



# Einstein Magazine

**& ALL MICRO NEWS**

**Number 94**

Published for users of Einstein (and other) computers  
by RPM Society.

**Publisher and Secretary:-**

**A E Adams, Ivy Cottage, Church road, New Romney,  
KENT TN28 8TY**

\*\*\*\*\*



Great Moments in Computer History: 3  
The Einstein Boffin finds the bug he's been hunting.  
Unfortunately it's not the Millennium Bug!



# Einstein Magazine

**& ALL MICRO NEWS**

**Number 94**

**Published for users of Einstein (and other) computers  
by RPM Society.**

**Publisher and Secretary:-**

**A E Adams, Ivy Cottage, Church road, New Romney,  
KENT TN28 8TY**

\*\*\*\*\*

**EDITOR: Ted Cawkwell**

**9 King Street Winterton N.Lincs. DN15 9RN**

\*\*\*\*\*

**SHOWS, SOFTWARE LIBRARY and USEFUL BITS**

**Steve Potts 85 Thorold Ave, Cranwell Village, Lincs.  
NG34 8DS**

\*\*\*\*\*

## Contents

- 2 Using the ADC ports - you asked for it!**
- 6 What's Wrong with Basic. The last article of the saga.**
- 8 Type it in - this time it's a Compass**
- 9 The All-electronic AABBBBAA Switch. Is it really worth all the hassle?**
- 13 Pull the Other One with this novel add-on by Bob Deeley.**

\*\*\*\*\*

Articles are Copyright the original author and the Einstein User Group.  
Articles without a byline are Copyright the Editor and the EUG.

\*\*\*\*\*



## USING THE ADC PORTS

I would like to thank the anonymous member who requested details of using the User Port to input temperature readings. As soon as I started to look at the problem it became clear that the ADC port was a better bet because all of the electronic thermometer circuits I could find had an analogue output. The joystick ports are designed for this very purpose, whereas the user port needs a digital input.

I had never tried using the ADC ports before so my first job was to see if anyone else had, and I soon found an article in Einstein Monthly Vol 1,3 where Peter Moon has an article for graphing the output of a 1.5 volt battery. It is on page 10. Using his method I evolved the following program to check the voltage of a small battery (less than 2.0 volts).

It is fairly precise on my TC01 depending what type of voltmeter is used to check. It looks very good on the old needle type meter but is only accurate to the first decimal point on a modern LCD one. How it works on another machine I am hoping members will tell me!

10 REM BATTERY VOLTAGE

20 REM T.CAWKWELL 7/7/99

30 XMIN=3.11

40 XMAX=183.6

50 MAX=1.47

60 FMT 3,2

100 DEF FNA(X)=(X-XMIN)/XMAX\*MAX

110 CLS:PRINT@13,2;"READING PORT0"

120 GOSUB 190

130 PRINT@15,12;FNA(X)

140 FOR J=1 TO 4

150 GOSUB 190

160 PRINTTAB(16);FNA(X);NEXT

170 PRINT@12,20;"AGAIN (A) OR QUIT (Q)?:Y\$=INCH\$

180 IF Y\$="A" OR Y\$="a" THEN 110:ELSE END

190 SIGA=0

200 FOR I=1 TO 500

210 SIGA=SIGA+ADC(0)

220 NEXT I

230 X=SIGA/500

240 RETURN

There is so much electrical noise on the port circuit that a single reading is useless and it is necessary to make multiple readings and take the average. Peter used a 1000 times loop in line 200 (I=1 to 1000) and divided SIGA by 1000 in line 230, giving a rather better precision, but 500 is quick enough to make the loss worthwhile, about 3 secs per reading instead of 9-10 secs. Try changing it to see the effect.

To get the variables XMIN and XMAX I used the following program which will be the basis for a temperature reader. You input your minimum and maximum values with the thermometer circuit arranged to suit and can then use the values obtained in a copy of the first program. By judicious juggling of values it should be possible to obtain a reliable reading between the max and min values and possibly a little higher as well.

10 REM ADC PORT READER

20 REM T.CAWKWELL 7/7/99

30 FMT 3,2:INPUT"SET MINIMUM = ";MIN

40 GOSUB 190 :XMIN=X

45 PRINT "MIN = ";XMIN

50 INPUT"SET MAXIMUM = ";MAX

55 GOSUB 190 :XMAX=X

60 PRINT "MAX = ";XMAX

70 FOR J=1 TO 1000:NEXT

100 DEF FNA(X)=(X-XMIN)/(XMAX-XMIN)\*MAX

110 CLS:PRINT@13,2;"READING PORT0"

120 GOSUB 190

130 PRINT@15,12;FNA(X)

140 PRINT@14,19;"AGAIN? Y/N"

150 Y\$=INCH\$

160 IF Y\$="Y" OR Y\$="y" THEN PRINT@14,19;" ":GOTO 120

```

170 END
190 SIGA=0
200 FOR I=1 TO 500
210 SIGA=SIGA+ADC(0)
220 NEXT I
230 X=SIGA/ 500
240 RETURN

```

To get the figures for my battery reader I ran the above program and arranged a 1.5 volt battery in series with a 250kohm linear potentiometer and connected the wiper to pin 1 of a 7 pin DIN plug. Battery negative went to Pin 6 and also to Pin 2 which is signal ground. The DIN plug was then inserted into the Analogue 1 socket on the right side of the computer. THE POWER WAS OFF when I did this! A voltmeter was placed across the wiper and ground to monitor the actual voltage to the computer.

