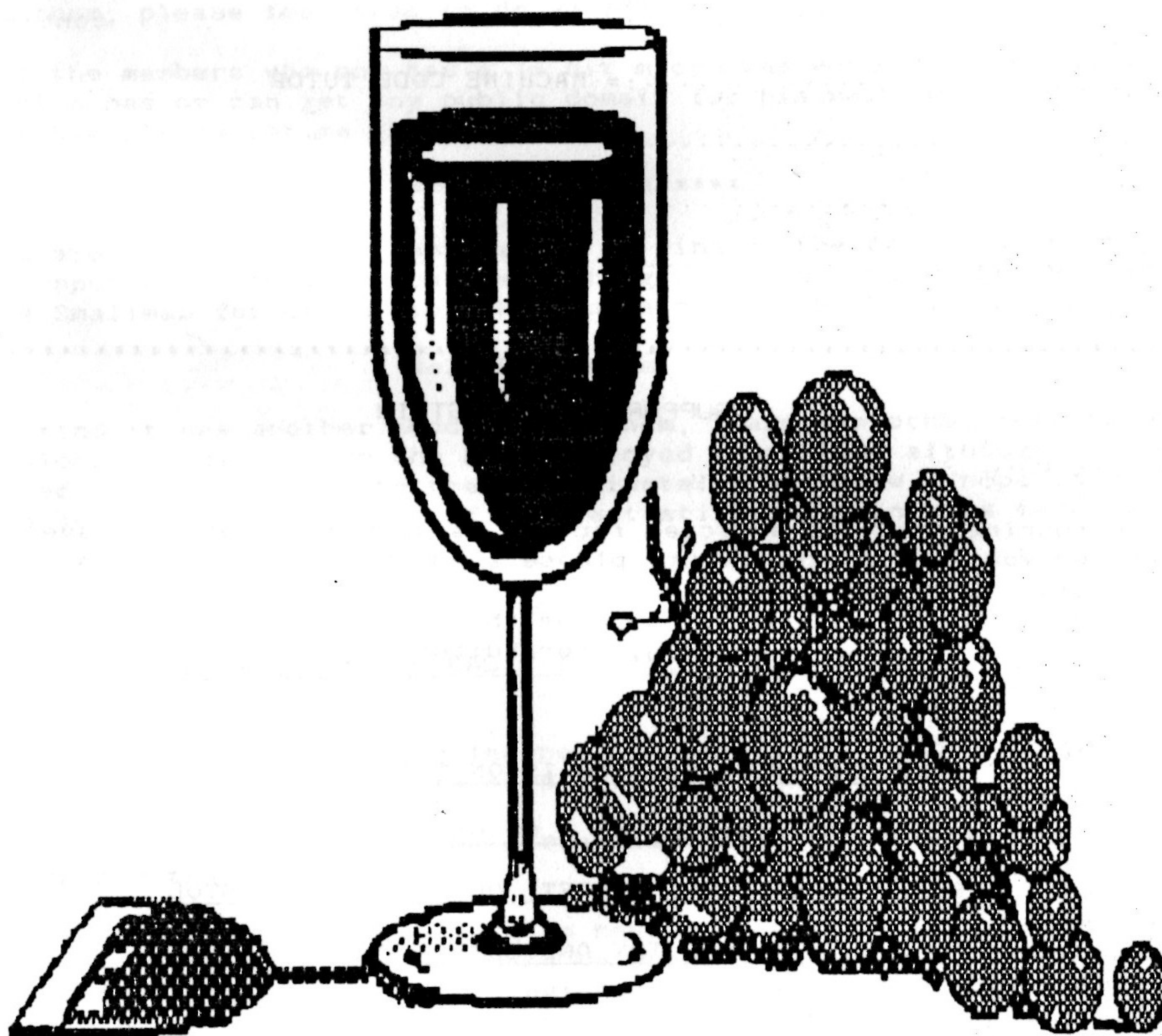


AVON EINSTEIN USER GROUP

Vol. 3. No. 4



PAGE INDEX

(1).....	= CLUB NEWS & BASIC FOR BEGINNERS
(2).....	= BASIC FOR BEGINNERS continued
(3).....	= LETTERS
(4).....	= THE EINSTEIN SHOW
(5).....	= ONE LINERS
(6).....	= HOW I TACKLED CP/M AND WON
(7).....	= " " " " continued
(8).....	= " " " " continued
(9).....	= MACHINE CODE TUTOR
(10).....	= " " "
(11).....	= " " "
(12).....	= " " "

SUPPORT THE EINSTEIN

Editors Note.....

