta = \frac{\text{V1 * SIN V3}}{\text{V12}}$$

Therefore, V13 = SIN V3
V14 = V1 * V12
V15 = V14/V12 (Sinß).

To determine V16:-

V16 = V2 * V15

[Because:- $\text{Sin } \beta = \frac{\text{V16}}{\text{V2}}$ therefore, V16 = V2 * Sin ß]
= V2 * V15

To determine V19:-

$V19 = \sqrt{V5 - V16^2}$ therefore, V17 = V16 * V16
V18 = V5 - V17
V19 = SQR V18

To determine V19:-

V20 = V12 - V19

This now provides us with all the information necessary:-

	X	Y
POINT P	0	0
POINT Q	V12	0
POINT R	V20	V16

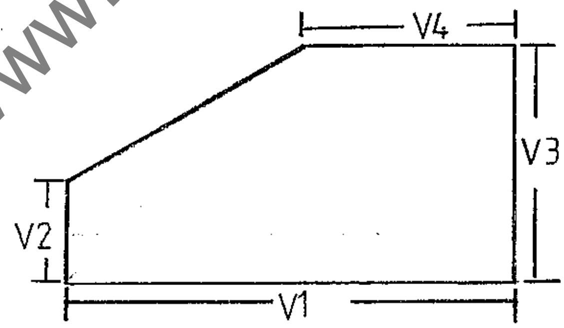
This information could now be incorporated in a programme as we have done before.

You will see from the two previous examples that a considerable amount of thought needs to be given to the initial layout of information and it seems tedious to go through this procedure.

Don't forget, though, that once this programme has been written, all that is necessary to change the size of the component is an alteration to the original 'INPUT' variables. Remember, if we didn't have this facility, we would need to perform these calculations EVERY time we changed the shape. This, indeed, could be a lot more tedious!

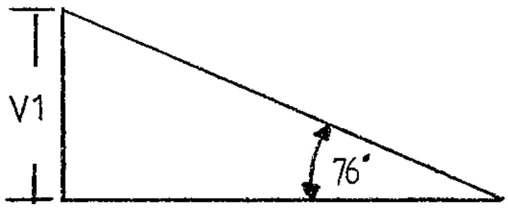
Again here are some problems for you to try.
[The answers are in Appendix 3.]

PROBLEM 3



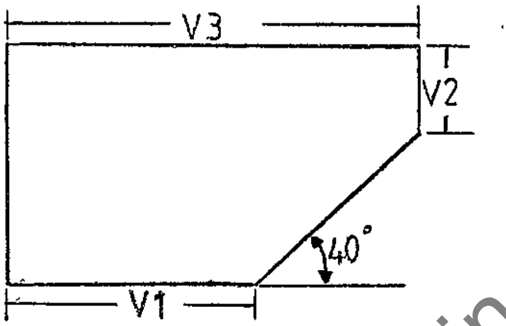
Write a programme for the above component. The input variables are V1, V2, V3 and V4

PROBLEM 4



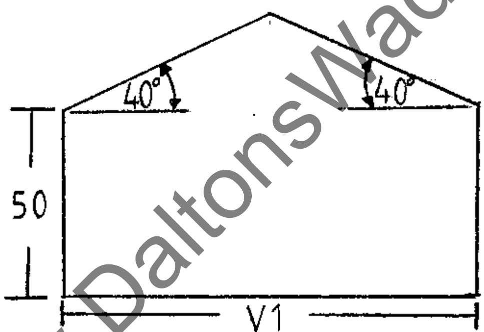
Write a programme to machine this component. Input variable is V1; Angle is always 76 degrees.

PROBLEM 5



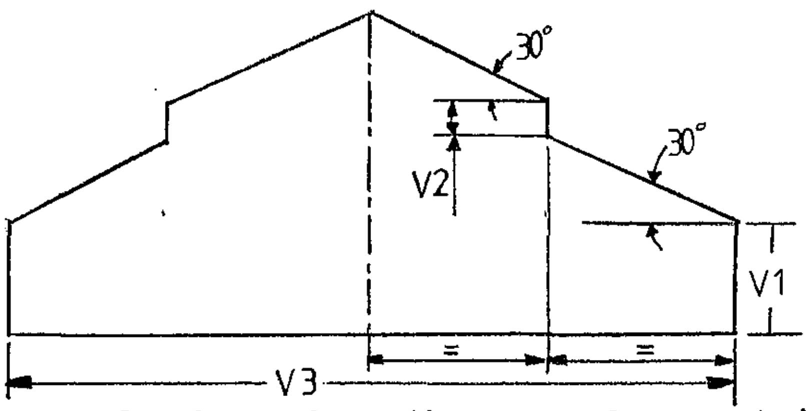
Write a programme for the above component. Again input variables are V1, V2 and V3.

PROBLEM 6



The input variable is V1.

PROBLEM 7



The previous examples have shown the uses of parametric programmes solely in the areas of arithmetic and trigonometry. Let us now look at some more features of parametric programmes.

3. PARAMETRIC PROGRAMMES INCLUDING JUMPS, COUNTS ETC.

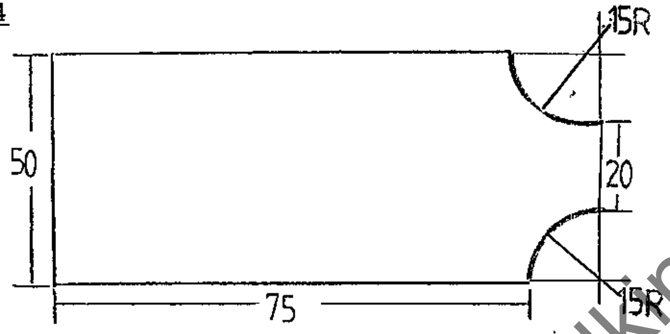
Consider the following situation:-

You have 3 components to manufacture, but of differing thickness - say 15, 20 and 25 mm.

Conventionally, using normal NC programmes we would have to write three different programmes to machine the components though only the Z axis commands would be different.

Compare the following two programmes which have been written for this shape:-

Example 4



THREE COMPONENTS TO BE MADE OF 15, 20 AND 25 MM THICK MATERIAL.

PROGRAMME A

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200
N5    G820
N6    G0  X-20 Y-20 Z5 T01
N7    G42 T01
N8    G1  X-10 Y0   F6000
N9    G1  Z-16 F1500
N10   G1  X75  F4000
N11   G2  X90  Y15  R-15
N12   G1       Y35
N13   G2  X75  Y50  R-15
N14   G1  X0
N15            Y-10
N16   G0  Z5
N17   G40 X-20 Y-20
N18   G53
N19   G0  X650 Y800 Z0 T00
N20   M30
```

PROGRAMME B

```
N1    G90
N2    G53
N3    G0  Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=15
N7    G22 P1
N8    G0  X-20 Y-20 Z5 T01
N9    G42 T01
N10   G1  X-10 Y0   F6000
N11   G1  F1500
N12   Z=VZ
N13   G1  X75  F4000
N14   G2  X90  Y15  R-15
N15   G1       Y35
N16   G2  X75  Y50  R-15
N17   G1  X0
N18            Y-10
N19   G0  Z5
N20   G40 X-20 Y-20
N21   G53
N22   G0  X650 Y800 Z0 T00
N23   M30
N24   $1
N25   V1=V1+1  VZ=V1* -1
N26   G99
```

Note that in Programme A we have in line N9 'Z-16. This tells the machine to go 16mm into the workpiece - 1mm below the bottom of a board which is 15mm thick. To produce the same component out of a board which is 20mm thick the programme needs to be rewritten or modified.

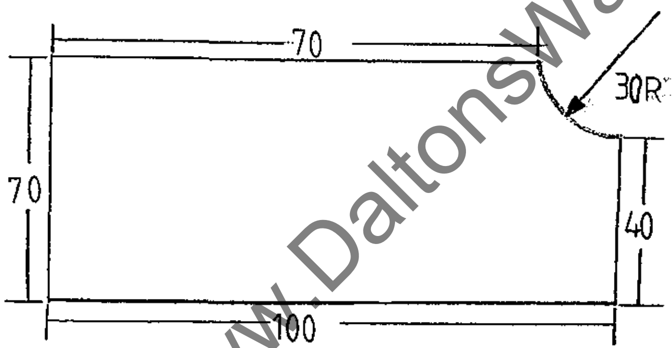
In Programme B we have in line N6, V1=15. This is the thickness of the board. Immediately after this, in line N7, we call up sub-programme $1. This adds 1mm to the thickness of the board and multiplies this number by '-1' to make it a negative number. In line N12 'Z=VZ' sends the Z axis to the current value of VZ (this being -16 in this case). Now, to produce the same component out of a board which is 20mm or 25mm thick we just need to alter V1 at the start of the programme and run it again.

This may not have very much appeal as a practical example because it is very easy in Programme A to modify Z-16 to Z-21, say. Just as easy, in fact, as modifying V1=15 to V1=20. Not much of a benefit it seems.

Let us consider a further example to try and incorporate parametrics to better effect.

<u>EXAMPLE 9</u>

THIS COMPONENT IS VARIABLE THICKNESS BUT NEEDS TO BE CUT IN 3 PASSES.



