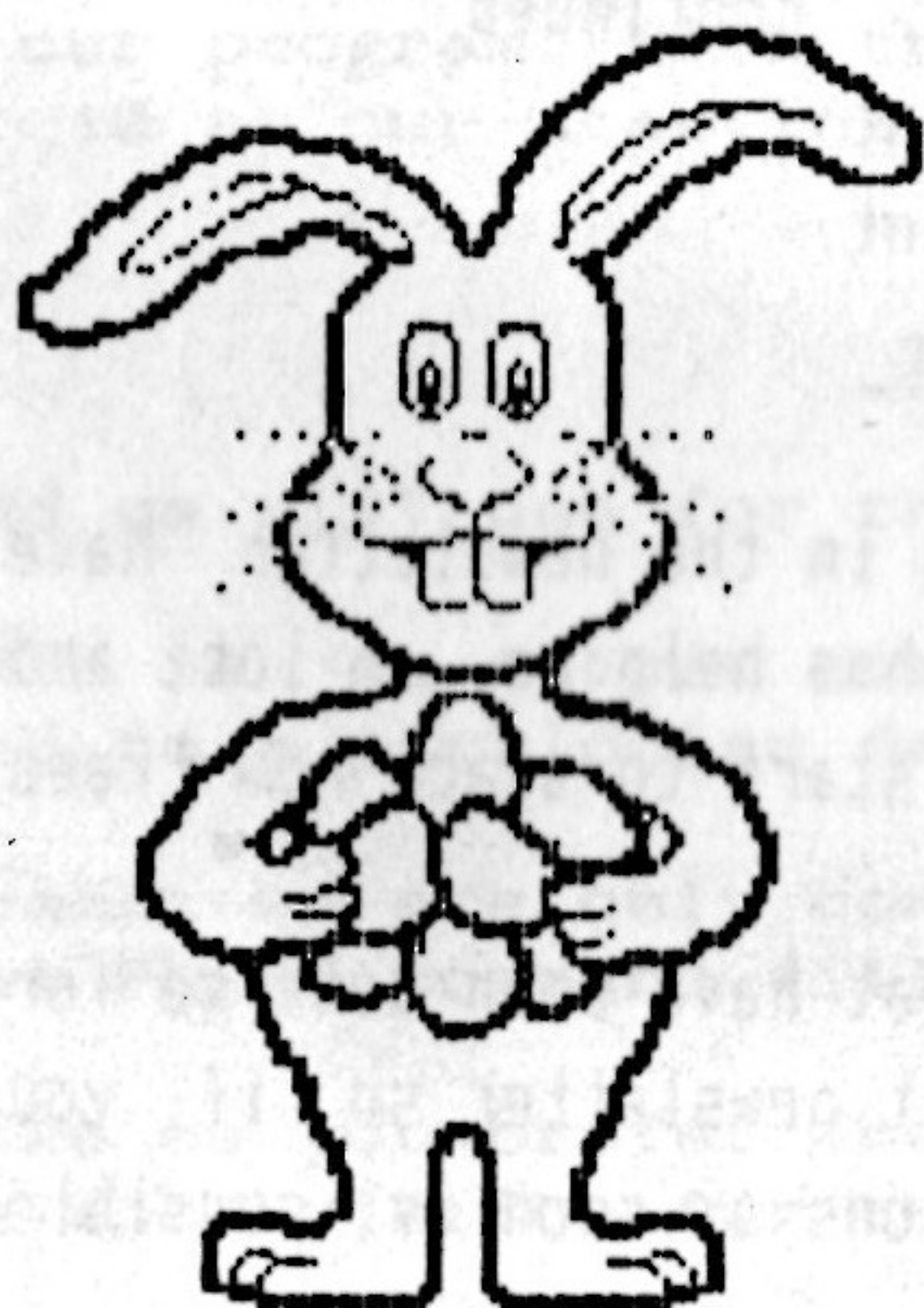


# Avon Einstein User Group



*Easter*  
With  
Albert !

**NOW SHOWING**

Vol 3 . No 2.



# PAGE INDEX

=====

- (1).....:= BASIC FOR BEGINNERS (Part2)
- (2).....:= " " " continued
- (3).....:= MACHINE CODE PROGRAMMING
- (4).....:= " " by J.NASH
- (5).....:= " " continued
- (6).....:= " " continued
- (7).....:= MNEMONIC/HEX TABLE
- (8).....:= " " continued
- (9).....:= " " continued
- (10).....:= " " continued
- (11).....:= TURBO-PASCAL AND ITS USE
- (12).....:= " " by P.SYMONDS
- (13).....:= " " continued
- (14).....:= " " continued
- (15).....:= advertisement
- (16).....:= advertisement

\*\*\*\*\*

Well for a change this month all the articles in the newsletter have been supplied by our own members !!! This has help me a lot and proves that you can do it if you try so don't start to slack now keep sending in those articles to me.

