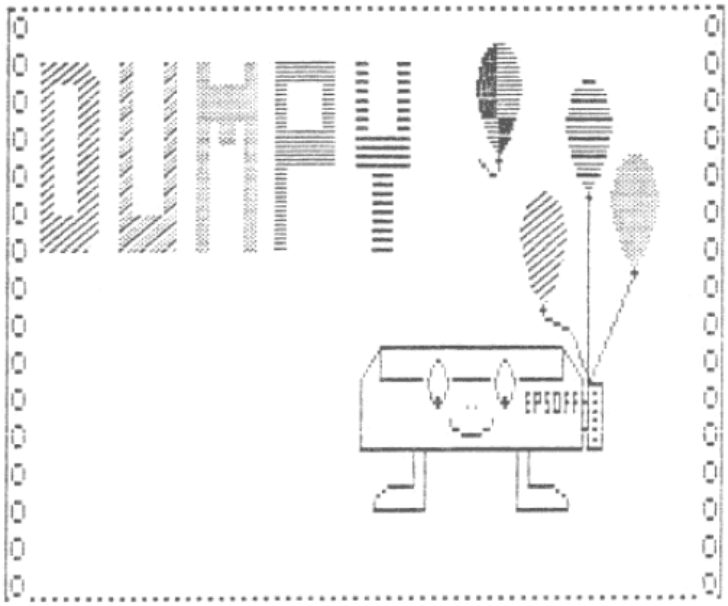


DUMPY Instruction manual

DUMPY is a program which has been designed to help mere mortals to create machine code screen dump routines, normally only the province of assembler freaks! The program is extremely easy to use and ever so versatile. By replying to a few questions and directions on-screen, you can create customised routines from a short 100 byte quicky to do a full screen copy in just a few seconds, to a rather longer routine which will produce poster sized output with the colours represented as different patterns of shading.

All the routines may be fitted wherever you require in memory - if there is space, of course - and DUMPY carries out a comprehensive series of error checks to warn of potentially disastrous combinations.

Many hours work have gone into this software. I hope you find it a useful and friendly program. Please don't just give my work away to all your friends - let them buy their own copy!

Firstly, let us look at the program, option by option. When you initially run DUMPY, all the options are reset and you will be asked first to input a value, and then to confirm it by pressing the ENTER key. If you re-run the program, the previously entered values are retained and only the ENTER key need be pressed to confirm the old value.

SCREEN 1.

Interfaces; DUMPY supports a selection of 13 different interfaces of both serial and parallel persuasions. Most of the codes supplied will fully initialise the interface ready for use, the only exceptions are the RS232 types (Interface 1, the Wafadrive and ZX Lprint III in serial mode), where the baud rate should be set from BASIC. There is no need to open a stream for Interface 1 unless you also need to LLIST from within the BASIC program for which you are creating your DUMPY code; DUMPY does the equivalent of an OPEN #3;"b" command the first time a screen dump is done.

The Opus Discovery disc drive centronics port is supported. As DUMPY uses the IX register, which is corrupted by the Opus 'print a character' hook code, entry is made directly into the Opus ROM. To do this, DUMPY actually looks at the ROM when configuring your screen dump code, so the Opus must be connected when you are using DUMPY. I don't know if there are different versions of the Discovery ROM; if there are, it is possible that a screen dump created on one drive might not work on another.

SCREEN 2.

Height; Three different heights of output are provided; input 1 for the single height, each pixel represented by one dot on the paper, or 2 or 4 for the double and quadruple height copies. Both these latter routines take up rather more memory and operate more slowly.

Width; The horizontal size can be set between 1 and 9 times magnification. Some printers (Brother M1009, Mannesman Tally Spirit 80, and probably others) will not support a value greater than 6 in this parameter. There is no way to tell whether or not your printer allows larger values, except by experiment. You will do no harm -all the routines which DUMPY produces have the break key enabled; holding down caps shift and space will stop the computer sending data at the end of the current line, and you just turn your printer off if it is feeding paper out uncontrollably.

Attributes; Input a value 0 to get a simple black and white copy or 1 for shaded mode, where the colours are represented by different stippled effects. This routine prints what you see e.g. yellow is the same pattern whether it is a paper or an ink attribute on the screen.

