



Hi - Speed Windows

A series for beginners By Richard Sargent

The way in which computers organise their screen layout is by no means simple, and no standard screen layout or operating technique has developed during the short evolution of the home micro. The Amstrad screen is, of necessity, complicated: the computer has to be capable of producing an 80-column screen for business and programming, a multi-colour screen for games and a general purpose screen for everyday use. The computer has three screen modes to help it with these tasks, and there are some things that can be done in one mode but not in another. The reason why the screen display on home micros is the victim of compromise is quite simple - there is insufficient memory (and often insufficient processing speed) to produce the ideal screen which would have high-resolution graphics combined with 256 shades of colour and fast text scrolling into the bargain.

However, if you understand the peculiarities of your micro's screen, you will be able to use it to the full. This and subsequent articles will show you how to manipulate the Amstrad screen in Basic and, more importantly, will demonstrate machine-code routines which will develop screen displays which Basic would be too slow to achieve.

The Basic Screen:

Mode 1

The Amstrad screen occupies 84000 or 16K of memory, but is regarded by Basic as a piece of graph paper divided into tiny squares. There are 640 of these along the horizontal or X axis and 400 along the vertical or Y axis. On some computers these squares represent the actual screen pixels. This area is the graphics screen, and its origin is 0,0 at the bottom left-hand corner of the screen. This graph paper may also be ruled off into larger squares, the character blocks, of which there are 40 along the horizontal axis, or columns, and 20 along the vertical axis, or rows. The origin of this text screen is the top left-hand corner of the screen and it is 1,1. On the text screen, coordinates 0,0 are illegal.

Windows

Although graphics and text share the same physical RAM memory and the same screen (or graph-paper), different ways of labelling that graph-paper are possible and can exist simultaneously. For the text screen, these labelling systems

are known as windows, and you can create as many as eight of them at any one time. When you first switch on the computer or give MODE 1 command, WINDOW 0 is automatically created to the dimensions 1,40,1,25, which is the full dimension of the screen (1 to 40 columns wide and 1 to 25 rows deep). The computer manual (Ch.5, p.10 for the 484 and Ch.5, p.26 for the 6128) explains WINDOWS and the examples given there should be studied carefully. The main point to be aware of is that data can be directed to a specific window and that a window can be scrolled, cleared and used to print INPUT prompts (as is though it were a mini-screen in its own right).

The window has its own coordinate system, so that the top-left of a window is always accessible by LOCATE# n,1,1, no matter where the window is positioned on the screen. The windows are numbered 0 to 7 and the default size for them all is 1,40,1,25. Since this covers the entire screen, the windows are not particularly noticeable until they are set down in size. Changing the dimensions of a window is achieved simply by issuing a new set of coordinates by telling the computer whereabouts (with reference to the whole screen) the new edges of the window are to be.

Thus the command WINDOW# 7,38,40,23,25 creates a small window in the bottom right-hand corner of the screen and thereafter LOCATE# 3,1,1 and LOCATE 38,23 both point to precisely the same physical point on the screen. LOCATE# 4,28,23 if the window you wanted, the 0 may be omitted. Overlapping windows are permitted. Program listing and the 'Ready' prompt appear in WINDOW 0 so it is inadvisable to set it to small dimensions. Consult one of the manuals which advise that before you start experimenting you should set a spare key (such as 0 on the numeric pad) in the following manner:

```
BY 00,7000 Y 004 0,1,20 1,25 PAPER BACK 0,1,25-0000 00
```

If you create small windows or strange colour combinations you will find it difficult to read (and therefore check) commands that you give the computer. By pressing key-pad 0, or 0 on the 6128, normally will be restored.

Printing messages, prompt, and calculation results onto screen windows rather than onto the screen as a whole is a sensible way to use a computer and the visual appearance of any program is likely to be enhanced by the use of just a few windows. Technically, windows are important because they establish the screen memory addresses and this is of special importance when it comes to a machine code programming because it makes the task of writing code that much more simple. A window can set aside a portion of screen which Basic does not use and so prepares the way for a machine code routine to handle that part of the screen memory directly.