```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=  (THICKNESS)
N7    G22 P1
N8    $2
N9    G0  X-20 Y-20 Z5 T01
N10   G42 T01
N11   G1  X-10 Y0   F6000
N12   G1  F1500
N13       Z=VZ
N14   G1  X100 F4000
N15            Y40
N16   G2  X70  Y70  R-30
N17   G1  X0
N18            Y-10
N19   G0  Z5
N20   G40 X-20 Y-20
N21   DEC VC
N22   BEQ P99
N23   VZ = VZ - V3
N24   G24 P2
N25   $99
N26   G53
N27   G0  X450 Y600 Z0 T00
N28   M30
N29   $1
N30   V2=V1+1
N31   V3=V2/3
N32   VZ=V3* -1
N33   VC=3
N34   G99
```


<u>BRANCHING EXPLANATIONS</u>:-

```
BEQ   IF CONDITION REGISTER   = 0   JUMP TO ?
BLT   IF CONDITION REGISTER   < 0   JUMP TO ?
BGT   IF CONDITION REGISTER   > 0   JUMP TO ?
BLE   IF CONDITION REGISTER   <=0   JUMP TO ?
BGE   IF CONDITION REGISTER   >=0   JUMP TO ?
BNE   IF CONDITION REGISTER   ≠0    JUMP TO ?
```

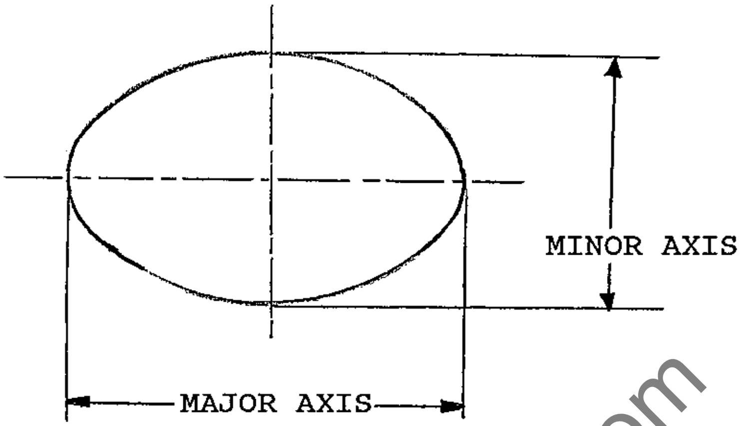
Also,

DEC VC, means DECREMENT VC by the value of 1 - disregard any digits after the decimal point.

Let us try, now, to incorporate a more practical example.

Example 10

AN ELLIPSE



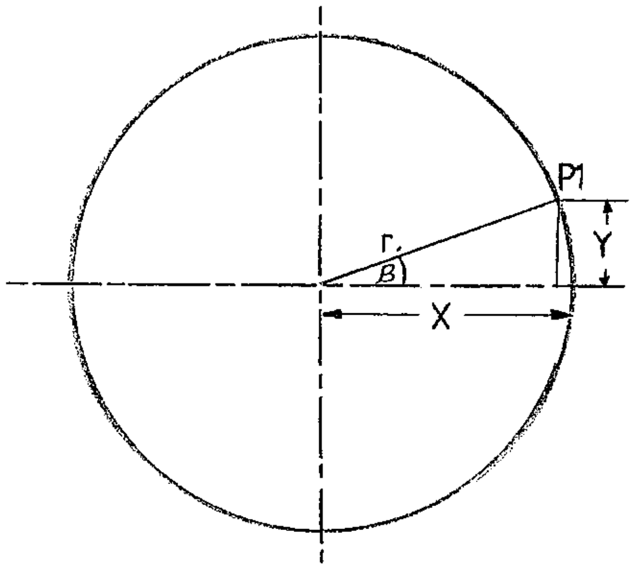
This is quite a common shape in many industries and to produce a 'true' ellipse often causes problems.

Using parametric programming we can overcome this problem 'relatively' easily.

Consider point P1. Using the centre of the shape as the component datum we can calculate the values for X and Y using angle ß.

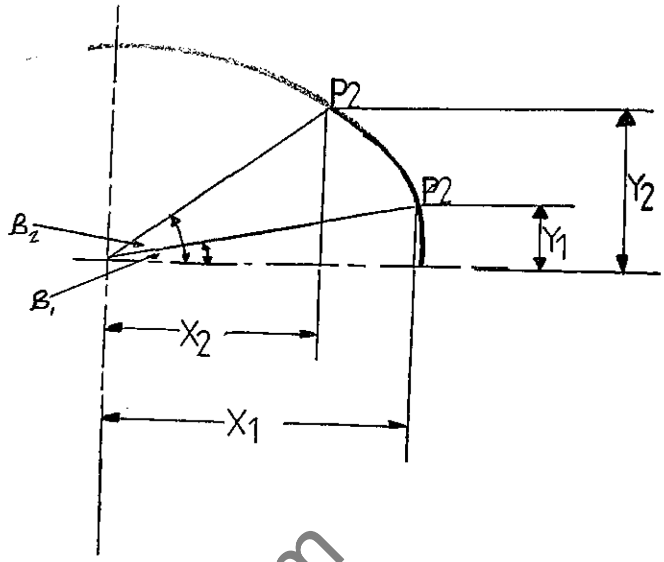
Imagine that the major and minor axes were equal - this infact, would be a circle. To calculate any point on the circumference of a circle we can use the following formulae:-

$$\left.\begin{aligned} x &= r.\cos \beta \\ y &= r.\sin \beta \end{aligned}\right\} P1$$



Now, this theory can be expanded to calculate points around an ellipse.

EXPLANATION OF FORMULAE FOR CALCULATION OF POINTS.



Using ½ the major axis as the radius, r, we can use the above formulae. When the Y dimension has been calculated we can then multiply this by the following ratio:-

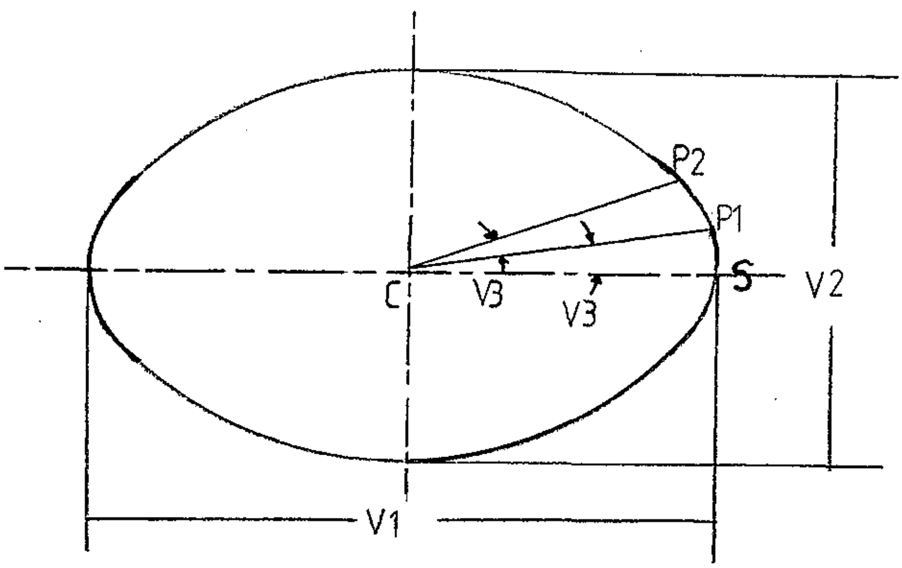
$$\frac{\text{½ minor axis}}{\text{½ major axis}}$$

This multiplication will then provide us with a reduced Y axis dimension.

If we increase the angle to ß2, and perform these calculations again we would have found another point P2. This could be continued to find many points around the perimeter of the ellipse.

So, let us try to adopt this theory and put it into practice on the CC 100M control via parametric programming.

We need to start off with some 'Input' variables. These need to be major axis (V1), minor axis (V2) and incrementing angle (V3).



First of all the major axis (V1) and minor axis (V2) both need to be halved:-

V4 = V1/2
V5 = V2/2

Now, to calculate the ratio, we need to divide ½ minor axis by ½ major axis.

V6 = V5/V4

Let us start from point, S, and calculate the X and Y co-ordinates.

Remember, X = r. cos ß

In this case the starting angle, ß, is 0 (zero) so we must state this now;

V7=0

[Note - This could be stated in the input parameters if required.]

So, to perform the calculation we must find the cosine of V7 and multiply it by ½ the major axis:-

V8 = COS V7
V9 = V8 * V4

We must now follow a similar method for the Y axis co-ordinate.

Again, Y = r. sin ß

So, V10 = SIN V7
V11 = V10 * V4

As we stated earlier, this must then be multiplied by a ratio to produce a reduced 'Y' axis dimension.

V12 = V11 * V6

So, we have now created an X and Y co-ordinate for point S.

Now the angle must be incremented and we must go through the whole procedure again to calculate point P1.

To calculate the new angle we must add the incrementing angle (stated in V3 above) to the angle for which we produced the last X and Y co-ordinates (in this case 0 (zero)).

V7 = V7 + V3

This now overwrites the existing value of V7 with the new angle. We can now repeat the above calculations to find a new set of X and Y co-ordinates for point P1.

Before we do this, though, we must check to see if we have got to the end of the ellipse.

To produce a full ellipse we must travel through 360°. So, in the same way we stated 0° as our start angle, V7, 360° can be stated as our finish angle.

V13 = 360

We must now compare the final angle value, V13, to the current angle value, V7. This can be done in a number of ways but we will use the following method.

V14 = V13 - V7

This line subtracts the current angle value from the final angle value. If the result of this calculation is greater than zero, ie. if the current angle value is greater than the final angle value, then the ellipse must be complete; if not, then we need to move to the point determined by the new incremented angle (current angle). Therefore the following line can be used in our programme:-

BGE P?

This should produce a jump back to the start of the calculations. If the result of the calculation is greater or equal to zero. The lines after this one will not be effected if this condition is met.

The programme will now repeat itself until it has produced an ellipse.

Let us combine all these lines, now, and write the programme.

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=400 VY=300
N5    G820
N6    V1=300  V2=180  V3=2
N7    G0  X170 Y-40 Z5 T01
N8    G42 T01
N9    G1  X150 Y-20 F6000
N10   G1  Z-10 F1500
N11   G1       Y0   F4000
N12   G22 P1
N13   G1       Y20
N14   G0  Z5
N15   G40 X170 Y40
N16   G53
N17   G0  X450 Y600 Z0 T00
N18   M30
N19   $1
N20   V4=V1/2  V5=V2/2
N21   V6=V5/V4
N22   V7=0  V13=360
N23   $2
N24   V8=COS V7
N25   V9=V8*V4
N26   V10=SIN V7
N27   V11=V10*V4
N28   V12=V11*V6
N29   X=V9  Y=V12      [NOTE 1.]
N30   V7=V7+V3
N31   V14=V13-V7
N32   BGE P2
N33   G99
```