I am running short of articles for the newsletter so if you have anything you wish to contribute please send it to me as soon as possible.

Bob Smith
Newsletter Editor

THE NEXT CLUB MEETING IS ON THE 11th of MAY

starting at 8.00 pm at the

BLACK HORSE, WEST STREET, OLD MARKET, BRISTOL

PLEASE TRY AND COME ALONG

Dear Members

Welcome to this months issue and the first that I am writing on my XT so please for give me if it dos'nt look quite right yet.

CLUB NEWS....

At the last club night (13/4/89), the club bought a 80 column card for the clubs computer and also a 80 track, double-sided, double-density disc drive for the librarian (Graham Higgins), so now if you have any 80 column software that you wish to show other members at the meeting you can !!!.

A good crowd turned up and everybody seemed to enjoy themselves so if you have nothing to do on the 11th May then come along and have a chat.

Several other members have up-graded to 16 bit computers so to go with the trend I will try and sort out a page in the newsletter for 16 bit users, those who now have 16 bit machines and can send me something about them, please feel free to do so ???.

One of the members who now has a 16 bit micro was wondering if the librarian has or can get any public domain for his machine so if you can Graham please let me know !!

If you are wondering what has a glass of wine on the front cover to do with computers ?, it is quite simply to give a toast to Graham Bettany & Mike Smallman for orginising the Einstein Show which was on the 22nd April.

Well Done Lads !!.

To my mind it was another success for them, I and the other members who came along to help out on the stand enjoyed themselves although it was rumoured that this would be the last Einstein only show I hope it will not be judging by the amount of interest still shown in this machine.

Programming in BASIC for Beginners - Part 4

So far in this series we have learned how to write a simple program and how to save and load a program on disk.

In this article we will be considering additions to that program which will enhance the display.

Let us suppose that we would like to have a nice clear screen for our words to be printed on. The command to erase all text from the screen is CLS which is CLeAr Screen abrieviated. Whilst considering the use of this command we must also bear in mind that XTAL BASIC gives us two on-screen print modes. One mode is 32 column and the other is 40 column. Columns refer to the number of characters which can be displayed on any line running from left to right. Therefore 40 column

means 40 characters per line. In order to change to 32 column we need to use the CLS command immediately followed by the figures 32 ie. CLS32. Similarly to change to 40 column mode we must use CLS40. If we find it necessary to clear the screen without changing the mode we simply use the CLS command and omit the figures ie. CLS. Note that CLS32, CLS40 and CLS may be used in the DIRECT or DEFERRED mode and once the display mode is specified it remains unchanged until another command is encountered. On power-up the column width is set to 40.

COLOUR

The EINSTEIN has 16 colour settings and the BACKDROP, TEXT and GRAPHICS colour may all be changed independantly of each other. On power-up, the Text and Graphics colour is set to WHITE (15) and the Backdrop colour is set to DARK BLUE (4).

In order to change the Backdrop colour the command BCOL is used which must be followed by a number in the range 0 to 15 ie. BCOL15 will result in the backdrop colour being set to White as 15 is the colour code for white. Similarly, TCOL15 will result in the text being displayed in white and GCOL15 will result in the graphics being displayed in white. Care should be taken to ensure that text and graphics colours differ from the backdrop colour. White text on a white backdrop = INVISIBLE PRINT !!!!!.

This is a list of the Colour Codes for the EINSTEIN.

- 0 - Transparent
- 1 - Black
- 2 - Medium Green
- 3 - Light Green
- 4 - Dark Blue
- 5 - Light Blue
- 6 - Dark Red
- 7 - Cyan
- 8 - Medium Red
- 9 - Light Red
- 10 - Dark Yellow
- 11 - Light Yellow
- 12 - Dark Green
- 13 - Magenta
- 14 - Grey
- 15 - White

The TCOL & GCOL commands can if need be, followed by two sets of figures. These two sets of figures must be separated by a comma ",". As previously stated, the first figure controls the actual colour of the text or graphics. The second figure (when included) controls the background colour of each character call.

Now key in and RUN the following program....

```
10 CLS32
20 BCOL3
30 TCOL 1,6
40 PRINT"ATUNG EINSTEIN"
50 END
```

Comparing this with our original program we should note the following

changes:

- (a) When the program is RUN the characters are slightly larger and we can only get 32 characters on one line. (Line 10 CLS32)
- (b) The screen is cleared before any text is written. (Line 10 CLS32)
- (c) The backdrop colour is set to Green. (Line 20 BCOL3)
- (d) The colour of the text is Black written on a Red background. (Line 30 TCOL 1,6)

Experiment with this program using different values on BCOL & TCOL until you are completley familiar with the system.