I am starting to get together some letters that have been sent me for a letters page which I hope to put in the next newsletter so if you have anything to ask then send me your questions as soon as possible and I will include them.

\*\*\*\*\*

Don't forget the next club meeting is on the 9th of MARCH

starting at 8.00pm at the

BLACK HORSE, WEST STREET, OLD MARKET, BRISTOL.

PLEASE TRY AND COME

\*\*\*\*\*  
(front cover designed and supplied by M.MADDOCKS. using an SBC XT and FIRST PUBLISHER)

-1-

## Programming in BASIC for Beginners - Part 2

Let us suppose that we need to write a program that will display the words TATUNG EINSTEIN on the screen. To do this we will need to tell the computer three things.

- (a)...A Line Number.
- (b)...A Command.
- (c)...What we want Displayed.

### The Line Number

Remember that a BASIC program begins operation at the lowest line number, it makes sense to pick a low number for the start of the program although in theory the number could be anything between 1 and 65,535. Once we have decided upon a starting number it also makes sense to number the following lines in multiples of ten. This will allow us to insert new lines at a later time should we wish to amend our program. With these two factors in mind, I have chosen the number 10 as our starting line number.

### The Command

The command we will use for this program is the Reserved Word PRINT.

PRINT is an output command and causes information/data to be transferred to a particular device such as screen, printer, disc drive etc. For the moment we are only concerned with output to the screen. Other uses of the PRINT command will be dealt with in later articles.

If like me you prefer the easy life, the question mark (?) is an acceptable abbreviation for the PRINT command.

If the PRINT command is used on its own ie. :-

10 PRINT

The result when the program is RUN will be a carriage return and a line feed.

What follows the PRINT command will be considered as a VARIABLE NAME when the program is RUN if the character(s) following the PRINT command are not enclosed in inverted commas ("). This would result in the numeric VALUE of the VARIABLE being printed. This will be explained in a later article.



### The Words

The actual characters that we want printed on the screen, (TATUNG EINSTEIN), must be enclosed with inverted commas ("), so we should have a program line which looks like this.:-

```
10 ? "TATUNG EINSTEIN"
```

If we now press the ENTER key the program line will be stored in RAM.

When the program is typed in it is stored in RAM until either it is cleared or the power is turned off. To input another program a special command must be used which clears the existing program from memory. If this is not done the new program will overlap the old program and sections of each will intermingle depending on line numbers. Typing the NEW command provides this facility.

To ensure that our program line has been entered correctly, type LIST <ENTER>. The computer should respond with. :-

```
10 PRINT "TATUNG EINSTEIN"
```

Note that the question mark that we originally entered has been replaced with the word PRINT.

All that remains to be done is to tell the computer that after line 10 has been executed the program ends. To do this we type. :-

```
20 END <ENTER>
```

If we now type LIST <ENTER> we should see. :-

```
10 PRINT "TATUNG EINSTEIN"
20 END
```

END indicates the finishing line of the program involved. In complex programs the END statement is not necessarily the final statement in the listing, however in less complex programs END is usually the last statement and may be omitted.

Now is the moment of truth!, typing the command RUN <ENTER> will cause the program to be executed and the words TATUNG EINSTEIN should appear on the screen.

### Summary

Line numbers must be within the range of 1 to 65,535. Usually the lowest line number is 10, the following lines being in multiples of 10.

The Question Mark (?) may be used as an abbreviation for the PRINT command.

The PRINT command may be used to display the VALUE of VARIABLES or to display particular characters which are enclosed in inverted commas (").

```
*****
*                                     *
*   Machine Code Programming       *
*                                     *
*****
```

So you want to learn machine code programming, WHY?

- ( 1 ).... Its dam difficult to debug & almost impossible to read
- ( 2 ).... Mathematical calculations labourious to perform
- ( 3 ).... More program instructions than BASIC

Now on the plus side we have.

- ( 1 ).... Program's run upto 60 times faster
- ( 2 ).... They take up less space in the memory
- ( 3 ).... Operations can be performed which cant be done in BASIC

So with this in mind I decided that the end result would be well worth the time and patience needed to make the transition from BASIC to MACHINE LANGUAGE.

I intend to cover all the Z80 CPU INSTRUCTIONS. As you can see by the set listed there are lots of them. But dont be daunted by this as they are in groups, all instructions in the group perform the same operation as the others except with a different register.