Connections to the ADC plug are on page 216 of the Introduction Manual.

I then set the voltage (by the meter) to 0 volts and told Albert that Min was 0 volts, then set volts to 1.55 and input 1.55 for max setting. After a short pause 1.55 volts showed on the screen and by twiddling the pot I could get the range from 0 to 1.55 volts. I then changed the 1.5 volt battery for a 3 volt and found that I could get up to 2.01 volts before the increase stopped. The sharpeyed Member who spotted that MAX in the program is 1.47 is quite right - this is the judicious twiddling I mentioned, and is akin to setting a false film speed on a camera to juggle the exposure.

This was as expected because the port will only accept up to 2.1 volts by design. The value read by ADC(x) will be between zero and a maximum of 255, and has to be converted to the output needed. DEF FN is perfect for this job. The expression after it may be translated as:-

$$\frac{X - X_{MIN}}{X_{MAX} - X_{MIN}} \times MAX$$

It will be clear that when X is equal to XMAX, the left half of the equation will equal one, so multiplying by the MAX value will give the desired answer, and values below, and to a certain extent, above will be in proportion. When X is half of XMAX then the result will be half of MAX, etc.

In the case of the thermometer circuit Fig 1, the top value is about 50 deg C and the minimum value 0 deg C. The circuit uses the collector to emitter junction of an AC128 PNP germanium transistor, which is very sensitive to

temperature changes. As shown, an NPN type, like AC127, also works.

The change is amplified by simple op-amp and the whole thing will run from a 9 volt battery, consuming about 3 milliamps. The 3 diodes hold the

collector voltage steady, and the change at the emitter is fed to the 741.

The AC128 needs to be mounted in some sort of waterproof housing to keep water away from the leads. I used a short length of polythene tubing, a tight fit on the transistor, for my experiments. Pin 6 of the 741 goes to the ADC Pin 1 and the 0 volt line (Note: NOT the battery negative) goes to ADC Pin 6.

Run the test program and set the two potentiometers at about halfway. Place a voltmeter across the output. Immerse the AC128 in ice water and after a couple of minutes adjust the

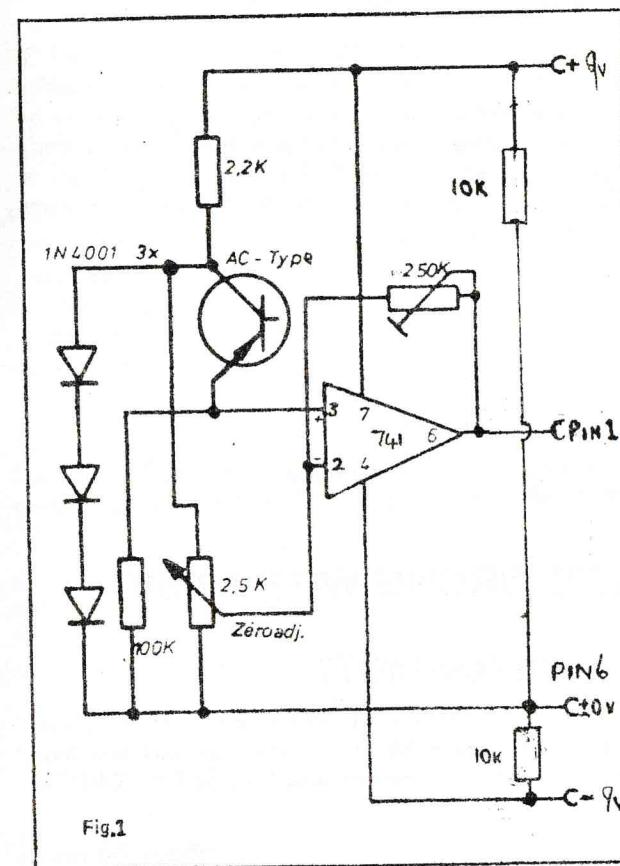


Fig.1

Zero pot for the lowest value you can get on the meter.

Input this voltage for the XMIN reading. Make a note of XMIN.

Change the icewater for a 50 deg C bath with a normal thermometer in it and when the voltage has settled down, adjust the 250k pot for maximum output (but not more than 2.1 volts), then input the observed temperature for the MAX reading. Note the XMAX value.

After a short pause, the bath temperature should come up on the screen. Run a few times by pressing A to see that all is well, then try water at some inbetween temperatures.

Once the circuit is set up transfer the XMIN, XMAX and MAX values to the



first program above, and use this as your temperature reader. The output can obviously be arranged according to your wants or needs. If you only want one reading just zap the FOR/NEXT out of the program. Note also that you can tailor the output by using FMT. If you only need one decimal place use FMT 2,1. This will give the range 00.0 to 99.9.

There are FOUR inputs available at the two ADC ports, using Pins 1 and 3 on both ports, quite enough for example for a weather station covering wind speed and direction, temperature and barometric pressure. The only requirement for the test modules being an output of less than 2.1 volts and, of course, an analogue signal. When using ports 1,2 and 3 the ADC(x) is changed accordingly wherever it appears in the program and a separate DEF FN needs to be set up for each input.

I have not tried to make this article simple enough for rank beginners because it just is not simple to anyone without the knowledge and skill to assemble a circuit and solder up a 7 pin DIN plug. For those that can, I hope that there is enough information to do their own thing. My thanks to my anonymous friend who launched me on a fascinating project. Perhaps, once he has got his thermometer going, he will send in an article telling us all just how he did it?