Density; This switches the display from single (usually about 480 dots/line) to double (960 dots/line) or quadruple density (1920 dots/line) graphics. Not all printers support the full range; check your printer manual if in doubt. The higher the density, the narrower the copy, so increase the width value to compensate. Of those printers which do have a quadruple density mode, there are two distinct control codes, 76 and 122. You will be asked which your printer uses only if you select the quad density mode.

Tab; This function controls the start position of the design on the paper, the left margin. Column numbers of up to 120 are accepted to cater for 132 column printers. Any data which has not been printed when the print head reaches its rightmost position will just be ignored, so the picture is cropped from the right hand side.

Program Linefeed; Many people have their printers set up to automatically linefeed when a carriage return character is received. Such systems require the program linefeeds switched off; input 0 to accomplish this. Input 10 to cause the program to send a CHR$ 10 linefeed code after each CHR$ 13 carriage return code.

SCREEN 3.

Top Line; This parameter is the first line of the screen that will be sent to the printer. The nomenclature is the same as the Spectrum print lines; 0 at the top and 23 at the bottom. If you want your DUMPY code to automatically find the first line with printing on each time it is called, then input 99 here.

Bottom Line; The line after the last one to be printed. This Parameter may take a value between 1 and 24 -YES- the whole screen may be printed, including the bottom reserved lines. Enter 99 for auto setting of the bottom line.

Starting Column; This is the first screen column to be printed and may take any value between 0 and 30. This parameter allows you to selectively crop the picture from the left hand side which, in conjunction with the extra wide print options enables posters up to about 36 inches wide to be produced and assembled by pasting the sheets of paper together.

Start Address; The address in memory where you want the code to start. Start addresses within the ROM or systems variables are not allowed and values within the screen display are not recommended.

End Address; You might prefer to stipulate the end address if, for example, you want a routine to join onto the bottom of some existing machine code. DUMPY will not accept both start and end addresses for a single routine as this puts impossible constraints on the code length.

Troublesome Printers; I wish you could all answer 0 to this, but unfortunately the term "Epson compatible" has been interpreted very liberally by some printer manufacturers. The list displayed is not exhaustive but reflects the ones that can easily be coped with.

SCREEN 4.

This is the end of the input section of DUMPY. You are now presented with a summary of the options you have requested and are asked to confirm them. If you have second thoughts, answer anything other than 'y' and you will be returned to the beginning of the program, but with the options you selected the first time around still intact, ready for acceptance or replacement.

SCREEN 5.

Next: The screen you don’t want to see. A series of potentially problematical and even disastrous situations is tested for and you are warned if an error is detected. The following checks are made:

- A routine longer than 256 bytes in the printer buffer area: Code ending higher than the physical RAMTOP: Code sitting in a position in memory where the machine stack would be corrupted and the computer crash. These routines should be aborted.

- Code sitting in the BASIC program area. This code can be produced, configured for your required address, but will be saved from 44000. For example, if you request a routine at 30000 DUMPY will inform you that this lies in the BASIC area. If you proceed, the code will be saved from address 44000. To run the code, clear out the program, type LOAD <filename> CODE 30000, then RAND USR 30000 will work.

- Code sitting just above the program area. This may easily be corrupted if saved to microdrive or wafadrive as in both cases the program is moved up in memory to create a buffer area, possibly over-writing some of the code you have created. Saving to tape is permissible as no buffer area is involved.

- Code at an address occupied by the DUMPY code library. This will proceed satisfactorily, but will corrupt the library so you must reload DUMPY before creating any more routines.

- Code sitting in the DUMPY workspace area. This causes no problem as the workspace is moved elsewhere before the code is relocated.

The warnings are given before any relocation of code so the run may be aborted without harm. All the warnings you receive are only advisory; you may still proceed, even with the lethal errors, but at your peril. I also apologise for any circumstances I have not forseen!

SCREEN 6.

If all is well there will be a short delay whilst the code is put together and relocated to the desired address and the final options are presented. The total length, and the start and finish addresses of the code are displayed for checking and you have the option to test the code, abort and re-run, quit the program or save the finished code to whatever storage system you have. To merge it with an existing program, just ensure that the code is loaded in with the program; any subsequent command to RANDOMISE USER <startaddress> will produce your screen dump. What could be simpler?