[NOTE 1] Because we are only moving a short distance we use a straight line movement, rather than a curve. Once the whole ellipse is produced the straight line is not usually noticeable (providing the incrementing angle (V3) - which determines the length of the straight line - is small enough).

www.DaltonsWadkin.com

It can be said, though, that the programme is still a little bit limited. Lines N7, N9, N10 and N15 all move to positions which are not totally flexible ie. not universal for any size of ellipse.

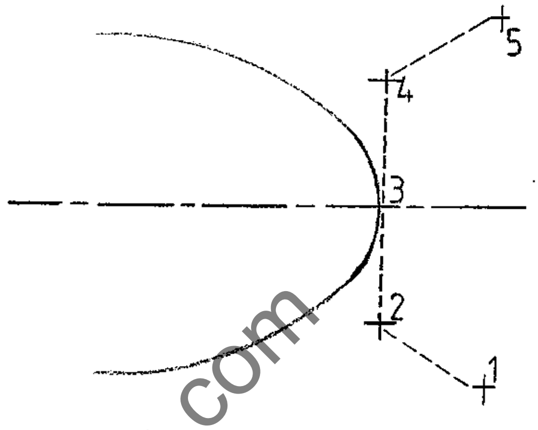
We can modify this programme still further to make it more flexible.

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=400 VY=300
N5     G820
N6     V1=300  V2=180  V3=2  V18=DEPTH
N7     G22 P1
N8     G0 T01
N9     X=V19  Y=V21  Z=V17
N10    G42 T01
N11    G1  F6000
N12    X=V4  Y=V20
N13    F1500
N14    Z=V22
N15    G1  F4000
N16    G22 P2
N17    X=V4  Y=15
N18    G0  Z5
N19    G40
N20    X=V19  Y=V16
N21    G53
N22    G0  X450 Y600 Z0 T00
N23    M30
N24    $1
N25    V4=V1/2
N26    V5=V2/2
N27    V6=V5/V4
N28    V7=0  V13=360  V15=20
N29    V16=40  V17=5
N30    V19=V4+V15
N31    V20=V15* -1
N32    V21=V16* -1
N33    V22=V18* -1
N34    G99
N35    $2
N36    V8=COS V7
N37    V9=V8*V4
N38    V10=SIN V7
N39    V11=V10*V4
N40    V12=V11*V6
N41    X=V9  Y=V12
N42    V7=V7+V3
N43    V14=V13-V7
N44    BGE P2
N45    G99
```


www.DaltonsWadkin.com

With this example we have changed the format around slightly and added one or two functions in.

As you can see, sub-programme 1 is called up straight after the input parameters are loaded (N7). This sub-programme now calculates the ratio and some values which are used as points for 'entry' and 'exit' moves into and out of the ellipse.



Point 1 is 20 mm away from the edge of the ellipse and 40 mm below Y0. From this point we can move to point 2 putting on cutter compensation. [This distance should be large enough for cutters up to about 27 mm radius].

At point 2 the cutter is positioned to the cutting depth.

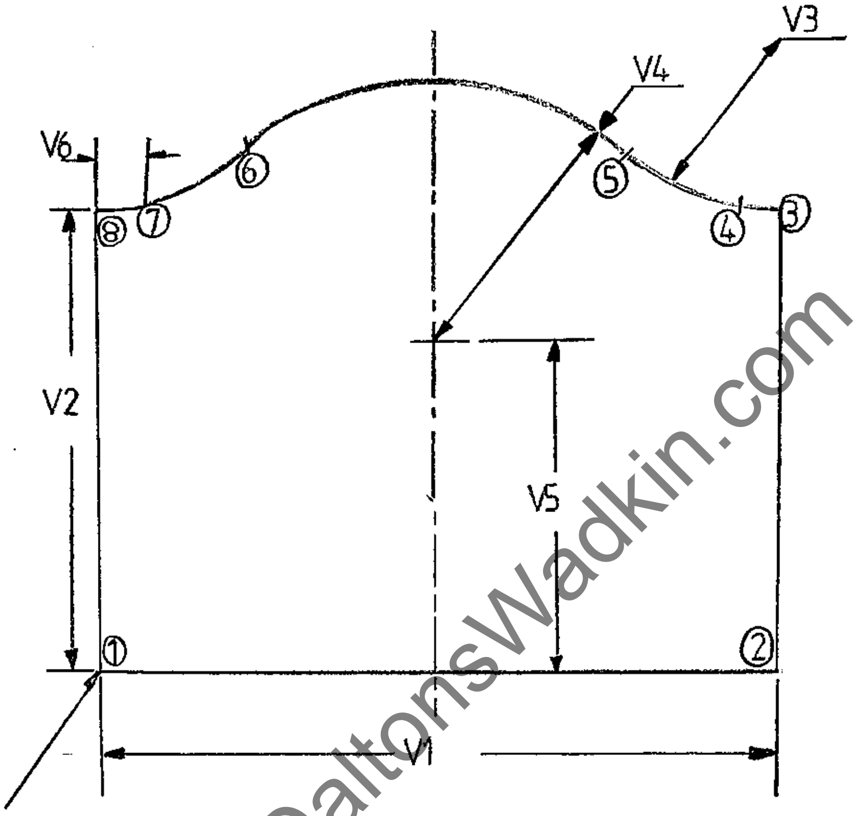
The machine now runs through sub-programme 2 numerous times and completes the ellipse. At this stage the machine is positioned at point 4, the cutter raised and the cutter compensation taken off between points 4 and 5.

This programme will now work for any size of ellipse.

Now that we have tried one practical example, let us try another one which is very popular in the woodworking industry.

Example 11

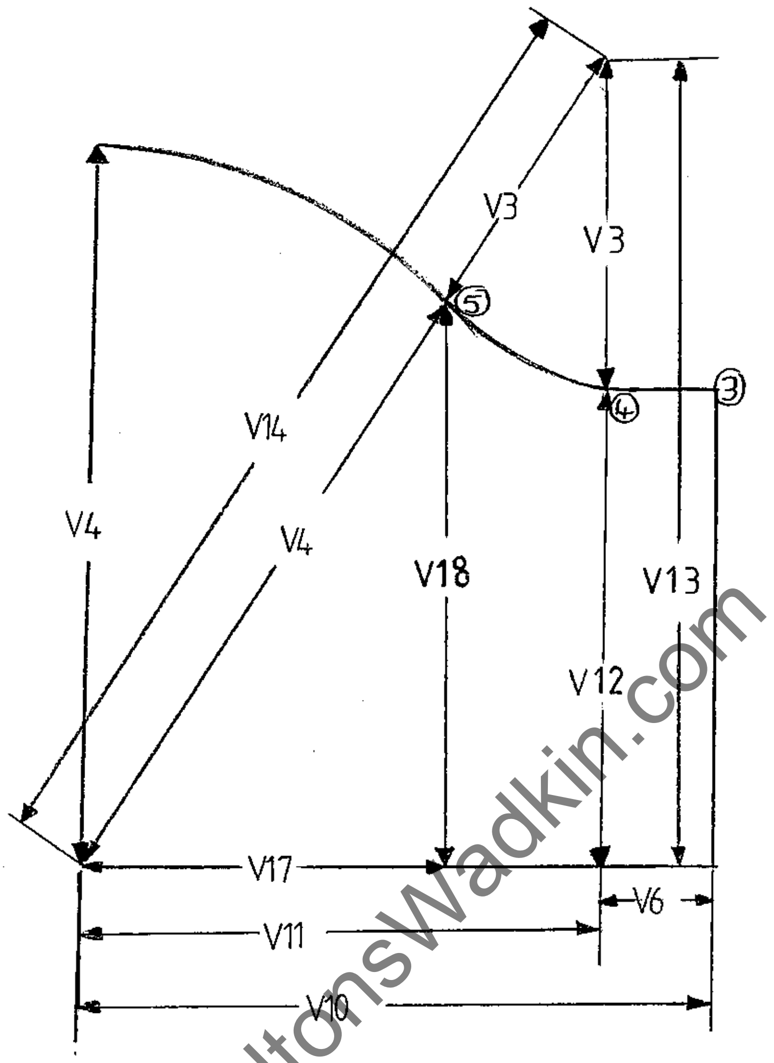
RAISED AND FIELDED KITCHEN PANEL



These are the 6 basic input variables used to calculate the points for the panel. These may be measured off an existing panel.

The numbers in circles denote the points for which we need to calculate X and Y co-ordinates. In actual fact, points (1), (2), (3), (4), (7) and (8) pose no real problem. It is only points (5) and (6) which are relatively difficult to determine.

We do this using the 'similar triangles' principle. You will see how this works as we go through the calculations.



PICTORIAL REPRESENTATION OF CALCULATION PARAMETERS

First we need to calculate the sides of the triangle V11, V13, V14. This we can do by means of simple addition, subtraction and division.

V10 = V1/2
V11 = V10- V6
V12 = V2 - V5
V13 = V12+ V3
V14 = V4 + V3

Now, we can say that all three sides have a certain relationship with each other. Taking this into account, we can determine two ratios:-