LETTERS

David Greathurst writes.....

Can anyone in the club explain to me how to alter DOS to run different disk drives ie. 5.25", 3.5" and 3".

Bill Salt writes.....

(Q.1.)... I had a RTTY program on cassette for the Spectrum but unfortunately the cassette got damaged. This program worked fine on the Spetrum because the beauty of it was it did not need any additional circuitry such as an interface etc. I bought a RTTY program from Mike Pugh but found that it was only a decoder for incoming RTTY signals (not much use for amateur's). I would appreciate any information or knowledge of this program or if one is available on the Einstein.

(Q.2.)...Could anyone explain the various extentions on programs such as DOC, OBJ, DAT, UFT, SRC, OVR, BAK, BAS, KEY, \$\$\$, DCT, 901, 907, ME etc etc. I have managed to cope with one or two such as DISP PROG.DOC, DISP FLIGHT.UFT also are they COM files or XBAS ????

(Q.3.)... Regarding the program "FLIGHT SIMULATOR", how do I get the confounded thing to start, I can't seem to get any RPM's, what have'nt I done ?.

(Q.4.)... I have the program CAT and very useful it is, but what are these other ones. CAT2, CATP, UDCAT, and MAST ?.

(Q.5.)... Is it possible to sort say 3 or 4 disks together onto a typed sheet, I realize it would be easy with a hard-disk but I have'nt got one. It would be very handy for checking, finding a file, also checking for duplicates of programs. I am thinking in terms of 400 to 500 files. Is it possible ?, any suggestions ?, nice ones please !!!!.

If anyone can help these members then please contact me, Bob Smith. (address on the back cover)

THE EINSTEIN SHOW

D-Day...April 22nd 1989.
Time....07.30 am.
Start...Weston-Super-Mare.
Destination...National Motorcycle Museum, Birmingham.

Well after loading up the car at 02.00 am, to save me doing it in the morning, I finally got to bed and got some sleep. At 07.30 am the alarm went off and I got up and got myself ready to go to the Show. I was meeting John Nash at Patchway so that for a change someone else could do the driving. We then filled John's car with all the stuff we were taking and set off.

Arriving at the show at 10.15 am we went into the show and found our stand which was being guarded by Mike Ivory and Russ Towler and started to set up the stand with the items that we bought with us.

Russ was doing a demonstration of his program "CASHFLOW" which after a slow start people started to show interest in, Mike and myself were doing a roaring trade in the items we had to sell infact Mike would make a good shop keeper !!, in the mean-time John had set up his computer and was showing his program "BIG PRINT". At about 11.00 am, Graham turned up and set up the Public Domain software (he remembered this time to bring the disks) and we settled down for a busy day.

I don't know the exact attendance figures but the hall seemed to be very busy all day, a bit later on I managed to get round and meet some of our members and colleagues who had stands there. It was nice to see Chris Pickles (SUPERSOFT), Mick Pugh (HOLLYWOOD BUSINESS SUPPLIES), Scott Huxley (M & B.E.U.G.) also Theo Steward and Les Morgan who I have spoken to many times on the phone. I was very surprised that more of our members who live in the Bristol area did not turn up ? after all if you do not support these shows how do you expect the Einstein to survive ?, if everyone left it to somebody else nothing would get done and the money you spent to buy the Einstein may as well been thrown down the toilet !!!!.

At about 4.30 pm things started to go quiet so as we had sold out of everything, John and myself loaded up the car and bid farewell to everybody and set of home, thats when our problems started, about 5 mins down the M42 the camshaft on Johns Rover decided to pack up. Fortunatley I am a member of the A.A so after contacting them we were relayed home.

Even with this problem, the day was very enjoyable and although there was a lot of rumours going around that this would be the last Einstein only show I hope it will not be. I have not got a Einstein any longer but it still was one of the best computers around and got me rely interested in computers (I wish the IBM was as friendly). So hears looking forward to the next show.

Bob Smith (Editor)

ONE LINERS (XBAS 4.2) (Best on a colour monitor)
REPRODUCED WITH THE KIND PERMISSION OF THE U.K.E.U.G.

