machine code graphics and sound

for the commodore 64

easy to load routines and ideas

mark england and david lawrence

```
C965  60                    0060
```

Commentary

Lines 230–430: These lines set up the variables necessary for the execution of the high resolution plot of a single pixel.

Lines 460–600: These are the lines which test to see if the pixel to be plotted falls within the window designated by the next routine. If not, rather than generate an error, the plot instruction would be ignored.

Lines 620–650: At this point the position must be a valid one so a jump is made to the HPLOT routine, with program execution returning from HPLOT to the line drawing routine.

Testing

This routine is necessary for the execution of what follows but cannot be tested effectively at this stage.

Part 3. The line drawing algorithm (LINE PLOTTING)

There are many methods of calculation, or algorithms, available to aid in the plotting of a straight line between two points. Of these perhaps the most widely known and used in microcomputers is 'Bresenham's algorithm'. The basis of this method is to plot a line which has a tendency to be straight along whichever axis, vertical or horizontal, expresses the greatest distance between the start and finish points for the line. As this line is plotted, however, a record is kept of the amount by which the line deviates from the correct position along the other axis. Whenever the deviation amounts to one unit, or pixel, the next pixel moves not only one position along the longest axis but also one pixel along the subsidiary axis.

Suppose that we wished to draw a line from position 0,0, the top left-hand corner of the screen, to position 10,1, a position one pixel down from the top and 10 pixels to the right. The first thing to do is to identify which axis has the greater movement and it is clearly the one where the line must move from zero to ten. We therefore set two variables, X1 and X2, equal to the start and finish points on this axis. Note that, for the purposes of this line drawing routine, the axis along which there is the greatest movement will *always* be called the X axis, regardless of whether it moves across the screen or up and down.

We now have two values set:

X1 = 0
X2 = 10

The two remaining coordinates are now saved in Y1 and Y2:

Y1 = 0
Y2 = 1

The two ends of line are now defined as X1,Y1 and X2,Y2. We now need two more variables to represent the *difference* between the ends of the line on both axes, and we shall call these DX and DY:

DX = X2 − X1 = 10
DY = Y2 − Y1 = 1

We begin the process of plotting the line by setting one more variable, E, which will be called the 'error term', equal to DX times − 1, so that:

E = − DX = − 10

We can now get down to plotting pixels. The first one to be plotted is at X1,Y1, the beginning of the line. In fact, we take a temporary copy of X1 and Y1 into two more variables X and Y, since later we are going to have to change the values and we do not wish to play around with the record of the start position of the line.

Having plotted the pixel we now add to the error term, E, twice the difference between Y1 and Y2, as recorded in DY. Having done this we test to see whether E is now greater than zero. In this case E started off as − 10 and has had two added, so it has not reached zero. We now add one to the X coordinate but, because E failed the test, do nothing to the Y coordinate.

X is now equal to one and Y is equal to zero and we now plot that pixel. The process is repeated five times in all, with the following pixels being plotted:

0,0
1,0
2,0
3,0
4,0
5,0

At this point, E, which has had two added to it six times, has become greater than zero. This is a sign that we need to move along the Y axis since what E actually records is how far we have deviated from the correct position on the Y axis if we were drawing a true straight line. Y is now increased by one, so that the next pixel to be plotted will be 6,1. E has twice DX subtracted from it and thus returns to − 8.

The remaining pixels are plotted at:

7,1
8,1

9,1
10,1

without E ever exceeding zero. By comparing differences we have arrived at the correct position on both axes. Not only that, by adding *twice* the difference on the Y axis to the error term each time, we have even ensured that the one pixel change in the Y axis takes place in the middle of the line rather than at the end, so that the line looks more natural.

This straightforward method can be adapted to cover all circumstances for straight lines. For instance, the method we have just worked through is designed for lines which move positively along the X axis. For a line whose longest axis moves in a negative direction the simple solution is to swap the ends before drawing it. If the Y axis moves negatively, all that needs to be done is to subtract one, rather than add, every time the error term E exceeds zero.