## AND WHAT'S WRONG WITH BASIC

JOHN MARRIOTT

Well, back again - and how did you get on with last month's SHUFFLE and your version in BASIC? Yes, got quite a bit complicated, looked less than tidy AND if you think that your BASIC Program was intended to SHUFFLE 1800 Records...

...I actually put "Sorting can take some time" in to my Database Program, just as a warning - like. When the Program CALLED the SORT CODE, the screen cleared and up came the MAIN MENU screen, was less than happy with my SORT CODE - I was so sure it hadn't worked, even when I went looking in RAM with my MULTIFACE+3! To be honest, despite what my eyes showed me, I still didn't believe that 1800 Records could have been sorted - but they had!

No, a BUBBLE SORT is just about the slowest SORT MECHANISM you can have ... it's akin to part filling a bucket with building aggregate (sand/mixed shingle) and shaking it - in time (HAH!) you'll find that the finest particles will have settled to the bottom of the bucket and the largest to the top...

...deal yourself 13 shuffled playing cards, now sort them in to SUITS and SUIT NUMERICAL order - if you do what I do, just stick each SUIT between

separate fingers THEN number sort each SUIT ... which is an easier way of describing the slightly faster SHELL SORT although the Programming is naturally more complex. Incidentally, don't be fooled by HOW you THINK you're sorting them - the BRAIN has a marvellous capacity of "short term" evaluation re-call which appears to be used a lot when driving, and because we DON'T recognise how much we are EVALUATING ... we IGNORE it!

The point I'm coming to is that I'd forgotten that what the COMPUTER is KING at is NUMBER CRUNCHING! That's what it was originally designed for, intended for, developed for - it was only when mere stumbling Mortals such as we, who HAD to have an EASIER INTERFACE to the COMPUTER, got involved - then HIGH LEVEL LANGUAGES such as BASIC came along ... yes, and the problem of INTERPRETTING with its LOSS of SPEED penalty...

...what's happened to the sniggers as that estimated time to SORT those Records has fallen from 11 days, to 3 days, to a blink of an eye? So let's get back to the job in hand.

The problem that I find with ASSEMBLERS is the amount of time typing in - of the SHUFFLE Program there were only 88 bytes that were of interest, the rest were nothing more than "overheads". I also find that having to "add" extra "lines"

can cause problems with JUMP RELATIVE Operations because "they" now can't JUMP far enough, or suddenly there's NOT an ASSEMBLY INSTRUCTION for what I want the program to do - so, I've tended to forget about COMPUTER ASSEMBLING and to "do it by hand"...

...oh dear! Off you go, say three "Hail Mary's"...

One of the things that endears me to the EINSTEIN is its MACHINE OPERATING MODE, which allows you to DIRECTLY MANIPULATE memory - AND that it happily accepts, no it expects you to enter values in HEXADECIMAL. Those 88 bytes could be directly punched in to memory ... hands up those who didn't think of that! So what is to stop you developing your Coding direct from a BASIC Program outline?

As you can see from SHUFFLE, it's longer than it needs to be - well, there's CODE for 3 Fields when I'm sure with a little more bit of "elegant" programming it could be brought down to one loop - but for the extra "speed" is it worth the time? If you take a section of GAME CODE, you'll find the efforts which Programmers put in to obtain just that - but look at the overwhelming amount of "information" they've got to handle...

...at the beginning, leave it simple and chose a program project which is going to be of USE to YOU. From my experience, people learn better on the "need to know" basis, rather than the "sausage machine" technique our Educational System INSISTS on oppressing us with - and you'll learn



quicker. When, and only when your CODE is functioning 100% look at it and see what you can change/learn from. To give you an idea, the final update to that Database Program was on 9.9.91 when the BASIC portion of Database was finalised - just about 2 years after the SHUFFLE ... the Foreman Fitter had been upgraded to Trainee Technical Officer because of his "effective work on the logbooks" and I was on Early Retirement Leave, pondering about how unfair Life was...

...although there is a sequel to that, it's of no real interest here, other than to say ALWAYS put a COPYRIGHT MESSAGE in your ORIGINAL work (I HAD!). By now, that Database Program was RIGID in its FUNCTION, bore no relationship to what started out, even on a SPECTRUM+2 was fast, most of it brought about by simple changes, simple re-programming and the main thing - THINKING.

Looking back - for hindsight is wonderful, I realised that there was no real need to ACTUALLY sort the Records in RAM - let's call it a COMPARISON RETRIEVE, this is where you can pull out information from MEMORY by comparing a STRING to MEMORY content and displaying it - most DIS-ASSEMBLERS and WORDPROCESSORS have it, but I have the sneaky feeling that the CODE will turn out to be rather long and rather complex as it tries to emulate some form of RUBBISH SYNTAX check...

...but isn't hindsight part of our LEARNING CURVE? Just one slight P.S. - if YOU don't support YOUR computer by contributing your time, interest AND money to your USER GROUP - why should anybody else?

### USE IT - OR LOSE IT!

—@@@—

## POINT THE WAY WITH COMPASS

I originally wrote this for my GOLF program but it made a mess of the display so was never used, but it may be useful to someone. The circle is not really necessary but is just included for a reference point.

```
10 CLS:GCOL1:A=0:X=120:Y=90:ORIGIN 0,0:L=32
```

```
15 ELLIPSE 120,90,40,1
```

```
20 PRINT@5,4:INPUT"Compass deg.from N? ";A$
```

```
30 IF A$="" THEN 20
```

```
40 AA=VAL(A$):IF AA<0 OR AA>360 THEN 20
```

```
50 A=A+AA:L=L+LL
```

```
60 IF A<0 THEN A=A+360
```

# INPUT #

**INPUT#** (Input Device Number)

**Syntax:** INPUT#J

Where J gives a "device number" assigned to a device and in the range 0 to 254.