How DUMPY Works.

DUMPY is able to produce such a large variety of outputs by two strategies. Firstly, it has a series of subroutines in a machine code library and combines the required ones into a single block of code. All the possible combinations are shown in the diagram below; not all of these will be used in any one program, of course.

VARS is always present, followed by the code for your interface. TOPLINE and BOTLINE are only present when the auto option has been selected for their respective lines. Next comes the relevant print routine and finally, if the shaded attribute has been selected, the colour subroutine.

```
       I/F CODES                          1*HEIGHT PRINT
VARS - I/F CODES - TOPLINE - BOTLINE - 2*HEIGHT PRINT - COLOUR
       I/F CODES                          4*HEIGHT PRINT
```

Once the components of a particular routine have been assembled, the only way to alter it is by going back to DUMPY and starting again.

The second strategy is the use of a block of program variables which are read by the different subroutines. Some of these are only for use by the program and must not be altered. Others may profitably be altered by POKEing from BASIC. These variables are;

Location	Function	
+8	TLIN	Screen line from which copy starts.
+9	BLIN	Screen line at which copy ends.
+10	TAB	Printer column for copy start.
+11	WID	Width of copy.
+12	DENS	Density of copy.
+14,15	COLAD	Address of data for colour shading.
+16	SCOL	Screen column from which copy starts.

The location of the variables is given relative to the beginning of the machine code i.e. if the routine starts at 50000, then TLIN is at 50008. Note that DUMPY checks for sensible values of these parameters when configuring the machine code, but no such checks are made at run-time. It is your responsibility not to POKE anything silly into the variables! e.g. a bottom line value higher on the screen than the top line will result in up to the entire RAM contents being interpreted as a screen display and printed. This may be pretty, especially with shaded attributes, but is hardly useful.

TLIN, BLIN; These may be POKEd to alter the area of the screen to be copied. Please note that if entered directly, the top line is 24, the bottom is 0, in contrast to the normal convention. i.e. you should POKE the value with 24 minus the required line number. Similarly, BLIN is 24 minus the number of the first line you do not want printed. If the auto option was set up for either of these parameters, POKEing will have no effect as the auto machine code will reset them.

Incidentally, if you wish to auto set BLIN, but excluding the reserved bottom lines, POKE 42601 with 3. This is useful in a program such as Masterfile or Trans Express where you want to print the data from the screen without blank areas or the status lines. The POKEs alter the library; re-load DUMPY or POKE the same address with 1 to reset to full screen.

SCOL; Also alters the area of screen to be copied by specifying the column position on the screen from which printing starts. You could incorporate this into a program to print out posters; e.g. at 8* width, each line on the printer would cover 10 screen columns, so SCOL needs to be increased by 10 until the whole screen has been dumped. In this case, you would have 4 sheets to glue together.

TAB; Allows you to alter the copy position on the paper. This could be useful for printing those Prestel mailboxes; arrange your VTX enhancement program to print half the frames, wait for you to wind the paper back (unless you have a posh printer with reverse linefeed!) and print the others on the right hand half of the paper.

WIDTH; Alters the number of times each data byte is sent to the printer; i.e. the copy width. If you specify more than the printer is able to print on one line, the excess will be ignored.

DENS; You could alter the copy density by POKEing the printer command code for the different modes; ie ESC..75..n..n is single density graphics on most printers, so POKE DENS with 75. 76 is double density, 122 or 90 is quad density. There is considerable INcompatibility in the quad density implementation; see your printer manual for details.

COLAD; This 2 byte variable may be PEEKed to get the address (least significant byte first) of the start of the colour specifiers; thus the address is PEEK (start+14) + 256* PEEK (start +15). The colour data are stored at this address in order of their codes e.g. black = 0 and located at address+0, yellow is at address+6. Think of these as bit patterns used to mask off the ink and paper data and POKE them as BINary numbers. The diagonal pattern is produced
by rotating the first four patterns at each call to the routine.