The windows share the screen with each other, but they also share the screen with the graph paper. It is essential to understand how the graph's screen windows interact, but now that windows are no longer a mystery, a short program can illustrate the workings of the graphics screen.

Type in Listing 1 exactly as it is shown, with the BASIC keywords 'protecting' certain instructions. With the listing in the computer (and saved), type MODE 1 directly, followed by RUN. A triangle is partly drawn and the screen coordinates of the various points of the triangle are printed, causing both the text and the graphics to scroll upwards. Two more RUNs make the screen very messy. Now LIST and retype line 11 without its REM keyword. Type MODE 1 and RUN directly and observe the result. The text scrolls but

the triangle stays where it is. Further RUNs do not affect the position of the triangle, because the WINDOW statement of line 130, now active, has confined the text to the left-hand side of the screen. CLS, acting on the window, will clear the text but not the graphics. CLG will clear the graphics screen, which covers the whole screen so the triangle and the text will both disappear.

Listing 1

```
100 REM LISTING 1 (PT.1)
110 REM WINDOW 1,38,1,25
120 LOCATE 1,24
130 ORIGIN 400,700
140 TAB:PRINT "0,0";:TAB(1)
150 PLOT 0,0
160 PRINT SP0;:PPO
170 DRAW 100,100
180 PRINT SP0;:PPO
190 DRAW 0,-100
200 PRINT SP0;:PPO
210 DRAW -50,-50
220 PRINT SP0;:PPO
230 STOP
```

If you want the graphics screen to be a different colour from the standard blue background, then CLG 3 will set the graphics paper to ink number three, which is normally red. In fact, you can think of the graphics screen as yet another window. It has no channel number attached to it, and it is permanently set to full-screen size. How then, do you push the graphics into just one part of the screen? Normally, the base coordinates of the graphics screen are X=0, Y=0, it is to the bottom pixel on the left-hand side of the screen. The triangle on the right of the screen is occupying pixel positions in the X=450 Y=250 area of the screen, yet the printed reports given by XPOS and YPOS are much lower figures. What has happened is that the program has set a new base for the graphics screen. Line 130 really means 'Take position 400,700 of the whole screen and make it position 0,0 of a new graphics screen'. All subsequent PLOTS, MOVES and DRAWS take place with reference to the new origin. With coordinate 0,0 in their new position, it is possible to plot using minus X and minus Y values, as the program illustrates, but positive values keep the graphics over in the right-hand side of the screen.

Window Design

Listing 2 is a Basic program which demonstrates the way in which windows can be used to enhance a screen display. Figure 1 is the screen dump of the program which illustrates the OEM-like qualities which can be achieved even on the MODE 1 Amstrad screen. For the record, WINDOW 0 is the whole screen, WINDOWS 1 to 4 are the four shadow-outline boxes, WINDOW 5 is the 'Hello World' box, WINDOW 6 is, by design, invisible and carries the INPUT prompts. The remaining window is 7 which carries the headings of the 'pull-down' menus. The program is a demonstration only, but is nevertheless instructive. The information for the right windows is carried in the DATA statements of lines 1380-1430 and is later transferred to an array and also the variable W,L,N and S. The relationship for window 0 is shown here:

FEATURE

1300 DATA

```

2 : a(1) : Paper in ink 2
1 : a(2) : Pen in ink 1
1 : a(3) : W : west edge col.1
40 : a(4) : E : east edge col.40
2 : a(5) : N : north edge row 2
25 : a(6) : S : south edge row 25
  
```

The border colour, ink colour and height of the window at the top of the screen (which carries the names of the pull-down menus) all find their way into the Zin array which carries a total of 54 pieces of information. If you don't want to type in the whole program, you might like to try the key subroutine at lines 1320 - 1340 and 1520 - 1550. The former routine places the window (and its shadow) on the screen, while the latter forms the special symbols required for the shadowing, and the symbols needed for the large arrow (DOWN) and the hand (HAND). The few lines of Basic needed to run the two subroutines are as follows:

```

10 GOSUB 1520
12 INK 0,1:INK 1,24:PAPER 0:PEN 1
14 MOVE 1:DIM I(55):I(14)=0:P=0
15 LOCATE 1,1:PRINT SPACES (24)
16 PRINT SPACES(24):LOCATE 1,1
17 INPUT @,E,S,N,~:@,E,S,N,S
20 IF @=40 OR @=25 THEN GOTO 15
22 GOSUB 1330:GOTO 15
  