Don't forget if you want to get all the characters on one line you will have to extend the line length, this is done when you have loaded XBAS by typing CLEAR &EB00:PTR12,&EB00:PTR13,249 (ENTER)

This month we have a two oneliners from D.Coverdale which are pattern generators and one each from D.Williams and P.Swan and M.V.R.Paterson.

```
1 CLS:FOR T=5TO266 STEP 3:FOR R=23TO124:GCOLT/12:DRAW T,R TO R,T:
NEXT:NEXT
```

```
1 CLS:FOR T=5TO266 STEP 3:FOR R=23TO190:GCOLT-3:DRAW T,R TO R,T:
NEXT:NEXT
```

For more 'special effects' on the second program add VPOKE 64313+T,R in between the two NEXT statements. It runs for about 4 mins then you will need to reset the machine.

PATTERN (XBAS 4.2) by David Williams

```
1BCOL1:CLS32:FORA=2TO37:PRINT@2,0;A:ELLIPSE163-4*A,96,75-2*A:FILL163-4
*A,96,A:NEXT:TCOL15,1:PRINT@3,0;"THE END":TCOL1,1:BEEP
```

SPIRAL (BBCBASIC) by Paul Swan

```
10 MODE0:GCOL0,15:MOVE500,400:FORA=0TO173STEP0.2:DRAW500+(3*A*SIN(A)),
400+(2*A*COS(A)):NEXT
```

CHAOS (XBAS 4.2) by M.V.R.Peterson

Here are a couple of oneliners converted for the TCOL and 256 from the SINCLAIR QL.

```
1BCOL1:CLS:A=3.13:T=1/SQR(2):FORI=1TO5E3:GCOL2+I/100MOD13:Z=Y-SIN(X):Y
=A-X:X=Z:U=T*X+T*Y:V=-T*X+T*Y:PLOT127+5*U,96+4*V:NEXT
```

```
1BCOL1:CLS:A=2.76:T=1/SQR(2):FORI=1TO5E4:GCOL2+I/300MOD13:Z=Y-COS(X):Y
=A-X:X=Z:U=T*X+T*Y:V=-T*X+T*Y:PLOT117+2*U,96+2*V:NEXT
```

256 version ...adjust the first parameters of the PLOT statement to suit greater horizontal resolution. eg. PLOT 247+3*u,96+2*v.
Values of A should be around PI for the first (SIN) program and between 2.725 and 2.795 for the COS version.

How I tackled CP/M and won
=====

The implementation of CP/M for the Einstein (well, the implementation which I have anyway, from ACC) appears to be deficient in at least one respect - like DOS 1.31 it assumes that all the available drives are of the 3", 40-track, single-sided type. If you want to stay with DOS, then there are two possible upgrades to systems which are able to read/write other capacities of drive (single/double-sided, 40-80 tracks), System-80 and XTAL-DOS 2.5. Unfortunately, there is no equivalent upgrade for CP/M.

This presents a bit of a problem when one is playing around with Public Domain software. Most of it is CP/M-based and some of it will *not* run under DOS. In addition, some of the files are far in excess of 188K, which presents a bit of a problem, as CP/M doesn't recognise the larger capacity drives, so I had to do something about it.

For each of the four available drives (0 - 3, or A - D) there is certain information specific to each of the drives held in certain locations in the OS (initially on the disk and subsequently written into memory during bootup). This information is known as the Disk Parameter Header (DPH) and it contains the addresses of the areas in memory that contain information which pertain to that disk drive.

Part of this information is a specific area which contains the data on the type of drive. This is known as the Disk Parameter Block (DPB). In the DPB is contained things like number of sectors per track, number of boot tracks taken up by the OS, etc. There should be a DPB for each possible drive type. In order to add a different type of drive to the system, all you should have to do is find out which DPB matches the new drive and arrange for its address to be put into the matching DPH.

Simple enough.

Well there *should* be several DPB's, (single and double sided, 40 or 80 tracks) but in DOS 1.31 and Einstein CP/M there is only one (40-track SS) --- and all the DPH's point to this single DPB. That's why, if you want to add anything different to the standard 40-track, SS type, you have to buy an upgraded OS. This is all well and good for DOS, both Dos 2.5 and System 80 support different drive types, but there isn't an equivalent upgrade for CP/M - hacking the system is necessary.

In principle, it's not too difficult a task; find out what the details are for the DPB's for the drives you wish to add, include that information in the OS and tell it where to look.