```
V15 = V11/V14
V16 = V13/V14
```

This means now that if we use these ratios, we can calculate V17 and V18.

```
V17 = V4 * V15
V18 = V4 * V16
```

Now that we have found these two values, it is again a matter of simple arithmetic to determine all the co-ordinates for the panel.

```
V19 = V1 - V6
V20 = V10+V17
V21 = V5+ V18
V22 = V10-V17
```

Let us combine all these calculations in a programme for the CC 100M control. [NOTE:- It is very important to get the correct values in the input parameters - particularly V3 and V4, otherwise it will not produce the correct size panel].

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=300  V2=460  V3=80
N7    V4=163  V5=330  V6=28
N8    G22 P1
N9    G0  X-70 Y-20 Z5 T01
N10   G42 T01
N11   G1  X-50 Y0    F6000
N12   G1  Z-15 F1500
N13   G1  F4000
N14       X=V1
N15            Y=V2
N16       X=V19
N17   G2
N18       X=V20 Y=V21 R=V3
N19   G3
N20       X=V22 Y=V21 R=V4
N21   G2
N22       X=V6  Y=V2 R=V3
N23   G1  X0
N24             Y-50
```

```
N25   G0  Z5
N26   G40 X-20  Y-70
N27   G53
N28   G0  X450  Y600 Z0 T00
N29   M30
N30   $1
N31   V10=V1/2
N32   V11=V10-V6
N33   V12=V2-V5
N34   V13=V12+V3
N35   V14=V4+V3
N36   V15=V11/V14
N37   V16=V13/V14
N38   V17=V4*V15
N39   V18=V4*V16
N40   V19=V1-V6
N41   V20=V10+V17
N42   V21=V5+V18
N43   V22=V10-V17
N44   V3=V3* -1
N45   V4=V4* -1
N46   G99
```

<u>Example 12</u>

As with some previous examples, this programme is not very versatile; it will only go round the component one way (anti-clockwise), at one depth and one feedrate.

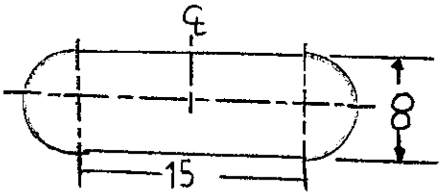
The following programme is based on the same principles, though it is rather more flexible.

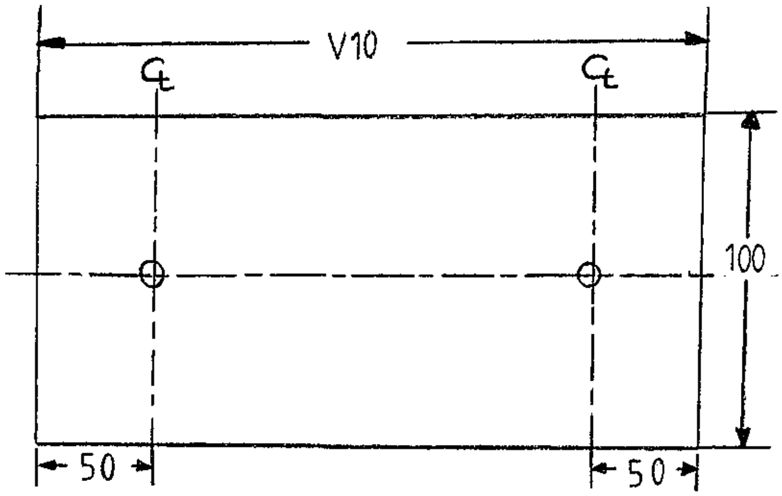
With this example, we can also use a variable for the feed, a variable for the depth of cut and a variable to control which way round we cut - either conventionally (anti-clockwise around the component) or climb cutting (clockwise around the component).

* VF = Feed in mm/min, eg. 4000
VZ = Depth of cut in mm, eg. -20 (include the minus sign).