The only really tricky manouevre is when the longest axis is the vertical one rather than the horizontal. Even so, it is easy enough to record the fact and remember that what is being calculated is not the position across and then the position down but, rather, the other way around. When it comes to actually plotting the position on the screen, it will be necessary only to swap the two coordinates.

The actual routines for line plotting are the most complex you will come across in this book, involving, as they do, a variety of calculations of a fairly complex nature. For that reason we will depart from the normal format of the rest of the book, and in this section enter and test the BASIC version. To speed up the execution of the BASIC routine, it is assumed that you will be using it with the high resolution machine code routines in the memory rather than the BASIC example routines entered so far. You can, if you wish, quite easily alter the BASIC program given here so that it will mesh with the BASIC routines given before. The result, however, will be slow and unlikely to be satisfactory.

Line Plotting — BASIC listing

```
1 GOTO 3
2 SAVE"@0:LINE DRAW",8 : VERIFY "0:LINE
DRAW",8 : STOP
3 REM
10 REM ROUTINE TO DRAW STRAIGHT LINES
20 POKE56,12:CLR
30 COLOUR=3072
40 BITMAP=8192
50 SYS (51044) BITMAP,COLOUR
70 SYS 51003
80 SYS 51116
90 GOSUB 2000
170 SYS 50944
```

```
180 END
1000 REM DRAW STRAIGHT LINE
1010 SWAPED = 0
1020 DX = ABS(X2-X1) : DY = ABS(Y2-Y1)
1030 IF DY>DX THEN SWAPED=-1 : T=X2:X2=Y
2:Y2=T:T=Y1:Y1=X1:X1=T : GOTO1020
1040 X=X1 : Y=Y1
1050 IF X2<X1 THEN X1=X2:X2=X:Y1=Y2:Y2=Y
 : GOTO 1040
1060 YS = SGN(Y2-Y1)
1070 E = -DX
1080 FOR X = X1 TO X2
1090 A = X : B = Y
1100 IF SWAPED THEN B = X : A = Y
1110 IF A<320 AND A>=0 AND B<199 AND B>=
0 THEN SYS (51304) 0,A,B
1120 E = E+2*DY
1130 IF E>0 THEN Y = Y+YS : E = E-2*DX
1140 NEXT
1150 RETURN
2000 REM TEST LINE DRAW
2010 FOR I1 = 10 TO 310 STEP 300
2020 FOR I2 = 10 TO 190 STEP 25
2025 X2 = I1  : Y2 = I2
2030 X1 = 160 : Y1 = 100
2040 GOSUB 1000
2050 NEXT I2,I1
2060 GET T$  : IF T$="" THEN 2060
2070 RETURN
```

Commentary

Lines 20–180: These lines set up and clear the high resolution screen using HSCREEN, HIRES and HCLEAR, then call up the line-drawing test part of the program, finally restoring the screen to low resolution.

Lines 1000–1150: The line-drawing algorithm, which will be commented upon in sections.

Lines 1010–1030: The routine will always work on the assumption that the greatest degree of movement for the line will be along the X axis. This does not mean that the movement will always be across the screen in the way we would normally define the X axis, but simply that the names of the coordinates will be swapped in order to ensure that the difference between X1 and X2 is always greater than or equal to the difference between Y1 and Y2.

Lines 1040–1050: The routine is designed to work with lines which travel in a positive direction in terms of the X axis. If a line is defined which travels backwards, the ends are simply swapped.

Line 1070: 'E' is the error term which will be used to determine whether the line being drawn has deviated enough from the correct course along the Y axis to justify moving one pixel along the Y axis. It is originally set to the difference between the two X coordinates.

Line 1080: The line drawing loop will plot as many pixels as there are positions along the X axis.

Lines 1090–1110: These lines use the HPLOT routine to plot the specified pixel on the screen. The coordinates may need to be swapped for plotting if the line drawing routine has already swapped the X and Y axes for the purposes of calculation.