Unfortunately, extending the OS in this fashion rather requires that you have the assembler sources of the operating system, so you can add the information, compile it and put it back as boot tracks on the disk. No dice. So instead, I came up with a hack (surprise, surprise).

First of all, I needed the information to make a DPB. That wasn't too difficult, all I had to do was look carefully at the memory after booting up from a DOS 2.5 disk - remember, the DOS 2.5 system must have that information if it can cope with disk capacities different from the standard 40-track SS type.

The DPB for the standard 40-track SS drive looks like this:

28 00 04 0F 00 5E 00 3F 00 80 00 10 00 02 00

The DPB for an 80-track DS drive looks like this:

50 00 04 0F 01 8A 01 7F 00 C0 00 10 00 01 00

I'm not going to go into detail about what the bytes actually mean, suffice to say that they all convey information about the type of drive and how the OS can use it. If anyone wants more detail, any *good* book on CP/M will tell you, or you can refer to back issues of the UKEUG (which I did).

The next problem was finding somewhere in the OS to put this extra information --- I couldn't just change the existing DPB, otherwise it would think that ALL my disks were of this new type (there's only one DPB remember --- the task is to add another one. After spending some time prowling round the OS in memory with PATCH (good old PATCH!), I discovered some "unused" space into which I could put the 15 bytes required for an extra DPB for my 80-track double-sided drive.

Changing the DPH was no problem, there are four DPH's, one for each drive --- and in the middle of each is an address (that's four addresses in total) --- in CP/M, each of those addresses was the same (all pointing to the standard DPB for a 40-track SS drive). All I had to do there was change the address of the DPH for my extra drive to point to the area in memory where I was going to put the extra DPB. The first DPH was for drive 0/A, the second for 1/B and so on. I wanted to make my new big drive the 1/B drive, so I had to change the DPH for drive 1/B

I hacked up a small assembler program, defining where the DPH was in memory, where I was putting the extra DPB and the information which was to go into the extra DPB. My use of quotes round "unused" was deliberate --- I wasn't sure whether it would remain unused, the blank area may have been a scratchpad or buffer which the OS might write into - and thus overwrite the disk information, so I arranged that the program would be run on each boot. Any problems caused by the OS not being able to use that scratchpad (if it was indeed one), I felt I could live with.

By using the AUTOBOOT utility, I was able to make the program run on a boot, but unfortunately, the termination of the program *caused* a boot, so it started looping round and round. I solved this one with a typical hack --- I'm not well-enough versed in assembler to solve the problem of the program terminating in a boot, so I adopted a different tack.

The only time the new DPB got erased was when the OS was read from disk (booted), then the booting would put back the original blanks where I had put my DPB. When you set up an autoboot, the name of the program you wish run on a bootup is written onto the disk. On bootup, that name is then written into a specific place in memory -- where the OS normally stores what you type to the > prompt (i.e. the command buffer). Then it simply behaves as though you had just typed it in.

So, my program is run on bootup, then to avoid it running again, the last thing it does is write blanks into the command buffer, overwriting its name which was put there on bootup - but it only does this *after*

it has done its job adding a new DPB and changing the address pointer in the appropriate DPH. Well, it works like a charm. For those of you who are interested, here's the assembler code --- it is assembled with ZASM, the PD macro assembler (available from the AEUG PD library).

```
ORG 0100H; Set the address of the start of the prog
IBUF EQU 0E407H; Address of CCP input buffer
DPBB EQU OFC90H; Disk parameter block address for B drive
DPBD EQU OFCA0H; Disk parameter block address for D drive
DPHB EQU OFA4DH; Disk parameter header for B
DPHD EQU OFA6DH; Disk parameter header for D
```

```
INIT:
LD HL,DPB1; Load address of disk parameter details for B into HL
LD DE,DPBB; Load DE with address of DPB for B
LD BC,15; Load BC with length count
LDIR; Dump it
LD HL,DPB2; Load address of disk parameter details for D into HL
LD DE,DPBD; Load DE with address of DPB for D
LD BC,15; Load BC with length count
LDIR; Dump it
LD HL,DPHB; Load HL with the address of B's DPH
LD (HL),090H; Load that address (LSB) with B's new DPB address
LSB INC HL; Move to MSB
LD (HL),OFCH; Load the MSB with MSB of B's new DPB address
LD HL,DPHD; Load HL with the address of D's DPH
LD (HL),0A0H; Load that address (LSB) with D's new DPB address
LSB INC HL; Move to MSB
LD (HL),OFCH; Load the MSB with MSB of D's new DPB address MSB
LD HL,IBUF; Load HL with address of input buffer
LD DE,SPAC; Load DE with address of info to be dumped
LD BC,15; Load BC with count
LDIR; Drop the info in memory
RET; Return

DPB1: DEFB 50H,00H,04H,0FH,00H,89H,01H,7FH,00H,0C0H,00H,10H,00H,01H,00H
DPB2: DEFB 50H,00H,04H,0FH,00H,89H,01H,7FH,00H,0C0H,00H,10H,00H,01H,00H
SPAC: DEFB 20H,20H,20H,20H,20H,20H,20H,20H,20H,20H,20H,20H,20H,20H,20H
END
```