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200
N5     G820
N6     V1=300  V2=460  V3=80
N7     V4=163  V5=330  V6=28
N8     VF=FEED  VZ=DEPTH OF CUT
N+8    VC=1(FOR CLIMB CUT) OR 0(FOR CONVENTIONAL CUT)
N9     G22 P1
N10    G53
N11    G0  X450 Y600 Z0 T00
N12    M30
N13    $1
N14    G22 P2
N15    V24=1
N16    V25=V24-V
N17    BEQ P3
N18    G0  X-80 Y-70 Z5 T01
N19    G42 T01
N20    G1  X-50 Y0   F6000
N21    G1  F1500
N22    Z=VZ
N23    X=V1 F=VF
N24          Y=V2
N25    X=V19
N26    G2
N27    X=V20 Y=V21 R=V3
N28    G3
N29    X=V22 Y=V21 R=V4
N30    G2
N31    X=V6  Y=V2  R=V3
N32    G1  X0
N33          Y-50
N34    G0  Z5
N35    G40 X-20 Y-70
N36    G99
N37    $3
N38    G0  X-20 Y-70 Z5 T01
N39    G41 T01
N40    G1  X0   Y-50 F6000
N41    G1  F1500
N42    Z=VZ
N43    Y=V2  F=VF
N44    X=V6
N45    G3
N46    X=V22 Y=V21 F=V4
N47    G2
N48    X=V20 Y=V21 R=V3
N49    G3
N50    X=V19 Y=V2  R=V4
N51    G1
N52    X=V1
N53          Y0
N54    X-50
N55    G0 Z5
N56    G40 X-70 Y-20
N57    G99
N58    $2
N59    V10=V1/2
N60    V11=V10-V6
N61    N12=V2-V5
N62    V13=V12+V3
N63    V14=V4+V3
N64    V15=V11/V14
N65    V16=V13/V14
N66    V17=V4*V15
N67    V18=V4*V16
N68    V19=V1-V6
N69    V20=V10+V17
N70    V21=V5+V18
N71    V22=V10-V17
N72    V4=V4* -1
N73    V4=V4* -1
N74    G99
```

PROBLEM 8





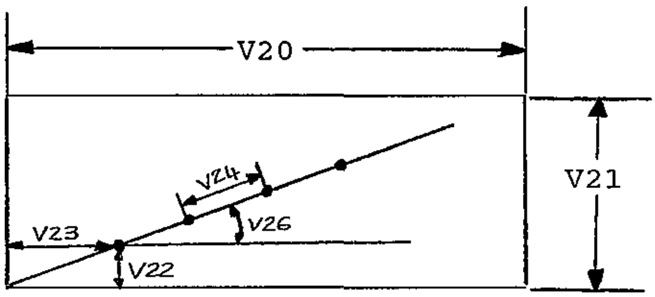
V10 = Input variable - (length of component (MUST BE MULTIPLE OF 50))

(a) Write programme to produce series of slots - all at 50 mm pitches (from ℄ to ℄). Programme must determine correct number of slots.

(b) As (a) but incorporate a check which terminates the programme if V10 is not a multiple of 50 (and displays an error message).

PROBLEM 9

V25 = NO. OF HOLES



Write a programme to drill a series of holes in a panel. The holes are along a straight line which may rotate through an angle - V26. The programme must halt if faulty data is input ie:- The board not being long enough or wide enough to accept the number of holes or the angle being more than 90^{o}.

4. FURTHER USE OF PARAMETRIC FUNCTIONS

As mentioned in the introduction, you have been using VX and VY values from 'Day 1'. We will now explain how these values are dealt with and the corresponding effect.

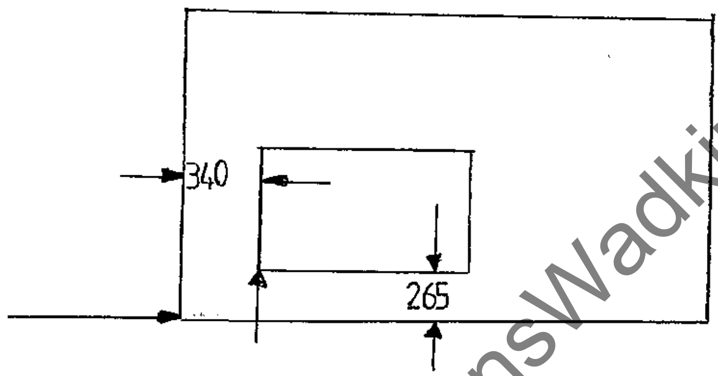
Example 13

Before we explain in detail the use of the offset cycles, it may be useful for us to review some basic N.C. programming first.

ZERO SHIFTS

We have 6 zero shifts available - G54 through to G59. These are used to 'locate' a component datum on the table. [G53 cancels any active zero shift value].

Consider this:-



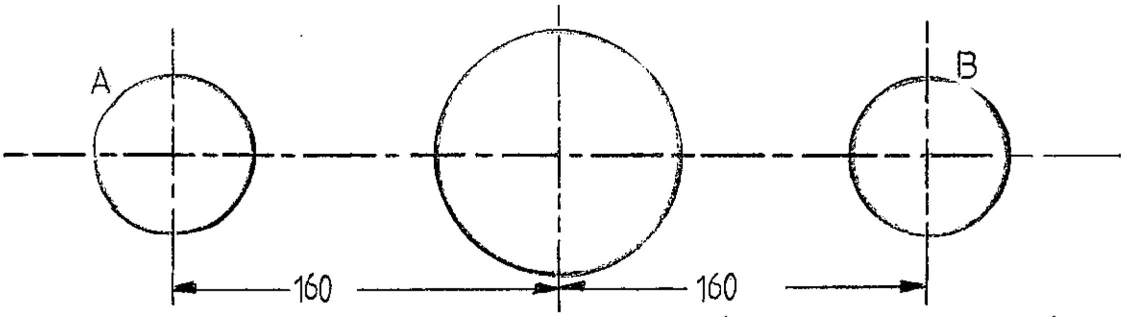
To set the component datum using zero shifts, we need to store the X and Y values in the zero shift store. For this example we will put these in the G54 store. This is done before the programme is first run (during the 'setting-up' of the machine).

Now, if we want to move to a point relative to the component datum, we just call up G54 in our programme. This transfers the active datum to the position which is stored in the zero shift table under G54.

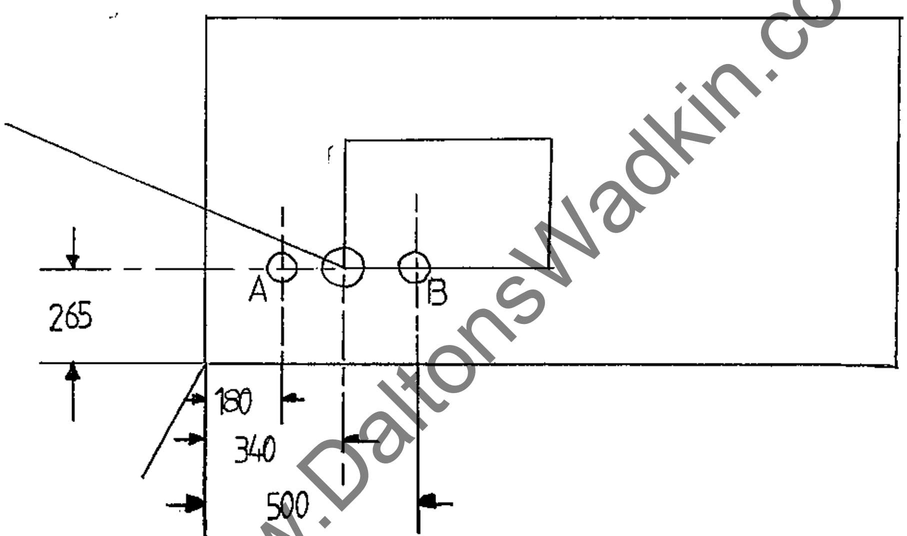
```
ie.   G90                    Machine datum is active datum till here
      G53
      G0 Z0 T00
      G54
      G0  X-20 Y-20 Z5 T01             Value stored in G54 is
      etc.                             now active datum. (ie.
                                       X340 Y265 is now treated
                                       as X0 Y0)
              [To cancel any active
               zero shift use G53]
      G53
      G0  X450 Y600 Z0 T00             Machine datum is
      M30                              active again.
```

Now, if we have a number of heads on the machine (indeed, this is very common) careful consideration has to be made of how we set the datum points.

For example, on a CC 2000 machine with one router head and 2 drills, we have the following head centres:-



Let us transpose the head centres drawing onto the previous diagram for the zero shifts.



Now, in this situation, we have the router head positioned over the component datum. This is at X340 Y265.

If we now move the left drill, A, over the datum point we have to move an extra 160 mm (away from X0). This reading would then be X500 Y265.

In turn, if we move the right hand drill head, B, over the datum point we have to move back 160 mm. This would be X180 Y265.

So, if we want to use both drills and the router head, we need to use 3 zero shifts.

These would have the values:-

G54 X340 Y265 - Router head
G55 X500 Y265 - Drill head A
G56 X180 Y265 - Drill head B

As previously mentioned, we only have six zero shift stores. If we wanted to cut more than two components or we had more heads on the machine, we would have a severe limitation due to the number of zero shift stores available.

Another problem occurs in that we have to calculate the X values of the latter 2 zero shifts (for the drills) using the distances between the drills and the router head.

This all appears very complex, long winded and limited in its scope.

AUTO-OFFSET CYCLES

To overcome this, Wadkin have written an 'AUTO-OFFSET CYCLE'. The benefits of this are:-

1. We only need to state a datum point once per component (using VX and VY).
2. We state this datum point in the programme (not zero shift store).
3. We don't have to do any calculations.
4. We can do many components just be re-stating a new VX and VY point within our programme.

Let us now consider how we use the AUTO-OFFSET CYCLE and how it works.

In any programme we typically use lines at the start such as:-

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=340 VY=265
N5    G820
      etc.
```

Here, we note that in line N4 we state the VX and VY values and immediately after this (in line N5) we put G820.