Lines 1120–1130: The testing of the error term to see whether a move along the Y axis is needed yet.

Lines 2000–2050: This part of the program conducts the overall test by defining a series of lines which form a star on the screen and by calling the line drawing routine to execute each one.

Testing

Simply RUN the program. You should see the high resolution screen set up and series lines drawn which form a star with its centre roughly in the middle of the screen.

Line Plotting — assembly language listing

Having tested the algorithm for line drawing in BASIC, we can now look at the machine code. The method corresponds exactly to what you have just seen and, as far as possible, the variables used have the same names.

```
ADD.   DATA      SOURCE CODE
00                 10 PRT
00                 20 ORG $C966
C966               30 SYM
C966               70 TEMP    = $14
C966               80 XHI     = $4B
C966               90 XLO     = $4D
C966              100 YHI     = $4F
C966              110 YLO     = $51
C966              120 MODE    = $02
C966              130 SWAPED  = $53
C966              140 DX      = $5F
C966              150 DY      = $61
C966              160 X1      = $63
C966              170 X2      = $65
C966              180 Y1      = $67
C966              190 Y2      = $69
```

```
C966                    200 AVAR    = $6B
C966                    210 BVAR    = $6F
C966                    220 E       = $6D
C966                    230 ABS     = $C8B0
C966                    240 NEGATE  = $C8B4
C966                    250 ADD     = $C8C2
C966                    260 MINUS   = $C8D2
C966                    270 MOVE    = $C8E2
C966                    280 SWAP    = $C8ED
C966                    290 TEST    = $C901
C966                    300 TIMES2  = $C90E
C966                    310 WINDOW  = $C93C
C966                    320 GETWRD  = $C000
C966                    330 GETBYT  = $C006
C966                    340 COMMA   = $AEFD
C966                    370 GETMOD
C966  2006C0            380 JSR GETBYT
C969  C903              390 CMP #3
C96B  B005              400 BCS IQERR
C96D  8502              410 STA MODE
C96F  4CFDAE            420 JMP COMMA
C972                    440 IQERR
C972  20B9CA            450 JSR L007
C975  4C48B2            460 JMP $B248
C978                    490 SETWIN
C978  A900              500 LDA #0
C97A  854D              510 STA XLO
C97C  854E              520 STA XLO+1
C97E  8551              530 STA YLO
C980  8552              540 STA YLO+1
C982  8550              550 STA YHI+1
C984  A9C8              560 LDA #200
C986  854F              570 STA YHI
C988  A940              580 LDA #64
C98A  854B              590 STA XHI
C98C  A901              600 LDA #1
C98E  854C              610 STA XHI+1
C990  60                620 RTS
C991                    660 GETXY2
C991  2000C0            670 JSR GETWRD
C994  2078C9            680 JSR SETWIN
C997  A214              690 LDX #$14
C999  A04B              700 LDY #XHI
C99B  2001C9            710 JSR TEST
C99E  B0D2              720 BCS IQERR
C9A0  A515              730 LDA $15
C9A2  48                740 PHA
C9A3  A514              750 LDA $14
C9A5  48                760 PHA
C9A6  20FDAE            770 JSR COMMA
C9A9  2006C0            780 JSR GETBYT
C9AC  C9C8              790 CMP #200
```

```
C9AE  B0C2      800 BCS IQERR
C9B0  A269      810 LDX #Y2
C9B2  A014      820 LDY #$14
C9B4  20E2C8    830 JSR MOVE
C9B7  68        840 PLA
C9B8  8565      850 STA X2
C9BA  68        860 PLA
C9BB  8566      870 STA X2+1
C9BD  60        880 RTS
C9BE            910 GETXY1
C9BE  2091C9    920 JSR GETXY2
C9C1  A263      930 LDX #X1
C9C3  A065      940 LDY #X2
C9C5  20E2C8    950 JSR MOVE
C9C8  A267      960 LDX #Y1
C9CA  A069      970 LDY #Y2
C9CC  4CE2C8    980 JMP MOVE
C9CF           1010 LINE
C9CF  2066C9   1020 JSR GETMOD
C9D2  20BEC9   1030 JSR GETXY1
C9D5  20FDAE   1040 JSR COMMA
C9D8  A563     1050 LDA X1
C9DA  48       1060 PHA
C9DB  A564     1070 LDA X1+1
C9DD  48       1080 PHA
C9DE  A567     1090 LDA Y1
C9E0  48       1100 PHA
C9E1  A568     1110 LDA Y1+1
C9E3  48       1120 PHA
C9E4  2091C9   1130 JSR GETXY2
C9E7  68       1140 PLA
C9E8  8568     1150 STA Y1+1
C9EA  68       1160 PLA
C9EB  8567     1170 STA Y1
C9ED  68       1180 PLA