Cheers,
Graham
=====

Machine Code Tutor Part 2

O.K here we go for part 2. Unfortunately there has to be a change from what I indicated we would do this month. My wife has been in hospital for several weeks and I just could not find the time to do the Function Key program.

Also I have decided to use the public domain assembler ZASM in this series. So if you have not got this package I recommend you see Graham, he will only be too pleased to give you a copy from the club library. I am not 100% sure of all its potential so we can find our way through it together (with loads of help from Graham and Russ). Who have pointed me in the right direction on this assembler. They tell me it is a pretty powerful and useful tool.

So it seems logical this month we will have an introduction to this assembler and then write another masterpiece of machine code that will (1) clear the screen. (2) move the cursor to a given location and print a message. (3) move the cursor back to the home position. (4) move a block of memory from one address to another then print the message from the stores.

The actual machine code generated by the assembler is in the second column of the listing. The address for the byte is in the first column. See line No.0036 in the listing. Address 0100 code 3E. look up the op code for 3E and you will see it is LD A,n.

Note the use of lables in the assembler listing, these are used to load up the registers. E.G LD HL,Stores. By using labels the assembler works out the address for us. If we should insert more code between any lines the labeling take care of the different addresses required because all the code has move down by the number of bytes you have inserted.

ASSEMBLER INSTRUCTIONS

Assembler instructions are reserved words which the assembler uses (just like basic). These words are not machine code but are commands to do something for us. Lets look at a few that are common to most assemblers I have used. Thers are others on this assembler but more about them another time. We will write a small program using these six below.

(1)	DEFB	DEFine Byte
(2)	DEFM	DEFine Message
(3)	DEFS	DEFine Store
(4)	DEFW	DEFINE Word
(5)	EQU	EQUals
(6)	ORG	ORGanize

(1) DEFB this means define a byte. A byte is an eight bit number that is a between 0 and 255. DEFB allows you to place a byte with the value of your choice at the address relative to the code where you have placed it. This byte can be numeric or a character. If a character is used then the ASCII code of that character will be inserted at that address. E.G DEFB 65 would put the value of 65 into that address. DEFB 'A' would also put 65 into that address because 65 is the ASCII value for the character A.

(2) DEFM means define a message, it allows you to place a whole message at the address(s). This is very useful because we don't have look up the ASCII codes for all the characters in the message. On the ZASM assembler it also adds the 80 hex value to the last character

for us. (not all assemblers do this). Adding 80 hex to the last character is required by the processor to let it know that the message characters have terminated and the following bytes are machine code instructions. E.G DEFN 'JOHN' would appear in the code as hex numbers 4A 4F 48 CE . From that we can see 4A=ASCII code J , 4F=ASCII code O , 48=ASCII code H , but the last one N is the ASCII code for the character N=4E hex, plus 80 hex totaling CE hex. Or in dec it would be $N = 78 + 128 = 206$. 206 dec = CE hex.

(3) DEFS this means define a storage area of n bytes at this location. E.G DEFS 9 would instruct the assembler to reserve 9 bytes at that address. A store is a place just like a store anywhere, it a place to deposit things in and take things out. In our case we put in or take out numbers.

(4) DEFW this is the same as define byte but is defines a sixteen bit number instead of an eight bit number. E.G DEFB 65221 . A number between 0 and 65535.

(5) EQU this means EQUALs or EQUate what it does is allow you to use a name to EQUAL a number. This make your program more readable. E.G CIs EQU 0Ch. From this every time the assembler find the CIs it will place the number 0C hex into that byte. Therefore at the begining of a program we declare all our EQS's, then we write our listing in a more readable format.

(6) ORGanise this tells the assemble where you want the code to run from. As you know COM files run from 0100 hex therefore most of you ORG declarations will be ORG 0100h. Alternatively you can make your ORG declaration high in the memory say at about 50000 dec. In this case you would use this to write a OBJ code sorce to use from a BASIC program using the BASIC CALL command.