This is perhaps best explained by an example: The data for green is 170; BIN 10101010, so green printing on the screen will be in fine horizontal lines. The data for magenta is also 170, but code for this colour is less than 4 so it is rotated and the lines become diagonal;

```
          11111111                10101010
          00000000                01010101      ^
GREEN     11111111       MAGENTA  10101010
          00000000                01010101      ^
          11111111                10101010   rotate
          00000000                01010110      ^
          11111111                10101010
          00000000                01010101      ^
```

If you want to alter the colour attributes for all your copies, the start address of the data in the DUMPY code library is 42553.

ROM Based Interfaces.

Both the Kempston E and ZX LPRINT III interfaces normally use the Spectrum printer buffer at 23296 - 23551 as a scratchpad area when the tokenisation is switched off. However, the driver codes used by DUMPY address the interfaces directly so that no use is made of this area of j memory. Consequently, even with these interfaces, a DUMPY screendump may be located in the printer buffer. In addition, after the first screen dump you will be able to LPRINT from BASIC without harm. Do not attempt to issue commands to these interfaces, especially turning tokenisation on or off, or your screen dump routine will be corrupted.

Back-up Copies.

Please see the paragraph on page 1. To make back-up copies or transfer DUMPY onto your microdrive, Wafadrive or disc drive you should SAVE etc. "dumpyx.x" LINE 9000: SAVE etc. "dumpycx.x" CODE 39740, 3405, where x.x is the current version number (see the REM statement in line 1). You will also have to alter the LOAD syntax in line 9000 for correct operation from the Opus Discovery drive.

Bradway Software is still an extremely small software house and still very user friendly. If you have any problems with a copy of DUMPY that you have bought, or any suggestions for improvement, or even if you find any bugs, please get in touch with Richard Walker by phoning 0742 350225 at any reasonable time (but best chance at the weekend), by letter (SAE please), or via Prestel mailbox number 742350225.

Bradway Software,
33, Conalan Avenue,
Sheffield,
S17 4PG
England.

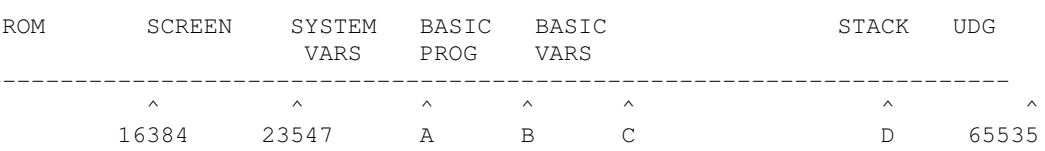
DUMPING for the Uninitiated!

IF you understand what addresses are, where programs live in memory and what RAMTOP is
THEN use these pages to light your barbeque
ELSE read on....

All information in computers, whether it is the controlling program which acts as the BASIC interpreter, or a program written in BASIC or machine code, or data belonging to a program or to tell the computer the shape of the letters it prints on the screen is measured in small aliquots known as bytes. Your Spectrum can contain 65536 of these bytes. This number is usually abbreviated to 64k, where 1k is 1024 bytes. As 16k are allocated to the ROM - the Read Only Memory that controls the overall functioning of the machine-, there are 48k of RAM (Random Access Memory) left for you to play with.

You may think of these 48k as 49152 containers all in a line, each of which can contain 1 item of information, 1 byte. The computer needs to be able to put values into these locations, and then read them at a later time and it does this by the simplest possible method of numbering the locations from 0 to 65535. These numbers are referred to as addresses. Thus, if the machine stores the value of the variable X at address 40282, then whenever the BASIC program needs to know what X is, the computer looks at address 40282 again. Very simple!

Some of the addresses in the Spectrum memory are available for you to use at will, others are reserved for use by the computer only. For example, addresses 0-16127 are occupied by the Spectrum ROM; 16384-23295 hold the contents of the screen display; a BASIC program starts at about 23816 and extends upwards in memory for as far as the program is long, and is followed immediately by the values of the BASIC variables- the X's and a$'s in your program. This may be represented diagrammatically and is called the memory map:



A: PRINT PEEK 23635 + 256 * PEEK 23636
B: PRINT PEEK 23627 + 256 * PEEK 23628
C: PRINT PEEK 23653 + 256 * PEEK 23654
D: PRINT PEEK 23730 + 256 * PEEK 23731

The formulae show how to find out the addresses of A,B,C and D. The area Between C and D is available for your machine code programs with the proviso that the item shown as the stack is used by the computer to store numbers as it is calculating and working. It is usually only a few bytes in length (unless you are using a lot of GOSUBs), but care must be taken not to corrupt it or else the computer will crash.