C9EE  8564     1190 STA X1+1
C9F0  68       1200 PLA
C9F1  8563     1210 STA X1
C9F3  2078C9   1220 JSR SETWIN
C9F6           1240 DOLINE
C9F6  A900     1250 LDA #0
C9F8  8553     1260 STA SWAPED
C9FA           1270 L000
C9FA  A25F     1290 LDX #DX
C9FC  A065     1300 LDY #X2
C9FE  20E2C8   1310 JSR MOVE
CA01  A063     1320 LDY #X1
CA03  20D2C8   1330 JSR MINUS
CA06  20B0C8   1340 JSR ABS
CA09  A261     1350 LDX #DY
CA0B  A069     1360 LDY #Y2
```

```
CA0D  20E2C8    1370 JSR MOVE
CA10  A067      1380 LDY #Y1
CA12  20D2C8    1390 JSR MINUS
CA15  20B0C8    1400 JSR ABS
CA18  A05F      1420 LDY #DX
CA1A  2001C9    1430 JSR TEST
CA1D  9017      1440 BCC L001
CA1F  F015      1450 BEQ L001
CA21  A9FF      1470 LDA #$FF
CA23  8553      1480 STA SWAPED
CA25  A265      1490 LDX #X2
CA27  A069      1500 LDY #Y2
CA29  20EDC8    1510 JSR SWAP
CA2C  A263      1520 LDX #X1
CA2E  A067      1530 LDY #Y1
CA30  20EDC8    1540 JSR SWAP
CA33  4CFAC9    1550 JMP L000
CA36            1570 L001
CA36  A265      1580 LDX #X2
CA38  A063      1590 LDY #X1
CA3A  2001C9    1600 JSR TEST
CA3D  B00A      1610 BCS L002
CA3F  20EDC8    1630 JSR SWAP
CA42  A267      1640 LDX #Y1
CA44  A069      1650 LDY #Y2
CA46  20EDC8    1660 JSR SWAP
CA49            1680 L002
CA49  A26D      1690 LDX #E
CA4B  A05F      1700 LDY #DX
CA4D  20E2C8    1710 JSR MOVE
CA50  20B4C8    1720 JSR NEGATE
CA53  A269      1740 LDX #Y2
CA55  A067      1750 LDY #Y1
CA57  20D2C8    1760 JSR MINUS
CA5A  A261      1780 LDX #DY
CA5C  200EC9    1790 JSR TIMES2
CA5F  A25F      1800 LDX #DX
CA61  200EC9    1810 JSR TIMES2
CA64            1830 LOOP
CA64  A26B      1850 LDX #AVAR
CA66  A063      1860 LDY #X1
CA68  20E2C8    1870 JSR MOVE
CA6B  A26F      1880 LDX #BVAR
CA6D  A067      1890 LDY #Y1
CA6F  20E2C8    1900 JSR MOVE
CA72  2453      1920 BIT SWAPED
CA74  1005      1930 BPL L003
CA76  A06B      1950 LDY #AVAR
CA78  20EDC8    1960 JSR SWAP
CA7B            1980 L003
CA7B  203CC9    1990 JSR WINDOW
```

```
CA7E  A26D      2010 LDX #E
CA80  A061      2020 LDY #DY
CA82  20C2C8    2030 JSR ADD
CA85  A56E      2050 LDA E+1
CA87  3021      2060 BMI L005
CA89  056D      2070 ORA E
CA8B  F01D      2080 BEQ L005
CA8D  A9FF      2100 LDA #$FF
CA8F  8514      2110 STA TEMP
CA91  8515      2120 STA TEMP+1
CA93  246A      2140 BIT Y2+1
CA95  1005      2150 BPL L004
CA97  A214      2160 LDX #TEMP
CA99  20B4C8    2170 JSR NEGATE
CA9C            2190 L004
CA9C  A267      2200 LDX #Y1
CA9E  A014      2210 LDY #TEMP
CAA0  20D2C8    2220 JSR MINUS
CAA3  A26D      2240 LDX #E
CAA5  A05F      2250 LDY #DX
CAA7  20D2C8    2260 JSR MINUS
CAAA            2280 L005
CAAA  E663      2290 INC X1
CAAC  D002      2300 BNE L006
CAAE  E664      2310 INC X1+1
CAB0            2320 L006
CAB0  A265      2330 LDX #X2
CAB2  A063      2340 LDY #X1
CAB4  2001C9    2350 JSR TEST
CAB7  B0AB      2360 BCS LOOP
CAB9            2380 L007
CAB9  A225      2390 LDX #$25
CABB  A900      2400 LDA #$00
CABD            2410 L008
CABD  954B      2420 STA $4B.X
CABF  CA        2430 DEX
CAC0  10FB      2440 BPL L008
CAC2  A94C      2450 LDA #$4C
CAC4  8554      2460 STA $54
CAC6  60        2470 RTS
CAC7            2480 END

 TOTAL ERRORS IN FILE --- 0


TEMP             14
XHI              4B
XLO              4D
YHI              4F
YLO              51
MODE             02
SWAPED           53
```

```
DX                  5F
DY                  61
X1                  63
X2                  65
Y1                  67
Y2                  69
AVAR                6B
BVAR                6F
E                   6D
ABS                 C8B0
NEGATE              C8B4
ADD                 C8C2
MINUS               C8D2
MOVE                C8E2
SWAP                C8ED
TEST                C901
TIMES2              C90E
WINDOW              C93C
GETWRD              C000
GETBYT              C006
COMMA               AEFD
GETMOD              C966
IQERR               C972
SETWIN              C978
GETXY2              C991
GETXY1              C9BE
LINE                C9CF
DOLINE              C9F6
L000                C9FA
L001                CA36
L002                CA49
LOOP                CA64
L003                CA7B
L004                CA9C
L005                CAAA
L006                CAB0
L007                CAB9
L008                CABD
 TOTAL NUMBER OF SYMBOLS --- 45
```