**Purpose:** This statement assigns a new input device (eg. serial port) indicated by the value of J.

All input statements, such as INPUT and INCH\$, will be received from the new device selected by J until another INPUT# statement is encountered to change the device selection.

When a program ends or aborts, either through an error or as directed from the keyboard, the input device **reverts** back to the keyboard (i.e. device 0).

If INPUT# is used in direct mode the corresponding input statement must appear in the same line.

The following two input devices are currently assigned within the BASIC language provided.

DEVICE	DEVICE NUMBER
KEYBOARD	0
SERIAL PORT (RS232)	2

### EXAMPLE:

```
INPUT#2
```

All input following this statement in a program will be received from the serial port.

INPUT# can also be used in the same manner as a normal INPUT statement but the "device number" must be followed by a semi-colon (;) so as to distinguish the remainder of the statement. INPUT# may NOT contain prompts.

**EXAMPLE:**

```
INPUT#2;X
```

**EXAMPLE:**

```
10 PRINT#1
20 INPUT#2;X
30 IF X= 0 THEN END
40 FORI = 1 TO X
50 INPUT A$: PRINT A$
60 NEXT I
70 INPUT#0
80 PRINT#0;"DO YOU WISH TO CONTINUE ";:Y$=INCH$
90 IF Y$="Y" THEN 10 ELSE END
```

This program illustrates the use of INPUT# and also PRINT# (see page 188). Data must be provided externally to the RS232 serial port in order to run this program successfully.

X represents the number of items of data to be read. First X is read, then data is accepted from input device 2. This data is printed, as it is read, on the printer (output device 1).

After X items have been read both input and output revert back to device 0 (keyboard and VDU screen) for further instructions from the user.

## USE WITH FILES

**Syntax:** INPUT# SV,J; <variable list>

**Purpose:** This is a File-Handling Command which takes input from the file specified by the file descriptor SV, starting at the first character of the record given by J.

For information relating to the application of this command refer to the section on FILE HANDLING, page 271 of this manual.

**Related Keywords:** INCH INCH\$ PRINT PRINT#

# KEY

## KEY

**Syntax:** KEY N, string expression

Where N is the function key number and the data is the particular function to be allocated to that key

**Purpose:** The KEY command allows the user to program the 8 special "function keys" labelled F0 to F7 on the keyboard, according to individual requirements.

The key numbers are 0 to 7 in "unshifted" mode (as labelled) and 8 to 15 in shifted mode (i.e. each key can be used for two separate functions)

BASIC reserved words are stored as tokens for efficiency. ASCII characters can also be stored.

The total storage capacity for all 8 keys is 128 bytes. This can be allocated to one key or shared between the 16 functions of all 8 keys.

### EXAMPLE:

KEY 0,"LIST c/r" - this programs function key 0 to list the current program when pressed.

KEY 1,"PRINT CHR\$(A) c/r" - this programs function key 1 to print CHR\$(A) when pressed.

KEY 3,"BCOL5:TCOL15 c/r" - this programs function key 3 to set backdrop and text colours when pressed.

# RST

RST (Restart)

**Syntax:** RST

**Purpose:** This command is used to reinitialise default values in the scratch-pad as follows:-

- a) Clears the screen and sets 40 column mode.
- b) Clears the function key table.
- c) Resets the character generator, replacing the resident MOS character.
- d) Remove any sprites which may be present, and reset the backdrop colour to dark blue.
- e) Turns off the discs if they are running.
- f) Resets the PSG.
- g) Resets the default auxiliary table pointers.
- h) Resets the default WIDTH, SEP and ZONE values.

The command does NOT destroy the current BASIC program or variables. It can, therefore, be used within a program where it would otherwise be tedious to use a string of commands to ensure that the correct modes were all set up.

### EXAMPLE:

```
10 RST
20 REM START TO PROGRAM
30 - - - - -
40 - - - - -
etc.
```

**Related Keywords:**



## APPENDIX A

### LIST OF RESERVED WORDS

<b>A</b>	
ABS.....	31
ADC.....	32
AND.....	34
APPEND.....	35
ASC.....	36
ATN.....	37
AUTO.....	38
<b>B</b>	
BCOL.....	40
BEEP.....	41
BIN\$.....	42
BTN.....	43
<b>C</b>	
CALL.....	45
CHAIN.....	46
CHR\$.....	47
CLEAR.....	48
CLOSE.....	49
CLS.....	50
CONT.....	51
COS.....	52
CREATE.....	53
<b>D</b>	
DATA.....	55
DEEK.....	56
DEF.....	57
DEG.....	59
DEL.....	60
DIM.....	61
DIR.....	63
DOKE.....	65
DOS.....	66
DRAW.....	67
DRIVE.....	69
<b>E</b>	
ELLIPSE.....	70
ELSE.....	73
END.....	74
EOF.....	75
ERA.....	77
ERL.....	78
ERR.....	79
ERR\$.....	80
EVAL.....	81
EXP.....	82
<b>F</b>	
FILL.....	83
FMT.....	85
FN.....	87
FOR.....	88
<b>G</b>	
GCOL.....	94
GOSUB.....	95
GOTO.....	97