Further on in the programme, should we want to use the left drill, A, we would programme G821 and in if we wanted to use the right drill, B, we would programme G822.

The G820 cycle is the key to all this. Remember, cycles are used in a similar way to sub-programmes, but they are global (or universal) which means the same cycle can be called up by any programme.

Let us look at a typical G820 cycle for a CC2000 machine with 2 drills.

[NOTE - Your machine's cycle 20 may look different to this, but the principle is the same].

```
N1     G53
N2     V20=-160  V21=160
N3     V22=VX-V20
N4     V23=VX-V21
N5     TRF=G54 X=VX Y=VY
N6     TRF=G55 X=V22 Y=VY
N7     TRF=G56 X=V23 Y=VY
N8     G54
N9     M2
```

We will now consider this cycle line by line.

N1	G53	- As you know, this cancels any active zero shifts (just to make sure there are none left active by mistake).
N2	V20=-160 V21=160	- This line stores the distances between the router head and the drill heads. [these are put in at Wadkin Colne after measuring the exact distances].
N3	V22=VX-V20	- This subtracts the current value of VX (which, remember, was stated in the programme) from the value stored in V20 (above). Using the previous example this would be 340 - (-160) = 500 [Note:- Subtracting a minus value gives a positve result].
N4	V23=VX-V21	- As above, except this time when we perform the calculation V21 is positive VX:- 340 - 160 = 180.
N5	TRF=G54 X=VX Y=VY	- Now, this line will actually TRANSFER information TO the zero shift table. In this case it stores the VX value in the X part of the G54 store in the VY value in the Y part of the G54 store. It will overwrite any existing value in the store.
N6	TRF=G55 X=V22 Y=VY	- As above, except V22 is transferred into the G55 'X' store, VY is transferred into the 'Y' store.
N7	TRF=G56 X=V23 Y=VY	- Again, as before except storing information in the G56 zero shift and using V23 for the 'X' value.
N8	G54	- This calls up the G54 zero shift and makes it active (as explained previously). This ensures that when we end the cycle, G54 is active.
N9	M2	- End of cycle.

So, we see that once the cycle G820 has been performed, G54, G55 and G56 are loaded with values for the router head, drill head A and drill head B respectively.

G821, G822 are very simple cycles, as follows.

CYCLE 21		CYCLE 22	
N1	G53	N1	G53
N2	G55	N2	G56
N3	M2	N3	M2

This means that if we were to call up G821, N1 would cancel any active zero shift, N2 would call up G55 and then end the cycle. This now means any point programmed is in respect of the left drill being set to the component datum. Its a similar situation when we call up G822.

The function which allows such a cycle to work is the transfer function - TRF. This is only one of the uses of this function, ie. loading values into the zero shift store. Here is a brief explanation of the other uses, along with some other functions.

[NOTE:- The functions described here are not used as commonly as those previously considered. They are only used in more specific cases. Once the following functions are understood, they can be used wherever it is felt necessary]

COPYING VALUES FROM A ZERO SHIFT TABLE:-

This line :-

```
TRF=G54 V1=X V2=Y
```

Would copy the X and Y values out of the G54 zero shift store. These values would be loaded into V1 and V2 respectively. (The values actually remain in the zero shift store and are not deleted).

These values would be used, then, for further calculations or merely for storage somewhere else.

[NOTE:- V1 and V2 could, in fact, be any variable. Also, G54 could be any zero shift store; G54 -G59].

COPYING G92 ZERO SHIFT :-

Rather similar to the last example.

```
TRF=G92 V1=X V2=Y
```

This would transfer the active X and Y datum (which must have been set using a G92 value, rather than a zero shift (eg. G54) value to V1 and V2 respectively.

Again, the reasons for doing this are numerous and varied. (Also, similar to before, the current G92 active values are not deleted, just copied).

COPY AN ACTIVE POLE

An active pole is set when using polar co-ordinates. It is the centre about which points are defined using an angle and a distance from the centre point (instead of using X's and Y's).

To copy the centre point values we would programme the following line:-

TRF=G20 V1=X V2=Y

This would, similar to before, load the X value into V1 and Y value into V2.

LOADING VALUES INTO THE TOOL STORE

This is a similar principle to loading values into the zero shift store.

Here, we use the function COR - as below.

COR=T01 R=V1 L=V2 (T01 could be any tool number, V1 and V2 could be variable number).

This will put the current value of V1 into the tool radius compensation store for T01, overwriting any existing value. At the same time it will also put the current value of V2 into the tool length compensation store for T01, again overwriting any existing value.

This is a very useful function in some circumstances.

COPYING VALUES FROM THE TOOL STORE

This has the opposite effect to the last function.

COR=T01 V1=R V2=L (T01 could by any tool number, V1 and V2 could be any variable number).

This will put the current radius value of T01 into variable V1 and put the current length of T01 into variable V2. Again, note that the tool store values are not deleted, just copied from the store.

This function and the last function work quite will together in some circumstances.

For example, if it is required to take a finishing cut on a component, we can use the following method.

1.	Run sub-programme (or cycle) which contains information for contouring the shape; ('roughing out' the shape)	[SUB-PROGRAMME]
2.	Transfer active tool number into V50	V50=T
3.	Transer current radius of tool into V51	COR=V50 V51=R
4.	Size of finishing cut (0.1mm off, for eg., all round component).	V52=0.1
5.	Subtract 'size of finishing cut' from radius compensation value of cutter	V53=V51-V52
6.	Load new radius back into tool store	COR=V50 R=V53
7.	Run sub-programme again - this time it will take a further 0.1 mm of all round	[SUB-PROGRAMME]

[Note:- This may need further adaptation to run on your machine and with your particular components but it highlights a principle which can be put to practical use].

TIME FUNCTION

TIM V1 (V1 could be any variable)

This function allows us to record the time from pressing the cycle start button. (TIME IN SECONDS).

This can be used to time a whole cycle, or part of a cycle.

If we want to time the whole cycle, we could put:-

TIM VT

for example, just before the M30. This woul load the cycle time into variable VT.

If, on the other hand, we wanted to time a particular part of the cycle, we would use the function at the start and finish of the relevant part of the cycle and subtract the two times.

An example will help to clarify this point:-

```
G90
G53
G0 Z0 T00
VX=200 VY=200
G820
M8 T01
G22 P1
M9
G0 Z5 T02
M68
G823
TIM VR
G22 P1
TIM VS
G53
G0 X650 Y800 Z0 T00 M69
VT=VS-VR (*)
M30
$1
G0 X-20 Y-20 Z5
G42
G1 X-10 Y0 F6000
G1 Z-10 F1500
G1 X200 F4500
   Y200
   X0
   Y-10
G0 Z5
G40 X-20 Y-20
G99
```

(*) VT will now be loaded with the time taken to complete sub-programme $1 for the second time.

APPENDIX 1

WADKIN (COLNE) PLC

PARAMETRIC PROGRAMMING COURSE

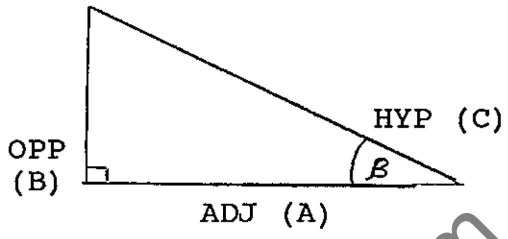
FORMULA SHEET

FOR RIGHT ANGLE TRIANGLES:-

ADJ= ADJACENT

HYP= HYPOTENUSE

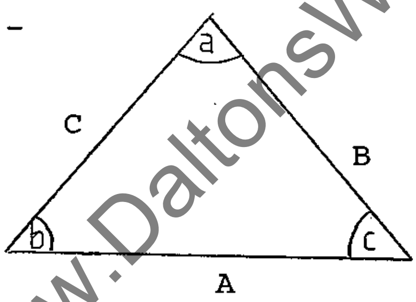
OPP= OPPOSITE



$\tan \beta = \frac{OPP}{ADJ}$ $\sin \beta = \frac{OPP}{HYP}$ $\cos \beta = \frac{ADJ}{HYP}$