Machine code

```
ADD   DATA                              CHECKSUM

C966  20 06 C0 C9 03 B0 05 85    3977
C96E  02 4C FD AE 20 B9 CA 4C    4444
C976  48 B2 A9 00 85 4D 85 4E    6C54
C97E  85 51 85 52 85 50 A9 C8    7402
C986  85 4F A9 40 85 4B A9 01    7607
C98E  85 4C 60 20 00 C0 20 78    6738
```

```
C996  C9 A2 14 A0 4B 20 01 C9   9D23
C99E  B0 D2 A5 15 48 A5 14 48   A7B4
C9A6  20 FD AE 20 06 C0 C9 C8   6C8A
C9AE  B0 C2 A2 69 A0 14 20 E2   A9C2
C9B6  C8 68 85 65 68 85 66 60   9B70
C9BE  20 91 C9 A2 63 A0 65 20   5E02
C9C6  E2 C8 A2 67 A0 69 4C E2   C5CE
C9CE  C8 20 66 C9 20 BE C9 20   8AFA
C9D6  FD AE A5 63 48 A5 64 48   CAB4
C9DE  A5 67 48 A5 68 48 20 91   84C1
C9E6  C9 68 85 68 68 85 67 68   9C2A
C9EE  85 64 68 85 63 20 78 C9   7621
C9F6  A9 00 85 53 A2 5F A0 65   7281
C9FE  20 E2 C8 A0 63 20 D2 C8   7184
CA06  20 B0 C8 A2 61 A0 69 20   659A
CA0E  E2 C8 A0 67 20 D2 C8 20   C368
CA16  B0 C8 A0 5F 20 01 C9 90   A716
CA1E  17 F0 15 A9 FF 85 53 A2   6004
CA26  65 A0 69 20 ED C8 A2 63   75CF
CA2E  A0 67 20 ED C8 4C FA C9   86BD
CA36  A2 65 A0 63 20 01 C9 B0   87B6
CA3E  0A 20 ED C8 A2 67 A0 69   3F75
CA46  20 ED C8 A2 6D A0 5F 20   7526
CA4E  E2 C8 20 B4 C8 A2 69 A0   BC7A
CA56  67 20 D2 C8 A2 61 20 0E   6922
CA5E  C9 A2 5F 20 0E C9 A2 6B   A023
CA66  A0 63 20 E2 C8 A2 6F A0   8526
CA6E  67 20 E2 C8 24 53 10 05   66D1
CA76  A0 6B 20 ED C8 20 3C C9   8591
CA7E  A2 6D A0 61 20 C2 C8 A5   8C8D
CA86  6E 30 21 05 6D F0 1D A9   4F7B
CA8E  FF 85 14 85 15 24 6A 10   ADAC
CA96  05 A2 14 20 B4 C8 A2 67   39EB
CA9E  A0 14 20 D2 C8 A2 6D A0   7062
CAA6  5F 20 D2 C8 E6 63 D0 02   689E
CAAE  E6 64 A2 65 A0 63 20 01   AD5D
CAB6  C9 B0 AB A2 25 A9 00 95   B461
CABE  4B CA 10 FB A9 4C 85 54   7186
CAC6  60                        0060
```