<b>H</b>	
HEX\$.....	98
HOLD.....	99
<b>I</b>	
IF.....	103
INCH.....	105
INCH\$.....	106
INP.....	108
INPUT.....	112
INPUT#.....	114
INT.....	117
IOM.....	118
<b>K</b>	
KBD.....	123
KBD\$.....	124
KEY.....	125
<b>L</b>	
LEFT\$.....	128
LEN.....	129
LET.....	130
LIST.....	132
LISTP.....	135
LN.....	136
LOAD.....	137
LOCK.....	140
LOG.....	141
<b>M</b>	
MAG.....	142
MGE.....	145
MID\$.....	147
MOD.....	148
MOS.....	149
MUL\$.....	150
MUSIC.....	151
<b>N</b>	
NEW.....	156
NEXT.....	157
NOT.....	158
NULL.....	159
<b>O</b>	
OFF.....	160
ON.....	161
OPEN.....	164
OR.....	165
ORIGIN.....	166
OUT.....	167
<b>P</b>	
PEEK.....	168
PI.....	169
PLOT.....	170
POINT.....	171
POKE.....	172
POLY.....	173
POP.....	179
POS.....	182
PRINT.....	183
PRINT#.....	188
PSG.....	191
PSW.....	192
PTR.....	194

<b>R</b>		TEMPO.....234
		THEN.....235
		TI\$.....236
		TO.....237
	<b>U</b>	
		UNLOCK.....238
		UNPLOT.....239
	<b>V</b>	
		VAL.....240
		VDEEK.....241
		VDOKE.....242
		VERIFY.....243
		VOICE.....244
		VPOKE.....246
<b>S</b>		
		WAIT.....247
		WIDTH.....252
	<b>W</b>	
		WAIT.....247
		WIDTH.....252
	<b>X</b>	
		XOR.....253
	<b>Z</b>	
		ZONE.....254
<b>T</b>		
		TAB.....230
		TAN.....231
		TCOL.....232

```

70 IF A>360 THEN A=A-360
80 PRINT@5,4;INT(A);"Degrees "
90 B=PI*A/180
100 X3=X+L*SIN(B);Y3=Y+L*COS(B)
110 DRAW X,Y TO X3,Y3:LL=0:AA=0
120 A$=INCH$:IF A$="A" THEN AA=-1 :REM ANTICLOCKWISE
130 IF A$="C" THEN AA=1 :REM CLOCKWISE
140 IF A$="L" THEN LL=8 :REM LONGER
150 IF A$="S" THEN LL=8 :REM SHORTER
160 IF A$="P" THEN PRINT#1;CHR$(27);"A";CHR$(8);PRINT#0:PRINT
CHR$(2) :REM PRINT
170 DRAW X,Y TO X3,Y3,1
180 GOTO 50

```

## THE ALL-ELECTRONIC AABBBBAA SWITCH

In a previous issue we dealt with an AABBBBAA switch for those 'too posh to push', but even this requires the pressing of a few keys to get it going so I will now make some suggestions for those too bone idle to even operate the keyboard.

But first a confession. The circuit in the last issue was obtained by combining a known, used and working circuit by Dave Arts with a 4166 circuit designed and breadboarded successfully by myself. Consequently I am confident that the two parts of the circuit work as stated. However, I have not actually built and tested the whole circuit described in its final form, for reasons of lack of time and the great difficulty I have in seeing well enough to make up such small boards. Using a magnifier all the time makes one distinctly cross-eyed, I find. I would expect that anyone attempting the construction of the project would have a fair knowledge of electronic circuits and would be able to sort out any minor problems.

Now that I feel better, let's look at some other theoretical things that I haven't tried. The Basic listing for the switch was given in the original article and is short and sweet. The equivalent machine code is even shorter:-

```
3ECF LD A,CF
```

```
D333 OUT(33),A ;configure port a
```



```

3E00 LD A,00
D333 OUT(33),A
3ECF LD A,CF
D333 OUT(33),A ;configure port b
3EC0 LD A,C0
D333 OUT(33),A ;control byte
3E01 LD A,01
D332 OUT(32),A ;data 1)
3E00 LD A,00 ; | AABB selected
D332 OUT(32),A ;data 0)
AF XOR A ;zero a
C9 RET ;return

```

That's 26 bytes for the entire routine. If BBAA is required then the data bytes would be 1F and 1E in that order. This is relocatable code as no addresses are involved, the question is, where to locate it for best advantage?

#### THE FIRST WAY.

Well, how about in the DOS track, so that it is run when the disk is booted? This is simple enough using XDOS 1, which you will have to if your drive 0 is 3".

There is unused space in the DOS where short routines may be run at bootup. This may be found at 0162H when the DOS track is read by MOS into memory at 0100H to 1B00H. If you find anything but C9 in this position it may be that some code is already in place. Way back, it was used by some members to change the drive stepping rate to suit older 5 inch drives. This will appear as 3E 02 32 B0 FB AF C9 or similar. This would alter the byte in scratchpad location FBB0H (from the default 01) to 02. If you no longer use 5" drives you can safely overwrite this area. The normal appearance here would be C9 followed by many FFs, FF being a nothing byte.