$C^2 = A^2 + B^2$

FOR ANY TRIANGLE:-



SINE RULE:- $\frac{A}{\sin a} = \frac{B}{\sin b} = \frac{C}{\sin c}$

COSINE RULE:- $A^2 = B^2 + C^2 - 2BC \cos a$

Transposed gives:-

$\cos a = \frac{B^2 + C^2 - A^2}{2BC}$

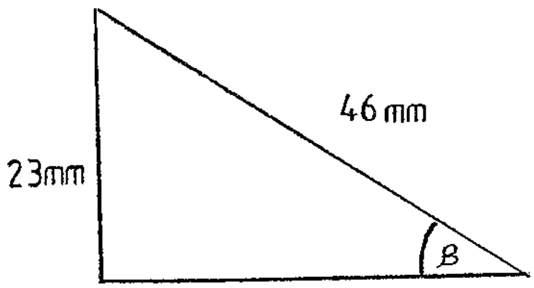
Also, all angles of a triangle = 180°

$\tan \beta = \frac{\sin \beta}{\cos \beta}$ $\sin^2 \beta + \cos^2 \beta = 1$

APPENDIX 2

1. BASIC TRIGONOMETRY AND ALGEBRA EXAMPLES

Example 1

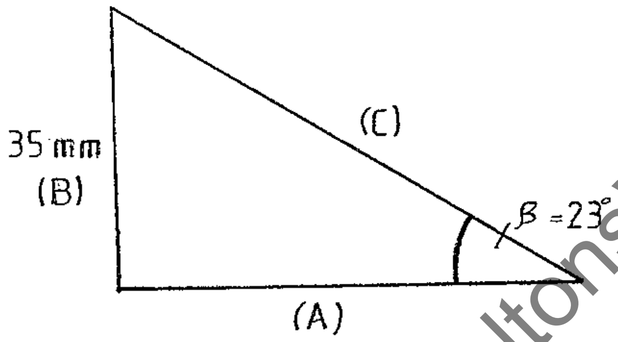


To determine angle ß:-

$$\sin \beta = \frac{OPP}{HYP} = \frac{23}{46}$$

$$\sin \beta = 0.5$$
$$\beta = \sin^{-1} 0.5$$
$$\beta = 30°$$

Example 2



To determine length A:-

$$\tan \beta = \frac{OPP}{ADJ}$$

$$\tan 23° = \frac{35}{A}$$

$$A = \frac{35}{\tan 23°} = \frac{35}{0.4245}$$

$$A = 84.45 \text{ mm}$$

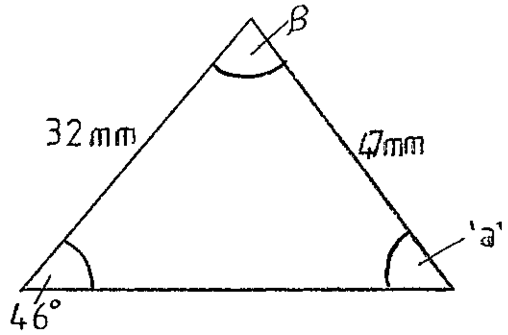
To determine length C:-

$$C^2 = A^2 + B^2$$
$$= 82.45^2 + 35^2$$
$$= 8023$$

$$C = \sqrt{8023}$$

$$C = 89.57\text{mm}$$

Example 3



(i) To determine angle 'a':-

Using formula:-

$$\frac{A}{\text{Sin a}} = \frac{B}{\text{Sin b}} \quad \text{(SINE RULE)}$$

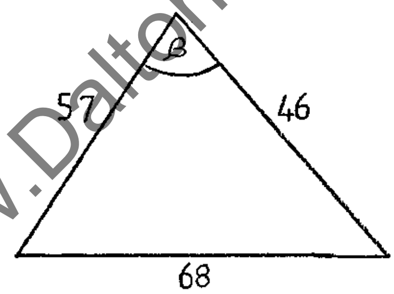
$$\frac{32}{\text{Sin'a'}} = \frac{47}{\text{Sin}46°} \qquad \text{Sin 'a'} = \frac{32\ \text{Sin}46°}{47} = 0.4898$$

Sin 'a' = 0.4898
'a' = 29.325°(ii)

(ii) To determine angle ß:-

Sum of angles of a triangle = 180°
therefore ß = 180° - (46°+ 29.325°)
= 180° - (75.325°)
= 104.675°

Example 4



To determine angle ß :-

Using formula:-

$$\text{Cos ß} = \frac{B^2 + C^2 - A^2}{2BC}$$

$$\text{Cos ß} = \frac{57^2 + 46^2 - 68^2}{(2 \times 57 \times 46)}$$

$$= \frac{741}{5244} = 0.1413$$

$$ß = \text{Cos}^{-1} 0.1413$$

$$= 81.877°$$

APPENDIX 3

SUGGESTED ANSWERS TO PROBLEMS

PROBLEM 1

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=(A)  V2=(B)
N7    G0  X-20 Y-20 Z5 T01
N8    G42 T01
N9    G1  X-10 Y0   F6000
N10   G1  Z-10  F1500
N11   G1  F4000
N12       X=V1
N13            Y=V2
N14       X0
N15            Y-10
N16   G0  Z5
N17   G40 X-20  Y-20
N18   G53
N19   G0  X650 Y800 Z0 T00
N20   M30
```

To run this programme to produce component 1, we would put a value of 120 in V1 and 40 in V2. When this programme is executed it will load V1 and V2, then in lines 12 and 13 it will move to the loaded values ie. X=120
Y=40

To produce components 2,3 and 4 we would just change V1 and V2

PROBLEM 2

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=(A)  V2=(B)  N3=(C)  V4=(D)
N7    G0  X-20 Y-20 Z5 T01
N8    G42 T01
N9    G1  X-10 Y0   F6000
N10   G1  Z-10 F1500
N11   G1  F4000
N12       X=V1
N13       X=V3 Y=V4
N14       X=V2
N15       X0   Y0
N16       X-5  Y-10
N17   G0  Z5
N18   G40 X-20 Y-20
N19   G53
N20   G0  X650 Y800 Z0 T00
N21   M30
```

Again, we have a similar principle to the last last problem.

PROBLEM 3

```
N1     G90
N2     G53
N3     G0   Z0 T00
N4     VX=200 VY=200
N5     G820
N6     V1=? V2=? V3=? V4=?
N7     G0   X-20 Y-20 Z5 T01
N8     G42 T01
N9     G1   X-10 Y0 F5000
N10    G1   Z-10 F1500
N11    G1   F4000
N12         X=V1
N13              Y=V3
N14    V5=V1-V4
N15         X=V5
N16    V6=0
N17         X=V6 Y=V2
N18              Y-10
N19    G0   Z5
N20    G40 X-20 Y-20
N21    G53
N22    G0   X450 Y600 Z0 T00
N23    M30
```

PROBLEM 4

```
N1    G90
N2    G53
N3    G0 Z0 T0
N4    VX=200 VY=200
N5    G820
N6    V1=?
N7    G22 P1
N8    G0  X-20 Y-20 Z5 T01
N9    G42 T01
N10   G1  X-10 Y0   F6000
N11   G1  Z-10 F1500
N12   G1  F4000
N13       X=V6
N14       X=V7 Y=V1
N15            Y-10
N16   G0  Z5
N17   G40 X-20 Y-20
N18   G53
N19   G0  X450 Y600 Z0 T00
N20   M30
N21   $1
N22   V2=76
N23   V3=SIN V2
N24   V4=COS V2
N25   V5=V3/V4
N26   V6=V1/V5
N27   V7=0
N28   G99
```

NOTE:- There is no TANGENT function on the control. This is overcome using the following formula:-

$$TAN\ \beta = \frac{SIN\ \beta}{COS\ \beta}$$

www.DaltonsWadkin.com

PROBLEM 5

```
N1     G90
N2     G53
N3     G0   Z0 T00
N4     VX=200 VY=200
N5     G820
N6     V1=? V2=? V3=?
N7     G22 P1
N8     G0   X-20 Y-20 Z5 T01
N9     G42 T01
N10    G1   X-10 Y0 F5000
N11    G1   Z-10 F1500
N12    G1   F4000
N13         X=V1
N14         X=V3 Y=V9
N15              Y=V10
N16         X=V11
N17              Y-10
N18    G0   Z5
N19    G40 X-20 Y-20
N20    G53
N21    G0   X450 Y600 Z0 T00
N22    M30
N23    $1
N24    V4=V3-V1
N25    V5=40
N26    V6=SIN V5
N27    V7=COS V5
N28    V8=V6/V7
N29    V9=V4*V8
N30    V10=V2+V9
N31    V11=0
N32    G99
```


www.DaltonsWadkin.com

PROBLEM 6

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200
N5     G820
N6     V1=?
N7     G22 P1
N8     G0  X-20 Y-20 Z5 T01
N9     G42 T01
N10    G1  X-10 Y0 F6000
N11    G1  Z-10 F1500
N12    G1  F4000
N13        X=V1
N14              Y=V3
N15        X=V7  Y=V9
N16        X=V10 Y=V3
N17              Y-10
N18    G0  Z5
N19    G40 X-20 Y-20
N20    G53
N21    G0  X450 Y600 Z0 T00
N22    M30
N23    $1
N24    V2=40  V3=50
N25    V4=SIN V2
N26    V5=COS V2
N27    V6=V4/V5
N28    V7=V1/2
N29    V8=V6*V7
N30    V9=V8+V3
N31    V10=0
N32    G99
```

PROBLEM 7