In each month of the AEUG MAG we will look at one or two of these groups of instructions and relate these to the BASIC equivalent. Also whilst exploring these instructions we will write short routines to give you the understanding of the machine as well as the machine language.

All programs will relate to the screen in 40 col MODE, as this tutorial is graphic orientated and not everyone will have a 80 col card. If you are interested in learning machine code you are going to need an assembler. The one I am using is the Glen Assembler, there are several different ones in the group. Ask Graham for one from the Public Domain. Failing this just give me a bell and I'll try and fix you up with something. ( It is not critical to have an assembler to write machine code but life is much easier with one ). In this months tutorial we will take things very simply and make do without one.

One thing to remember when writing program's be it machine code or BASIC, does it do what you want it to do. Dont worry that it is not the most elegant or economical way it could have been written or that someone else could have written it in a more professional manner. All this comes with experience. I will show you examples in this series that will be deliberately written longer to show the use of more instructions. Also how some times by using more instructions you can produce a faster section of code.

The examples and routines we will write each month can and should be experimented with at your leisure. Remember you can not harm your machine, and at worst you can cause the system to crash ( we've all done that ), just switch off and start again. Make backups and use write protection when experimenting. The Einstein has some built in routines which we will use. A list of these has been published in the Einstein Mags, I will explain each one and how to use it as we progress.

O.K LETS START !



# THE REGISTERS

First off. In machine language there are no variables as we know them in BASIC. The nearest equivalent we have are the registers. Unfortunately we only have 7 of these to play with, there are a few others but these have special functions. So we will concern ourselves with these 7 for the time being. They are as follows.

A,B,C,D,E,H,L

Each register can hold a number between 0-255

Now as you can see from this our options seem very limited, but all is not lost as 6 of these registers can be put together, this is known by the term REGISTER PAIR ( obvious isn't it ). The register pairs are arranged as follows.

HL, BC, DE

Each register pair can hold a number between 0-65535

If we take the first pair HL these two registers H and L were paired together to remind us that there is a HIGH and LOW part of the pair. This is how we can reach a number of 65535. As an example we will say the HL register holds the number 12674. The high part is held by the H register and can be found by dividing 12674 by 256 and rounded down ( 12674/256 = 49 and 130 left over ) the remainder 130 is what is held in the L register. Therefore H holds 49 L holds 130. The same is true of the BC pair where B is the high part and in the DE pair where D is the high part.

( HIGH \*256 + LOW = TOTAL VALUE OF PAIR.)

O.K. how do we get numbers into these registers? Its quite simple really, look at the list of op codes in the list I have prepared for you and you will see many instructions, all of these instructions manipulate the register in one way or another.

## WAYS OF ADDRESSING THE REGISTERS

- |                                    |                        |
|------------------------------------|------------------------|
| 1.... Immediate addressing         | LD r,n                 |
| 2.... Register addressing          | LD r,r                 |
| 3.... Register indirect addressing | LD (rr),r or LD r,(rr) |
|                                    | LD (HL),n              |
| 4.... Extended addressing          | LD A,(nn) or LD (nn),A |
| 5.... Index addressing             |                        |

No.5 uses the INDEX REGISTERS we will deal with these registers later.

In the above r = register n = number 0-255

rr = register pair nn = number 0-65535

any other indicates only that register may be used

The brackets used mean the contents of. For example LD (rr),r could be LD (HL),A meaning load the address contained in HL with the value in the A register. If this was executed and the HL register pair held then number 50000 and the A register held 231 the the value 231 would be put into the memory location 50000. So in BASIC this would be POKE 50000,231.

This is just a brief introduction into the register, the way in which we manipulated them is MACHINE CODE.

As we progress we will look at all the ops and go deeper into things one simple step at a time. Each time we will have working examples some totally useless and hopefully some usefully routines.

In next months MAG we will write a 100% machine code program to load up your FUNCTION KEYS from DOS.

## EXAMPLE No. 1

As this example is the first and we have alot introductory material I will keep it short and sweet.

## INSTRUCTIONS USED WILL BE

LD A,n .. which means Load the A register with a number  
The nearest BASIC instruction would be A=n  
LD B,n .. this is another LOAD instruction with the B register  
Again in BASIC it would read B=n  
RST 8 .. this means ReStart 8 it has no BASIC equivalent and this RST will tell the machine to expect a MCALL, this means the next BYTE must be the MCALL number. In this example we will be calling the print a letter routine which is MCALL 198 or 9E hex.  
INC A .. which means INCRiment the A register by one  
In BASIC it would be A=A+1  
DJNZ .. which means Decrement the B register and if the B register is Not Zero and Jump Relative to memory address.  
In BASIC it would be B=B-1 : IF B<>0 THEN GOTO line No.n  
RET .. RETURN this lets the CPU know its finish what you want it to and RETURN control to you. This is a must or you will just have to switch off and start again. There is no BASIC Equivalent.