The reason the byte at 0162H is C9 is because it is the end of a routine (the bytes before it) and C9 RETURNS the action to whence it was CALLED. For a further routine such as ours to be run the C9 has to be overwritten by the first byte of the new routine AND ENDED WITH A C9 to make sure the sequence is maintained. I have tried entering our little machine code program as described into the DOS of an Xdos 1.31 disk and then booting it. It worked absolutely normally, having no discernable effect and likewise no effect on my Einstein mouse which happened to be plugged into the User Port at the time. As I have nothing else I can use on the port, I cannot, of course, tell whether anything WOULD have happened, but I have some

plans for a simple LED display for this purpose. But don't hold your breath. There was adequate information in the original article for anyone with some electronic experience to make up something similar, so how about it? 5 LEDs to show the state of the D0, D1, D2, D3 and D4 lines. Any offers?

Before I give a key by key 'how to do it' on the above it will be interesting to discuss what we are about to achieve.

If we assume that we have four drives configured 0 and 1, 3" (40 track, singlesided) and 2 and 3, 3.5" (80 track, doublesided) and the electronic switch featured in the last issue fitted to the User Port, how are we going to operate everything?

At switchon the system will have a 3" drive as drive 0, so to boot up, a 3" disk will be required. If the DOS has the added routine at 0162H with the data bytes 1F and 0, then the status will not change. If the DOS routine has the data bytes 1FH and 1EH then the switch will operate and the 80 track DS drive 2 will be switched to Drive 0, with corresponding changes in the other drives, to give the BBAA setup. Therefore, to change from this configuration back to the original, a DD disk will need to be prepared with the appropriate DOS track modification, because you can only boot from a drive 0. You could also just switch OFF and then reboot with a normal 3" disk.

On the face of it then the bare minimum to get by will be one 3" disk with modified DOS using the data bytes 1FH and 1EH, which when booted will switch to the BBAA mode. The AABB mode can be acquired by switching off and booting with a standard 3" Xdos 1.31 disk. This is just as well as DOS 3 on DD disks has the DOS track byte 0162H already used for a short routine to make the disk compatible with Xdos 2, so you won't want to change that.

Right then, let's do it. Select a nice blank formatted 3" disk and put the computer into MOS. The easy way is just to switch ON with no disk in any drive. (You always do this don't you? Not wanting to risk corrupting a disk, like?) NOW, put the nice new formatted disk in drive 0 and type at the cursor (flashing arrow) R 100 1B00 and press ENTER. Note, the 0s are ZEROs, and the second space is vital. The drive LED will light and go out and nothing much else, but never fear, you have loaded the first part of DOS into memory at 0100H. MOS overlooks missing leading zeros.

To see what the DOS looks like type T 100 1B0 and press ENTER. The screen will fill with columns of digits and on the right columns of ASCII characters, some of which will make sense. The four digit block at the left are memory addresses starting at 0100H. Look down this list until you find 0160 and three pairs of characters to the right of this you should find C9, as explained above. If there is any other byte you will have to decide whether to overwrite it or try reformatting the disk using your Master System disk, which should be in original condition, and try the above again.



If all seems well, now try entering the 26 bytes of code above starting at 0162H.

Type M 162 and press ENTER. the address will appear with C9 under the cursor. Type in the code in a continuous line and end with a fullstop. Press ENTER. To see that all is well type T100 180 as before and you should see that the 26 bytes of code are in the display. Check that they are correct, and when satisfied type W 100 1B00 which will write the modified code back to the disk DOS track.

Label the disk carefully so that you know exactly what it is and try booting it. If you configured it to switch to BBAA, just select Drive 0 and check that the expected one operates.

### THE SECOND WAY.

The idea for the second way seems to be more trouble than it is worth, as you will see, but I include it for completeness. It involves loading the code into spare memory in the Video Ram, which exists at the address &3F80 and ends at &4000.

The big advantage of this memory area is that it is not cleared when using the reset button at the rear of the machine. To use it requires a 1 line program to load the code :-

```
10 VPOKE &3FC0,&CF,&CF,&56,&52,&41,&CD,&C9
```

There is no direct way to access and run this code so another short XBAS program is needed:-

```
10 CLS
```

```
20 CLEAR &8000
```

```
30 FOR J=1 TO 7
```

```
40 A=VPEEK(&3FBF +J)
```

```
50 POKE &7FFF+J,A
```

```
60 NEXT
```

```
70 CALL &8000
```

```
80 CLEAR &E9FF:END
```

This peeks the code in vram and pokes it into normal memory and runs it. The Clear &E9FF is to clear main memory for future use, and is actually top of memory for XBAS 5. This avoids an error being generated if using XBAS 5 as XBAS 4 memory is a few bytes higher, at &EBFF

Save prog 1 as VRAM.XBS and 2 as BBAA.XBS. If you now RUN them one after the other you will get a surprise, the word VRAM will be printed on the screen. Just my little joke really, but I don't see why we can't all join the fun!

If you want to use this for switching your drives you will have to make a few changes. In prog 1, after the first comma, insert the 26 bytes of code referred to above, overwriting the other code, (which is m/c for PRINT "VRAM"). In prog 2 line 30 make J =1 to 26 to accomodate the longer code string.

And that's it! My goodness we have come a long way from those hard-to-use push switches, haven't we? Science gone mad maybe, but then, What's New? My hope is that someone will read this and think "just what I need to operate my idea for an electronic scollap linger!" Please send the resulting article on disk. END.

## PULL THE OTHER ONE - ITS GOT AN EINSTEIN ON IT!

by Bob Deeley

At the end of your tether? Then you need this extra string to pull - an hour spent on this, the simplest of Albert's add-ons, and you will feel the flush of success!