```

Listing 2

```

100 REM LISTING 2
110 REM WINDOWING SUBROUTINE
112 REM FILE
115 REM SCREEN WINDOW INITIALIZED EVERYTHING
120 :
130 REM MAIN MENU PROGRAM STARTS HERE
150 :
160 W(1)=1040:W(2)=1040:W(3)=1041:W(4)
  1=1040:GOSUB 450:W(1)=1000:W(2)
170 INPUT @," WEST MENU":@
180 IF @="?" OR @="?" THEN GOTO 150
190 IF @="?" OR @="?" THEN WINDOW 1,40
  1,25:INK 0,1:PAPER 0:INK 1,24:PEN 1
  :CLEAR
200 @="?" THEN LOCATE 1,1:GOTO 140
210 @=1:W(1)=1000:W(2)=1000:W(3)=1000:W(4)
  1=1040
220 @=2:W(1)=1000:W(2)=1000:W(3)=1000:W(4)
  1=1040
230 INPUT @," WHICH MENU":@
240 W(1)=1040:W(2)=1041:W(3)=1040:W(4)
  1=1040
250 W(1)=1000:W(2)
260 IF @=1 THEN GOTO 140
270 IF @=2 THEN GOTO 150
280 IF @=3 THEN GOTO 160
290 IF @=4 THEN GOTO 170
300 :
  
```

```

310 IF @=3 THEN GOTO 180
320 GOTO 140
330 :
340 REM END OF MAIN PROGRAM
350 :
360 MOVE 1
370 REM HEAD SELECTED WINDOW DATA
375 LOCATE 1,10
380 W(1)=10:W(2)
390 W(1)=10:W(2)=100:W(3)=100:W(4)=100:W(5)
  1=10:W(6)
400 WINDOW 1:W(1)
410 @="?" MENU :
420 @="?" DRAWING :
430 @="?" DRAWING :
440 @="?" DRAWING :
450 GOTO 150
460 WINDOW 1:W(1):W(2):W(3):W(4):W(5):W(6)
470 WINDOW @,W(1):W(2):W(3):W(4):W(5)
480 WINDOW @,W(1):W(2):W(3):W(4):W(5)
490 W(1)=1040:W(2)=1041:W(3)=1040:W(4)
  1=1040:GOSUB 450
500 @=1:GOTO 160:W(1)=1000:W(2)=1000:W(3)
  1=1000:W(4)
510 RETURN
520 REM END OF INITIALIZATION
530 :
540 :
550 REM WINDOWING SUBROUTINE
560 REM SET TOP WINDOW COLOUR NORMAL OR
  WINDOW,INK
570 PAPER @,W(1):W(2):W(3):W(4):W(5):W(6)
  @:
580 PEN @,W(1):W(2):W(3):W(4):W(5)
590 PEN @,W(1):W(2):W(3):W(4):W(5)
600 RETURN
700 :
710 REM DRAW WINDOW 1
720 W(1)=10:W(2)=100:W(3)=100:W(4)=100:W(5)
  1=10:W(6)
730 REM PRINTED PROMPTS
740 LOCATE @,1:PRINT @," L. LEAD":P=0
  1 @
750 PRINT @," S SAVE":PRINT @
760 PRINT @," B BACK":PRINT @
770 PRINT @," E EDIT":PRINT @
780 PRINT @," F FRESH":PRINT @:W(1)=10
  :PRINT @:PRINT @," L,S,B,E,F B "
790 RETURN
800 :
810 REM HEAD WINDOW 2
820 W(1)=10:W(2)=100:W(3)=100:W(4)=100:W(5)
  1=10:W(6)
830 REM SCROLLING PATTERNS
840 @="?" TO 0
850 @="?" TO 255
860 PRINT @,~:W(1):W(2):W(3):W(4):W(5)
  1=10:W(6)
870 RETURN
880 :
  
```