```
N1    G90
N2    G53
N3    G0  Z0 T00
N4    VX=200 VY=200
N5    G820
N6    V1=? V2=? V3=?
N7    G22 P1
N8    G0  X-20 Y-20 Z5 T01
N9    G42 T01
N10   G1  X-10 Y0 F5000
N11   G1  Z-10 F1500
N12   G1  X0 F4000
N13   G91
N14       X=V3
N15             Y=V1
N16       X=V10 Y=V9
N17             Y=V2
N18       X=V10 Y=V9
N19       X=V10 Y=V11
N20             Y=V12
N21       X=V10 Y=V11
N22   G90
N23             Y-10
N24   G0  Z5
N25   G40 X-20 Y-20
N26   G53
N27   G0 X450 Y600 Z0 T00
N28   M30
N29   $1
N30   V4=V3/V4
N31   V5=30
N32   V6=SIN V5
N33   V7=COS V5
N34   V8=V6/V7
N35   V9=V4*V8
N36   V10=V4*-1
N37   V11=V9*-1
N38   V12=V2*-1
V39   G99
```

PROBLEM 8

(a)

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200  [BOTTOM LEFT HAND CORNER]
N5     G820
N6     V10= (LENGTH)
N7     G22 P1
N8     G0  X42.5 Y50 Z5 T01
N9     $2
N10    G22 P3
N11    DEC V6
N12    BEQ P99
N13    G91
N14    G0  X35
N15    G90
N16    G24 P2
N17    $99
N18    G53
N19    G0  X450 Y600 Z0 T00
N20    M30
N21    $1
N22    V2=50
N23    V3=V2*2
N24    V4=V10-V3
N25    V5=V4/50
N26    V6=V5+1
N27    G99
N28    $3
N29    G61
N30    G91
N31    G1  Z-10 F1500
N32        X15  F3000
N33    G0  Z10
N34    G62
N35    G90
N36    G99
```

PROBLEM 8

(b)

```
N1     G90
N2     G53
N3     G0 Z0 T00
N4     VX=200 VY=200  (BOTTOM LEFT HAND CORNER)
N5     G820
N6     V10= (LENGTH)
N7     G22 P1
N8     G0  X42.5 Y50 Z5 T01
N9     $2
N10    G22 P3
N11    DEC V6
N12    BEQ P99
N13    G91
N14    G0  X35
N15    G90
N16    G24 P2
N17    $99
N18    G53
N19    G0  X450 Y600 Z0 T00
N20    M30
N21    $1
N22    V11=V10/50
N23    V12=V11
N24    INC V12
N25    DEC V12
N26    V13=V11-V12
N27    BNE P98
N28    V2=50
N29    V3=V2*2
N30    V4=V10-V3
N31    V5=V4/50
N32    V6=V5+1
N33    G99
N34    $3
N35    G61
N36    G1  Z-10 F1500
N37        X15  F3000
N38    G0  Z10
N39    G62
N40    G90
N41    G99
N42    $98
N43    M0
N44    (INPUT VARIABLE IS NOT A MULTIPLE OF 50 - ENTER NEW VALUE OF V10)
N45    G24 P98
```

PROBLEM 9

```
N1    G90
N2    G53
N3    G0 Z0 T00
N4    VX=200 VY=200  (BOTTOM LEFT HAND CORNER)
N5    G820
N6    V20=?  V21=?  V22=?  V23=?
N7    V24=?  V25=?  V26=?
N8    G22 P1
N9    G=V19 X=V23 Y=V22
N10   G81  V1=5  V2=-10
N11   G1   F800  T01 M20
N12   G0   X=V23 Y=V22
N13   G=V30 P=V31 L=V32
N14   G80
N15   G53
N16   G0  X450 Y600 Z0 T00 M38
N17   M30
N18   $1
N19   V10=V25*V24
N20   V11=SIN V26  V12=COS V26
N21   V13=V10*V12  V14=V10*V11
N22   V15=V23+V13
N23   V16=V22+V14
N24   V17=V20-V15  BLE P99
N25   V18=V21-V16  BLE P98
N26   V19=20  V30=22  V31=2
N27   V32=V25-2
N28   V33=V26-90  BGT P97
N29   G99
N30   $2
N31   G91
N32   A=V26  D=V24
N33   G90
N34   G99
N35   $99
N36   M0
N37   (BOARD IS NOT LONG ENOUGH TO ACCEPT THIS NO. OF HOLES)
N38   G24 P99
N39   $98
N40   M0
N41   (BOARD IS NOT WIDE ENOUGH TO ACCEPT THIS NO. OF HOLES)
N42   G24 P98
N43   $97
N44   M0
N45   (ANGLE IS LARGER THAN 90 DEGREES)
N46   G24 P98
```

APPENDIX 4

PARAMETRIC PROGRAMMING COURSE

QUICK REFERENCE LIST

STATEMENT	FUNCTION
V1=n	LOAD a numerical value into a variable store. (V1 can be anything from V1 - V99 & VA - VZ)
X=Vn [m=Vn]	EXECUTION instruction - N.C. addresses are loaded from variable stroe [m can be any one of the following addresses - X,Y,Z,E,I,J,K,A,D,G,F,R,S,T,M,H]
Vn=X [Vn=p]	TRANSFER active data into variable store [p can be any of the following addresses - X,Y,Z,E,I,J,K,A,D,F,R,S,T]
V1=V2	COPY value from one variable into another variable.
V1=V2+V3 (V1=V2+10)	ADDITION of two variables or a variable and an integer.
V1=V2-V3 (V1=V2-10)	SUBTRACTION of two variables or a variable and in integer.
V1=V2*V3 (V1=V2*10) and	MULTIPLICATION of two variables or a variable an integer.
V1=V2/V3 (V1=V2/3)	DIVISION of two variables or a variable and an integer.
V1=SQR V2	SQUARE ROOT of a variable.
V1=SIN V2	SINE of a variable (-360° ← V2 ←360°)
V1=COS V2	COSINE of a variable (-360° ← V2 ←360°)
V1(degrees)=ATN V2	ARCTANGENT of a variable.
INC V1	INCREMENT value of a variable - disregard digits after the decimal point.
DEC V1	DECREMENT value of a variable - disregard digits after decimal point.
BEQ P5 [BEQ V1]	JUMP to target $5 (or value of V1) if condition register = 0 (equal to zero.)
BNE P5 [BNE V1]	JUMP to target $5 (or value of V1) if condition register =0 (not equal to zero.)

BGT P5 [BGT V1]	JUMP to target $5 (or value of V1) if condition register >0 (greater than zero.)
BLT P5 [BLT V1]	JUMP to target $5 (or value of V1) if condition register <0 (less than zero.)
BGE P5 [BGE V1]	JUMP to target $5 (or value of V1) if condition register >=0 (greater than or equal to zero.)
BLE P5 [BLE V1]	JUMP to target $5 (or value of V1) if condition register <=0 (less than or equal to zero.)
COR=T10 V1=R V2=L [COR=Vn]	COPY values from tool store into variable store. [Selected tool can be established by value in Vn
COR=T10 R=V1 L=V2 [COR=Vn]	LOAD values into tool store from variable store. [Selected tool can be established by value in Vn
TRF=G54 X=V1 Y=V2 Z=V3	LOAD values into zero shift store from variable store. [Selected zero shift can be established by value in Vn].
TRF=G54 V1=X V2=Y V3=Z	COPY values from zero shift store into variable store. [Selected zero shift can be established by value in Vn].
TRF=G92 V1=X V2=Y	COPY active X and Y datum values (set by G92) in variable store.
TRF=G20 V1=X V2=Y	COPY active pole centre values (set by G20) into variable store.
TIM Vn	RECORD the time from the programme start in seconds.
TST Vn	SET condition register. [Condition register can in one of three states: positive(+); negative (- zero (0)]
TST G1 [TST Gn]	SET condition register (CR) to zero if G1 is active. [n can be 0-3, 17-19, 39, 53-59, 62, 63, 65, 66, 90, 94, 95, 97].
TST QX [TST Qn]	SET condition register (CR) to zero if 'X' axis mirrored. [n can be X, Y, Z, E].
TST QM	SET condition register (CR) to zero if dimensions are metric.