The Einey's Reset button is tucked out of harms way at its rear but awkward to use, of course one could solder a pair of wires to the switch's circuit board pads and fit another switch somewhere at the front, but it would likely as not be pressed by mistake. Two switches either side, wired in series, needing to be pressed together, would prevent accidental resetting. However I judged this more complex to install, inefficient to use, requiring at least a two-finger action; and it has been done before anyway.

These thoughts, combined with the fact that the lid has been off my Einey for some time due to playing with disk drives... generated some horizontal cranial activity resulting in this entirely non-electrical modification which I have found to be quite satisfactory. The only drawback I can think of would be inquisitive children, and the not so young, seeing what is poking out of the speaker grill while one is diverted from the keyboard a few moments. Of course, they will need to be warned off, not easy though. I suppose everything must have its limitations.

I will leave you to come up with ideas for the pull handle, something rustic perhaps. I found a tiny wooden bobbin, but a shirt button would suffice.

### Ingredients:

- (1) 1off 400mm nylon string (kite string)
- (2) 1off 13mm diameter x 250mm long plastic or cardboard tube.
- (3) 1off standard paperclip



- (4) 1 off large paperclip or piece of wire 130mm long.
- (5) 1 off light duty tension spring 50 - 100mm long, diameter must be a free fit inside the tube.
- (6) 1 off cable tie
- (7) 1 off small ring pull or knob of choice.

### Tools:

Small screwdrivers flat and crosspoint.

Snipe nose pliers with cutter

Hand drill with 2mm twist drill.

### Method of construction:

\* Switch off everything, remove the Einstein Computer's power plug from the socket, don't return it until the cover has been replaced at the end of this project.

\* Remove two screws at the rear of Albert and release the cover, find the reset switch from the inside and with a small screwdriver near the edge of the gray button cap, push while steadying with the other hand from outside to remove it.

\* Take the paperclip (3) and straighten it out until only the inner loop remains, offer this loop to the inside of the cap you have just removed, to make sure it is a snug fit.

\* Follow the diagrams to finish forming and cutting the paperclip, and preparing the other items.

\* Fit the paperclip back into the reset button cap. You will need to turn it sideways to pass it through the small slot above the switch plunger into the case, then align it to push the button back onto the switch plunger. The operate loop should lie central and flush to the switch body. Check for freedom of movement pressing the button a few times. If it sticks or is sluggish, remove and re-bend the fouling part of the paperclip.

\* With a 2mm drill, make a hole on the left side of the speaker grill 20mm in from the edge, in its top slot. This allows just enough space for a finger or thumb by the recess for gripping.

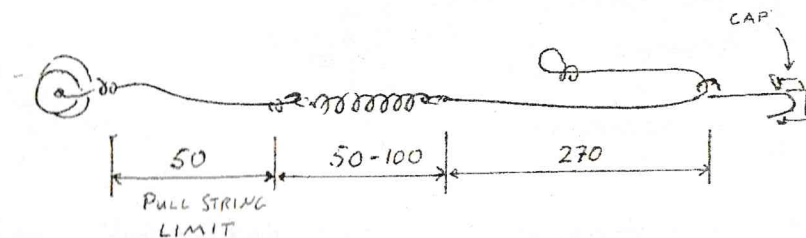
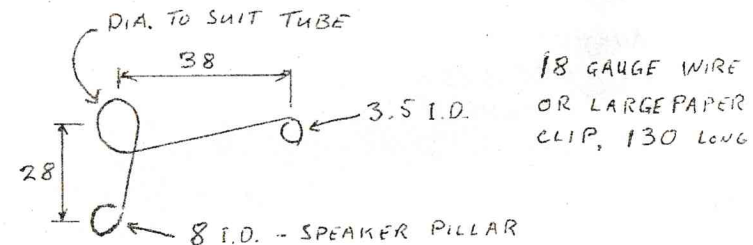
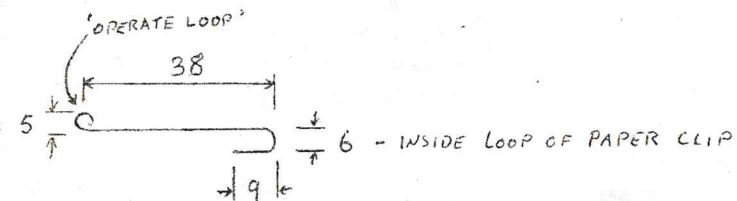
\* Assemble the tube support clip (4) as in the diagram, cut off 100mm of string (1), tie the strings to each end of the spring (5). By the way, don't use granny knots anywhere, a round turn and two half hitches is best. Pass the short end through the support clip and hole and attach to your chosen pull knob (7) at the correct length of 50mm. Trim the excess string.

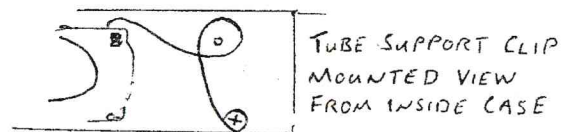
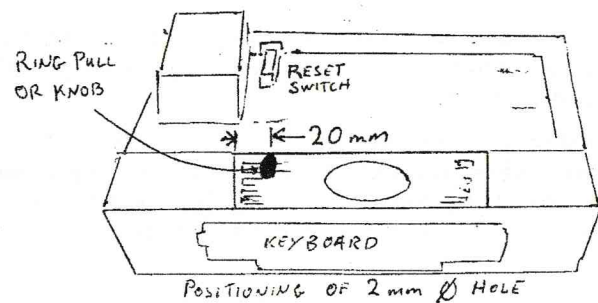
\* Thread the other end of the string through the tube (2). I loosely tied a loop at the end and gently blew it through. Draw in the spring and mount the tube into the support clip, push up close to the inside of the fascia and

anchor the other end to the power supply casing with the cable tie (6). Don't trim the tie.