Commentary

Lines 70–220: These are the locations for the variables that will be used by the routine, representing spare addresses in the zero page of memory. To the variables used by the BASIC routine are added TEMP, which will be a temporary variable used at certain points, and MODE, which will allow the line to be drawn or erased, as with an individual pixel earlier. As described in the previous routine, we also need parameters for the size of the screen, the variables to define this being HXI, XLO, YHI and YLO.

Lines 370–420: A subroutine to obtain the MODE parameter and test that it is in the range 0–2.

Lines 490–620: Subroutine to set the size of the screen window to 320 across by 200 down. Any lines which fall within this window will be passed by the previous window test routine.

Lines 660–880: A subroutine to pick up the coordinates of the end point of the line. The Y coordinate is tested against the limit of 0–199, since this never changes. The X coordinate is passed to the window test routine, since the width of the window may vary according to whether we are in multi-colour mode or not. If either coordinate is off the screen, an ILLEGAL QUANTITY error is generated.

Lines 910–980: A subroutine to perform the same check for the X1 and Y1 coordinates.

Lines 1000–1220: The main line drawing routine starts here. The previous subroutines are called upon to pick up the coordinates and the MODE. The resulting parameters are returned as numbers stored on the stack. They are pulled off the stack by these lines and placed into the locations specified in the table at the beginning of the routine.

Lines 1250–1260: The variable SWAPED, which indicates whether what would normally be horizontal and vertical have been exchanged for the purposes of calculation, is set to zero.

Lines 1290–1340: These lines perform the equivalent of the BASIC instruction (used in the previous BASIC line drawing routine) DX = ABS (X2 – X1)

Lines 1350–1400: Equivalent to the BASIC instruction DY = ABS (Y2 – Y1)

Lines 1420–1450: Equivalent to the BASIC line IF DX< = DY THEN...

Lines 1470–1480: Set SWAPED to – 1, or $FF, since in 16-bit two's complement format, – 1 = $FFFF.