O.K. So now you understand a little bit about the Spectrum RAM. Lets see how this helps you to use your DUKPY screendumps. DUMPY produces a small block of machine code which you have to put at a sensible place in the RAM and then RANDOMIZE USR the start address to produce your screen dump. If you are programming in BASIC and want to include a dump facility then proceed as follows;

1) Reset the computer- use the reset button on the Spectrum Plus, or switch the power off and back on again.

2) Load in the BASIC program you are writing or using.

3) With the formulae above, find out where C and D are; this is the spare memory space that you can use.

4) At this point you have to decide where to locate your screendump. Load in DUMPY and make a screendump to your requirements by following the menus. When you come to the place where you have to specify the address of the program, you have several options:

4a) It must start higher than point C and end well away from point D. If you have plenty of room, then just bung it somewhere in the middle- perhaps at 55000 or thereabouts. Save the resultant code to tape, microdrive etc., calling the routine, for example "scrndump". Make sure that your BASIC program will load the code using the line;

```
LOAD "scrndump" CODE
```

When you want your BASIC program to do a screen copy, do not use the BASIC keyword COPY, which is only for the ZX printer, (and some special interfaces). Instead use the command;

```
RANDOMIZE USR nnnn
```

where nnnn is the address at which you asked for the screen dump code to start.

4b) Alternatively, you can make a special, protected area for the screendump by reserving a section of RAM for it above point D (you may see D referred to as RAMTOP) by actually moving RAMTOP using the CLEAR instruction e.g. reserve 400 bytes -larger than the longest DUMPY screendump- by finding point D, subtracting 401 from this and then including the instruction CLEAR nnnn in your BASIC program. Thus, if D was at 65367 (the usual address if no other machine code routines are in use), then your BASIC program would be;

```
CLEAR 64966 : REM 65357-401
LOAD "scrndump" CODE
:
:
RANDOMIZE USR 64967 : REM to do the screendump
```

Your screendump should have been created to start at 64967, You must execute the CLEAR instruction to reserve the area of memory before loading in the dump code, and do the load before the BASIC program does anything else as the CLEAR instruction will also delete any of the program variables.

4c) If there is no room left in your memory, perhaps because you have large data requirements whilst using a program such as Masterfile, there is a third place you could put the code. A 256 byte area at 23296-23551 is reserved for use by the Sinclair ZX Printer. As you are not using this, then it becomes a convenient place to locate your code; specify a start address of 23296. Your BASIC lines could be;

```
LOAD "scrndump" CODE
:
:
RANDOMIZE USR 23296
```

Only the plain, not the shaded screendumps, are short enough to fit in this space, however. The Opus disc drive is also reputed to use this area.

If you have a program such as one of the art packages that saves a SCREEN$ to tape or microdrive, then you could print copies of the screen by writing a small program such as;

```
CLEAR nnnn-1 : REM if necessary- see 4b.
LOAD "scrndump" CODE
INPUT "Type name of picture";p$
LOAD p$ SCREEN$
RANDOMIZE USR nnnn : REM print the screen
```

Sizes.

The Spectrum screen is 32 characters = 256 pixels across. The Epson standard graphics mode is 60 dots/inch = 480 dots in one 8" carriage width. This is why the single density, single width copy covers only about 256/480 *8 = 4¼ inches width. If you double the width of the screendump to 2 times, this has the effect of sending each byte of data twice, giving 512 dots, 32 more than will fit on the printer width. That is why the last two columns of the screen display are cut off.

Many screen dump programs do not allow you to specify width and density separately; instead they automatically compensate for the decreased width of a double density copy. DUMPY gives you full control over both parameters, which allows you to increase the print to double density (i.e. packed in at 120 dots/inch), giving the equivalent of 960 dots per printer width. If you leave the dump width at 1 times, the copy shrinks to half its original width. However, increasing the width to 3 times causes the program to send 256*3 = 768 bytes of data, which more nearly fills the printer without losing any columns from the screen display. The simple rule is; double the density, same number of bytes = half the width.