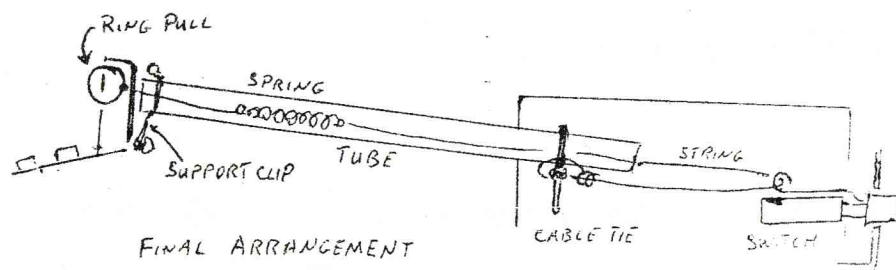
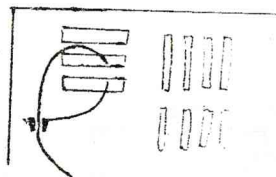
\* Pass the string through the paperclip operate loop, it is to be attached to the protruding end of the cable tie. Pull back the tension just enough without depressing the reset button and tie off.

\* Observe that the rest should now operate after 10 to 25mm of the string is pulled and fully return when released. The spring prevents any strain on the switch, with the overpull stopped at the spring knot against the hole. Happy? Then replace the case cover plus screws and fire up.





CABLE TIE FOR  
REAR END OF TUBE  
ANCHOR.  
PASSED THROUGH  
POWER SUPPLY  
VENTILATION SLOT



Thanks Bob, for a really novel idea. It just shows that there **is** something new under the sun! I will bet there are plenty more 'personal' modifications out there and we want to hear about them. Don't forget that every article published gets you a free issue of the magazine. Ed.



TONY'S BACK PAGE TIDY UP

OUR ILLUSTRIOUS EDITOR, TED CAWKWELL, HAS HAD MAJOR HEALTH PROBLEMS

which have made it very difficult for him to give his full attention to editing your magazine input -- YOU HAVE SENT SOMETHING IN FOR HIM TO EDIT, HAVEN'T YOU ??? -- and I'm sure it would boost his morale and his recovery no end if you sent him a card or a note, with your best wishes and your appreciation of his valiant efforts on YOUR behalf.

DID YOU GET YOUR FREE MEMBERSHIP ADVICE WITH THIS MAGAZINE ISSUE ???

You did if you contributed an article, letter, listing, etc. for us to include in this issue. If not, YOU KNOW HOW TO GET ONE, DON'T YOU ???

We try to compile your active input into a magazine of mutual interest that we mail out 6 times a year to every member of the user group. If you want its pages to be full of interesting facts and useful ideas,

IT'S UP TO YOU TO CONTRIBUTE SOMETHING REGULARLY !!!

All magazine input to Ted Cawkwell, Editor, address on front cover

All MEMBERSHIP ENQUIRIES/SUBSCRIPTIONS to Ivy Cottage, Church Road, New Romney, Kent, TN28 8TY, enclosing SAE. The membership year is six mailings, and membership subscriptions are on a "Bulk Buy" discounted sliding scale, so the more you buy, the cheaper it gets. i.e.

Membership Years:	1	2	3	4	5
Magazine Issues:	6	12	18	24	30
Subscription in £ :	10	18	24	28	30

Outside the UK add £ 3 per year

Please make cheques payable to EINSTEIN USER GROUP

\*\*\* MEMBERS' SALES/WANTS:- We'll include details with our mailings if space permits and if you send them to New Romney, but please send Steve Potts a copy too, as he tries to match up what's wanted with what's available, and he may have potential members in need of kit.

MAGAZINE BACK NUMBERS are 50p each (postfree), or any 12 (your choice) for £ 5.00, or only £ 20.00 for a copy of every issue from Einstein Monthly 1/4 onwards. Orders to New Romney.

SHARP USER CLUB MAGAZINE:- All issues available. Also most Sharpsoft magazines. Send 2 stamps for details and prices from SUC or New Romney.

\*\*CABLES/CONNECTORS/ROMS:- Are produced to order by Stuart Marshall. 01827-897920 or send SAE to 25 CARLCROFT, STONYDELPH, TAMWORTH, B77 4DL.

\*\*OTHER USEFUL BITS:- Contact Steve Potts, address on front cover.

\*\*OTHER SOURCES:- B & H COMPUTERS, BEACON BUSINESS CENTRE, SOUTHWORAM, HALIFAX, HX3 5UA, phone 01422 330408. \*\*\* IF YOU TRY THIS FIRM WITH EINSTEIN ENQUIRIES/ORDERS, PLEASE LET US KNOW HOW SATISFIED YOU ARE \*\*

\*\*\* EINSTEIN USER MANUALS:- We have a few spare copies of Introduction To Einstein, DOS/MOS Introduction, and BASIC Reference Manual

\*\*\* ALSO AVAILABLE:- Power Graphics For The Einstein, Memotech & MSX; VDP Discovered (Advanced Reference Manual) for the Einstein, Memotech and MSX; Basic Repair Manual For The Commodore 64 Computer.

Contact New Romney. Prices negotiable.