O.K now for a summary of what is needed.

- | | | |
|-------|----------|-----------------------------------|
| (1) | VED.COM | editor |
| (2) | ZASM.COM | assembler |
| (3) | LOAD.COM | to make your file into a COM file |

(1) VED is the editor in which you write your file to be assembled by ZASM. The listing that follows was written using VED. The file you write may have any name you wish but the EXTENSION MUST BE Z80. E.G ANYNAME.Z80 this is because ZASM looks for this extension only. Any comments you wish to make must be preceeded by a semi colon. This acts just like a REM statement in BASIC. The assembler will ignore anything after the semi colon.

(2) ZASM is the assembler, it assembles the listing produced on VED. There is documentation with this program so I will not go into it more than to say it is implemented from DOS by ZASM AAX.ANYNAME HEX There are other arguments but the ones above are for single drive machines. The command above means ZASM this invokes the assembler ZASM the next three characters AAX = DRIVE:DRIVE:OUTPUT TO SCREEN. This means DRIVE A for sorce file DRIVE A for destination file, X is to show output to the screen, a Y would give output to the printer, a Z would give no output. HEX means a HEX file will be created on your destination drive.

(3) LOAD this program makes your HEX file created by ZASM into a COM file. It can then be run just like any COM file.

	0001	:	In the first issue we used the machine instructions
	0002	:	LD RST INC DJNZ RET. We will use these again plus LDIR
	0003	:	A note on the LD instruction, in this months program we are also
	0004	:	using indirect addressing E.G LD (nn),A refer to the first issue
	0005	:	notes on ways of addressing the registers.
	0006	:	The LDIR instruction we are going to use in this program needs some
	0007	:	explaining. It means Load INCrease and Repeat. Broken down this is
	0008	:	Load the DE register pair with the contents of the HL register pair
	0009	:	then increase the HL register, increase the DE register, decrease
	0010	:	the BC register until BC = ZREO.
	0011	:	As you will see from this it is a block movement of bytes to move
	0012	:	a block of memory from one place to another. HL being the sorce
	0013	:	address, DE being the destination addresss, and BC the counter.
	0014	:	So before we can use this instuction we must Load up HL,DE, and BC
	0015	:	with their values.
	0016	:	The instruction LDDR the same in reverse meaning Load DECrease and
	0017	:	Repeat.
(0100)	0018	ORG 0100h	; ORGanize code to run from 0100 hex
	0019		; Normal start for COM FILES
(000C)	0020	CIs: EQU 0Ch	; 0C hex 12 dec Control character to clear
	0021		; the screen and home the cursor
(FB4A)	0022	Col: EQU 64330	; Address FB4A hex 64330 dec is scratch pad
	0023		; location for column position of cursor
(FB4B)	0024	Line: EQU 64331	; Address FB4B hex 64331 dec is scratch pad
	0025		; location for line position of cursor
(00CF)	0026	Messq: EQU 207	; MCAL CF hex 207 dec. This MCAL prints the
	0027		; whole message and is terminated by adding
	0028		; 80 hex to the last character to print.
	0029		; This assembler adds it for you.
(009E)	0030	Print: EQU 9Eh	; MCAL 9E hex 158 dec
	0031		; This MCAL prints the character held in the
	0032		; A register.
	0033		
	0034		; ***** START OF CODE *****
	0035		
0100 3E0C	0036	LD A,CIs	; In this case it is a control character
0102 CF	0037	RST 8	; Tell the processor to expect a MCAL
0103 9E	0038	DEFB Print	; We are going to use MCAL 9E hex.
	0039		; This will clear the screen because we
	0040		; have tried to print a control character.
0104 3E05	0041	LD A,5	; We are going to move the cursor to column
0106 321AFB	0042	LD (Col),A	; 5. By poking the 5 in the A register into
	0043		; the address in the brackets.
0109 3E0A	0044	LD A,10	; Now move the cursor to line 10 as above.
010B 324BFB	0045	LD (Line),A	; See declarations Col EQU 64330 dec and
	0046		; Line EQU 64330 dec
010E CF	0047	RST 8	; O.K lets print something at this location
010F CF	0048	DEFB Messq	; We are going to use MCAL CF hex 207 dec
0110 57652068	0049	DEFN	'We have moved to line 10 col 5'
012E 3E00	0050	LD A,0	; Lets move the print position back to col 0
	0051		; and line 0
0130 324AFB	0052	LD (Col),A	; Load up address 64330 with the A reg ie zero
0133 324BFB	0053	LD (Line),A	; Load up address 64331 also with the A reg
	0054		; The cursor will now move to col 0 line 0