The Epson 'standard' is 72 dots/inch vertically. Don't ask me why this is different from the horizontal density, but the net effect on an ordinary screendump is a vertical elongation of the image. The combination of the double density, 3 times wide specification with a double height gives a more nearly linear representation of the screen, the distortion being reduced to 10% instead of the previous 17%.

These calculations do not apply to some printers such as the Mannesman Tally Spirit 80 and the Shinwa CP80, which use slightly different graphics densities.

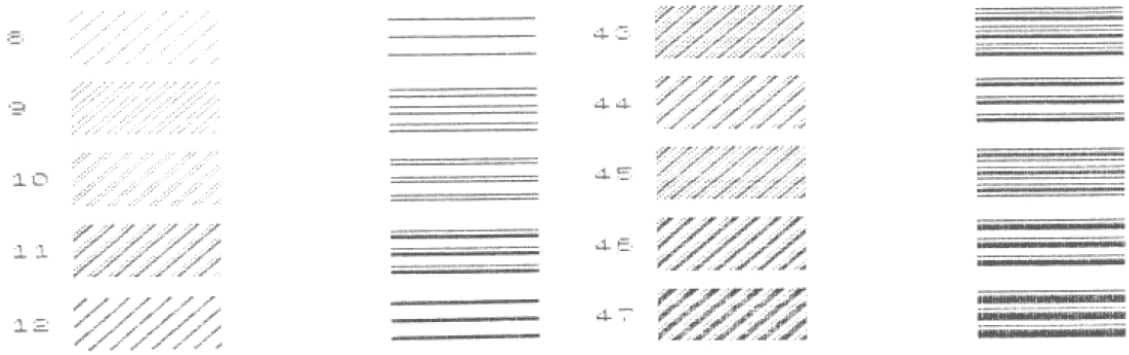
Colours

Here is a worked example of how to alter the way in which colours are represented by shading in DUMPY.

1). Make up a screen dump starting at 50000 with height 1, density 1, shading 1, top line 0, bottom line 12. Save it with the name "dtest". You will find that if you PRINT PEEK 50014 + 256*PEEK 50015, you will get an address, COLAD, somewhere around 50260, the precise value depending upon which interface you are using. This is the location of the shading data for black, colour 0 (see your keyboard). PEEK this address, you will find the value 255, which means all black. The next byte has the data for colour 1 (blue), the next for colour 2 (red) etc., so you can alter the representations by POKEing (COLAD + colour number) with the data; e.g. POKE (COLAD + 4),nn alters the green data.

2). If you didn't understand the technical explanation in the DUMPY manual, just accept that the colours 0-3 are diagonal lines and 4-7 are horizontal lines. That's the way the program works and can't be altered.

3). Now for the number to POKE into the address you've found. Some examples of the data values are shown below, along with the pattern they produce when POKEd into one of the colours 0-3 (left column) and 4-7 (right column).



If you want to find out what all the possible combinations are, then type in the following program. It uses the screen dump you created in step 1 above and prints out the 256 shading representations, many of which repeat several times. Please note that it needs about 10 sheets of paper to do this!

```
  5 LOAD *"m";1;"dtest" CODE
 10 FOR n=0 TO 2
 20 FOR y=3 TO 9
 50 PRINT PAPER n+1;AT 4*n,y;" ";AT 4*n+1,y;" ";AT 4*n+2,y;" "
 60 PRINT PAPER n+4;AT 4*n,y+15;" ";AT 4*n+1,y+15;" ";AT 4*n+2,y+15;" "
 70 NEXT y
 80 NEXT n
 85 LET colad=PEEK 50014+256*PEEK 50015
 90 POKE colad,255: POKE colad+7,0
100 FOR n=0 TO 255 STEP 3
110 POKE colad+1,n: POKE colad+4,n
120 POKE colad+2,n+1: POKE colad+5,n+1
130 POKE colad+3,n+2: POKE colad+6,n+2
140 PRINT AT 1,0;n;AT 5,0;n+1;AT 9,0;n+2
145 RANDOMIZE USR 50000
150 NEXT n
```

I hope some of these suggestions make your initial attempts at using the screendumps from DUMPY less traumatic and enable you to get going until you understand the system more fully. The memory map is explained in more detail in the original Spectrum manual although, unfortunately, this essential information was deleted from the useless Spectrum+ manual.