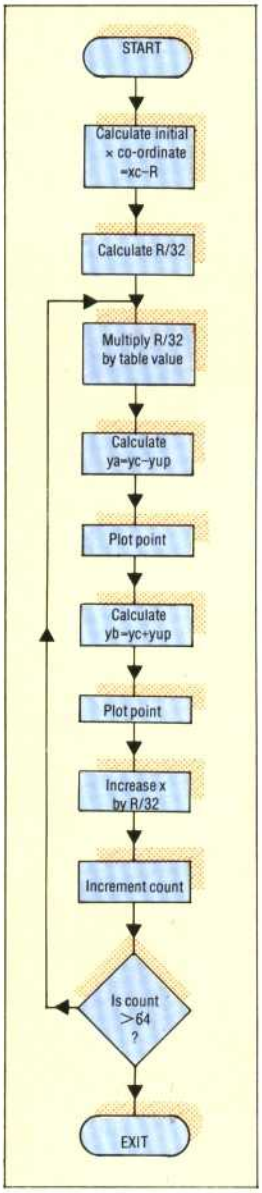

to calculate the corresponding y co-ordinate in the bottom half of the circle:

yb = ycentre + yup

DEVELOPING OUR ROUTINE

We can now map out the structure of the routine in detail. The flowchart shows how each point on the circumference can be calculated. We can see that the routine should be fairly fast as only one multiplication needs to be performed for each point plotted. However, there are two slight problems with the routine as it stands. First of all, we will not produce a continuous circle, only a series of points around the circumference. Secondly, although this technique produces well-defined circles when used from BASIC, there are inaccuracies when it is done in machine code.

The first problem can be solved by making use of Linesub (see page 419) to join the points with short lines, to produce a continuous circle. The second problem is due to inaccuracies in the calculation of the square root 'look-up' table. Calculating the values in BASIC and POKEing them into a series of bytes set aside for the table means that only the integer part of each value is actually stored. To get more well-defined circles, we must improve the accuracy of the values stored in the table. The maximum number to be stored is 32, and therefore we can multiply each number by eight before storing it, and still use only one byte per table entry. 32 is the only number, when multiplied by eight, which can't be held in a single byte. To simplify the routine we'll approximate this value. Our machine code routine can then divide the table value by eight, by performing three LSRs (logical shifts right) on it. More importantly, it can keep the remainder after division for use in further calculations.

The source code listing sets aside 65 bytes at the beginning of the program to store the table, the first byte of which is labelled. Subsequent entries in the table can be accessed by indexed addressing. The source code listing can be entered and assembled into memory as usual. However, before SAVEing the assembled code, the following BASIC program should be entered and RUN. It will not corrupt your object code, as this is located high in memory. The program calculates the 65 table values required by our Circsub program, multiplies these values by eight and POKEs the result into the area of memory set aside in your machine code program. Once the table creation program has been RUN the machine code can be SAVEd in the usual way, but ensure that you SAVE the table area in with the object code. The table starts at $C500. Once this has been done, the 'look-up' table will be loaded automatically whenever you load Circsub.

```
5 REM**** CREATE CIRCSUB TABLE ****
10 FORN=0TO64
20 X=SQR(64*N-N↑2)*8
25 X%=X:DF=X-X%
27 IFDF>=.5THENX%=X%+1
28 IFX%>255THENX%=255
29 POKE50432+N,X%
30 NEXT
```

The following program shows how Circsub can be used from within a BASIC program. All that is required is to specify the centre co-ordinates and the radius. The subroutine at line 2000 splits the x co-ordinate into LO-byte/HI-byte form, then POKEs the specified values to Circsub and makes the appropriate SYS call. Note that, as Circsub uses Linesub, which in turn uses Plotsub (see page 337), all three subroutines must be loaded at the start of the program. This program draws circles with increasing radii across the screen.

```
3 REM**** CIRCSUB TEST PROGRAM ****
5 DN=8:REM FOR CASSETTE DN=1
10 IFA=0THENA=1:LOAD"PLOTSUB.HEX",DN,1
15 IFA=1THENA=2:LOAD"LINESUB.HEX",DN,1
20 IFA=2THENA=3:LOAD"CIRCSUB.HEX",DN,1
100 GOSUB1000
110 YC=100:R=2
120 FORXC=30TO210STEP7
130 R=R+3
140 GOSUB2000
150 NEXT
160 GETA$:IFA$=""THEN160
170 GOSUB3000
180 END
1000 REM **** SET HIRES ****
1010 POKE49408,1:POKE49409,1
1020 POKE49410,3
1030 SYS49422
1040 RETURN
1050 :
2000 REM **** ENTER CIRCSUB ****
2010 CHI=INT(XC/256):CLO=XC-256*CHI
2020 POKE50497,CLO:POKE50498,CHI
2030 POKE50499,YC
2040 POKE50500,R
2050 SYS50521
2060 RETURN
2070 :
3000 REM **** CLEAR HIRES ****
3005 RESTORE
3007 PRINTCHR$(147):REM CLEAR SCREEN
3008 PRINTTAB(10)"CIRCSUB VARIABLES"
3009 PRINT
3010 POKE49408,0:SYS49422
3030 FORI=50497TO50497+23STEP2
3035 READA$
3040 PRINTTAB(2);A$,PEEK(I),
3045 READA$
3047 PRINTA$,PEEK(I+1)
3050 NEXT
3060 RETURN
3070 :
5000 DATAXCTRLO,XCTRHI,YCTR,RADIUS,INTR
5010 DATAREMR,TOTX,RESREM,RESLO,RESHI,OLDXLO
5020 DATAOLDXHI,NEWXLO,NEWXHI,OLDYA
5030 DATAOLDYB,NEWYA,NEWYB,XFLAG,YFLAG,OLDY,NEWY
5040 DATAINTTAB,REMTAB
```

As with the other high resolution subroutines for the Commodore 64, the machine code can also be entered as a series of DATA statements if you do not have an assembler. The listing below should be typed and RUN to load Circsub into memory. Note that the BASIC loaders for Linesub and Circsub should also be loaded and RUN prior to loading the demonstration program. As the three routines will now be in memory, lines 10, 15 and 20 of the demo program should be omitted. Note that the BASIC loader for Circsub already contains the look-up table data and so it is not necessary to RUN the 'create table' program in this case.

Coming Soon

The machine code section of The Home Computer Advanced Course has so far concentrated on home computers with the Z80 and 6502 microprocessor. But there is a third popular microprocessor, the 6809 chip, which is used in the Dragon and Tandy Color computers. The course will soon be turning its attention to this chip and teaching 6809 owners to program in machine code too