CROMEMCO Z80 Macro Assembler version 03.04
Source File: TUTOR2A

Page 0002

	0055		; O.K Lets try another way of printing using
	0056		; the MCAL 9E hex again. This time in a loop
	0057		; but getting the characters from a store
	0058		; to make use of the DEFS assembler directive
	0059		; But first of all we must have something in
	0060		; the stores to print. We will do this by
	0061		; using the LDIR instruction. I have put a
	0062		; DEFB directive after the stores with the
	0063		; label Trans. This shows how to use the
	0064		; characters to give ASCII codes.
	0065		; We will use the labels to give us the
	0066		; addresses required for the HL and DE
	0067		; registers.
0136	216D01	0068	LD HL,Trans ; Put source address into the HL reg
0139	114D01	0069	LD DE,Stores ; Put destination address in DE reg
013C	012000	0070	LD BC,32 ; Number of bytes to move
013F	EDB0	0071	LDIR ; Move it
	0072		; O.K the bytes are now in the stores
	0073		; Now lets get them out of the stores
	0074		; and print them.
0141	0520	0075	LD B,32 ; Load the B reg with 32 the number of
	0076		; character we want to print.
0143	214D01	0077	LD HL,Stores ; Load the HL reg pair with the address of
	0078		; the stores. The assembler will look for the
	0079		; label Stores see its address and Load this
	0080		; value into the HL reg.
0146	7E	0081	Loop1: LD A,(HL) ; Load the A reg with the contents of HL reg.
	0082		; In other words the () around a register
	0083		; means what is at the address contained in
	0084		; the brackets is put into the A reg.
	0085		; In BASIC A=PEEK(address).
0147	0F	0086	RST 8 ; Tell the processor to expect a MCAL
0148	9E	0087	DEFB Print ; MCAL 9E hex as used earlier.
0149	23	0088	INC HL ; Increase HL to point at the next character
	0089		; in the stores.
014A	10FA	0090	DJNZ Loop1 ; Decrease the B register B=B-1 and Jump
	0091		; Relative back to Loop1 if B is Not Zero
014C	C9	0092	RET ; RETurn to system
014D	(0020)	0093	Stores: DS 32 ; Reserve 32 bytes at this address
	0094		; The message below is 32 bytes long and is
	0095		; used to fill the stores to suit our print
	0096		; routine. A store is more commonly used to
	0097		; store DATA bytes (like the scratch pad)
016D	536F6D65	0098	Trans: DEFB 'S','o','m','e','t','h','i','n','g',' '
0177	746F2070	0099	DEFB 't','o',' ','p','u','t',' ','i','n','t','o'
0182	20746865	0100	DEFB ' ','t','h','e',' ','s','t','o','r','e','s'

Errors	0
Range Count	0

AVON EINSTEIN USER GROUP COMMITTEE MEMBERS

CHAIRMAN MIKE IVORY, 1 HEATH ROAD, HANHAM, BRISTOL, BS15 3JT.
TELEPHONE No. 0272-616281

SECRETARY JOHN NASH, 38 BURFOOT GARDENS, STOCKWOOD, BRISTOL,
BS14 8TE.
TELEPHONE No. 0272-838028

TREASURER ROLF GEORGE, 25 FAIR VIEW, CHEPSTOW, GWENT, NP6 5BX.
TELEPHONE No. 02912-4916

LIBRARIAN GRAHAM HIGGINS, 31 LENA STREET, EASTON, BRISTOL,
BS5 6BD.
TELEPHONE No. 0272-559874

EDITOR.....BOB SMITH, 2 INGLETON DRIVE, WORLE, WESTON-SUPER-MARE,
AVON, BS22 0SR.
TELEPHONE No. 0934-517465

Any articles, listings, reviews, questions, complaints !! please send them to the Editor.

For all Public Domain enquiries !! please contact the Librarian.

PLEASE REMEMBER IF YOU ARE SENDING DISKS, AND YOU WANT THEM RETURNED ?

THEN ENCLOSE AT LEAST 28p RETURN POSTAGE
(Return address label if possible)

Please Note All articles and programs become the copyright of the A.E.U.G. as well as the original authors !!!!.