Lines 1490–1540: Swap X1/Y1 and X2/Y2 using the previously entered subroutines.

Lines 1580–1610: Equivalent to IF X1> X2 THEN GOTO L002.

Line 1630: Swap X1 and X2 only.

Lines 1640–1660: Swap Y1 and Y2 only.

Lines 1690–1720: Equivalent to E = – DX.

Lines 1740–1760: Equivalent to Y2 = Y2 – Y1. Note that this is *not* in the original BASIC program. There is nothing subtle about this command — it is simply that there are not that many storage locations available in the early pages of the memory when we are working with machine code and BASIC. Since the variable Y2 is not going to be used for anything again, the place where it was stored is now a convenient location to store the sign of Y2 – Y1, or YS as it is called in the BASIC program. All that happens is that the two-byte address previously holding the value Y2 will be loaded with the result of Y2 – Y1. The fifteenth, or sign bit, will be a convenient indicator of the value of YS, being set if YS is – 1 and reset if YS is 1.

Lines 1780–1810: From now on, it will not be DX and DY which are required but 2*DX and 2*DY. These lines perform the multiplication in advance.

Lines 1850–1900: Equivalent to A = X1 : B = Y1

Lines 1920–1930: Equivalent to IF SWAPED> 128 THEN GOTO L003. Remember that SWAPED will be either 255 or zero according to whether the X and Y coordinate have been swapped or not.

Lines 1950–1960: These lines swap the values in A and B if necessary.

Line 1990: Calls WINDOW to test and plot the point in question.

Lines 2010–2030: Equivalent to E = E + 2*DY.

Lines 2050–2080: Equivalent to IF E< = 0 THEN GOTO L005.

Lines 2100–2120: In BASIC it is a simple matter to add to Y the value of the sign of Y2 – Y1. All we need to do is to use YS = SGN(Y2 – Y1) and later add YS. There is no such neat equivalent in machine code. To perform the same operation it is simpler to add together two 16-bit numbers, even if one of them (called TEMP in this case) is only going to be plus or minus one. These two lines place the value – 1 into TEMP.

Lines 2140–2170: The contents of the sign bit of Y2 (we are using this to store the result of Y1 – Y2 now) are transferred to the negative flag. The negative flag is now tested and, if positive, a jump is made around the two lines which send TEMP to the negate routine entered earlier, where it will

have its sign changed.

Lines 2200–2220: Equivalent to Y1 = Y1 + YS.

Lines 2240–2260: Equivalent to E = E – 2*DX.

Lines 2290–2360: Equivalent to NEXT X, but notice that to reduce the number of variables we do not bother with a new variable X, we simply use X1 as the loop variable. Equally we could have had a loop FOR X1 = X1 TO X2 in the BASIC program.

Lines 2390–2460: These lines play no part in the actual line drawing. They are needed because this routine has used up so many of the spare locations available for storage when BASIC is running, that we are actually using some locations which will be needed by the BASIC interpreter when our machine code routine returns control to the 64's ROM. These have to be tidied up.

Testing

Take the BASIC line drawing program that you entered earlier and alter it so that the whole section from line 1010 to line 1140 is replaced with:

1010 SYS (51663) 2,X1,Y1,X2,Y2

and lines 2010 and 2020 to:

2010 FOR I1 = 10 TO 310 STEP 20
2020 FOR I2 = 10 TO 190 STEP 20

The result should be a far more complex star pattern which would have taken the earlier program an eternity to print. Do not worry if some of the lines appear to have gaps in them — they are being drawn in inverse mode and they erase each other where they brush.

Syntax

The syntax for LINE is:

SYS (51663) { MODE} , { START POSITION X} , { START POSITION Y} ,{ END POSITION X} ,{ END POSITION Y}

MODE is defined as for the PLOT routine. X must be in the range 0–319 and Y in the range 0–199.

8. HIGH RESOLUTION CIRCLES (CIRCLE)

Having seen the complexity of the line drawing commands, you may not be surprised to discover that a command to draw a circle is also a relatively