addr	op code	nmemonic	basic equivalent
0100	3E 41	LD A,41	: A=&41 ( 65 dec )
0102	06 1A	LD B,1A	: B=&1B ( 26 dec )
0104	CF	RST 8	: none
0105	9E ( this is the MCALL No. to print the CHR\$ in A reg)		
0106	3C	INC A	: A=A+1
0107	10FB	DJNZ	: B=B-1 IF B<>0 THEN
			: GOTO addr 0104
0109	C9	RET	

This program will print the alpabet ( big deal ), this is how its done. First Load up the A register with 41 hex / 65 dec as we all know this is the character code for the letter A. Then Load up the B register with 1B hex 26 dec. Then the number of letters to print. The B register is normally used as a counter because it is a favoured register for this purpose. It suits other op's as we will see in later examples. But this does not prohibit you from using other registers for counting. O.K we now do a ReStart in this case it is a ReStart 8 which tells the computer we are going to use one of its own routines, in the case we want to print a character. The routine to print a character is 9E hex. This M/CALL BYTE must follow the RST 8. M/CALL 9E hex prints the character held in the A register. Initially the A register has been Loaded with 41 hex, it does not need much imagination to find that the letter A will be printed. Right the letter A has been printed, what next? Well we INCRiment the A register ( A=A+1 ). No prizes for telling me that the A register now holds 42 hex and that is the character B. The next instruction is a little more complex as really it is a small routine of its own. It's DJNZ which is like the NEXT in a BASIC FOR / NEXT LOOP. Firstly it decrements the B register ( B=B-1 ) then it looks at the ZERO FLAG ( more on FLAGS in future lessons ), if the FLAG shows NZ ( NOT ZERO ) then the program JR ( Jumps Relative ) n BYTES. See the RELATIVE JUMP TABLE I have



prepared for a more details explanation of this. It is now apparent that in running through this loop that the A register will be incremented 26 times, as this is the value held in the B register therefor the complete alphabet will be printed by the time the B register will reach zero and the last instruction will be executed ant that is RETURN which terminates the program and RETURNS back to the operating system.

Now look up the op codes we have used in the list of operating instructions, put them together and it will look like this.

3E 41 06 1A CF 9E 3C 10 FB C9.

THIS IS YOUR MACHINE CODE FOR THE PROGRAM ABOVE.

Go into MOS and type M0100 you will see the cursor flashing on the character to be MODIFIED. Type in all the hex numbers above without the spaces but include the full stop, then press ENTER. Now type Y then ENTER this will give you a WARM start back into DOS. Type GO then ENTER and WOW! the whole alphabet is printed. Because we are feeling totally elated we will spare no expence and actually SAVE this MEGA masterpiece as a COM FILE. TYPE SAVE 1 MCODE1.COM  
Go on be a devil and try typing MCODE1 and see if it works.

THE RELATIVE JUMP TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8 :	-128	-127	-126	-125	-124	-123	-122	-121	-120	-119	-118	-117	-116	-115	-114	-113 :
9 :	-112	-111	-110	-109	-108	-107	-106	-105	-104	-103	-102	-101	-100	-99	-98	-97 :
A :	-96	-95	-94	-93	-92	-91	-90	-89	-88	-87	-86	-85	-84	-83	-82	-81 :
B :	-80	-79	-78	-77	-76	-75	-74	-73	-72	-71	-70	-69	-68	-67	-66	-65 :
C :	-64	-63	-62	-61	-60	-59	-58	-57	-56	-55	-54	-53	-52	-51	-50	-49 :
D :	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39	-38	-37	-36	-35	-34	-33 :
E :	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17 :
F :	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1 :
0 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 :
1 :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 :
2 :	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47 :
3 :	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63 :
4 :	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79 :
5 :	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95 :
6 :	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111 :
7 :	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127 :

MNEMONIC	HEX	MNEMONIC	HEX	MNEMONIC	HEX
LD L,H	6C	: RES 1,(HL)	CB8E	: RES 6,L	CB85
LD L,L	6D	: RES 1,(IX+dis)	DDCBxx8E	: RES 7,(HL)	CB8E
LD R,A	ED4F	: RES 1,(IY+dis)	FDCBxx8E	: RES 7,(IX+dis)	DDCBxx8E
LD SP,(addr)	ED7Bxxxx	: RES 1,A	CB8F	: RES 7,(IY+dis)	FDCBxx8E
LD SP,nn	31xxxx	: RES 1,B	CB88	: RES 7,A	CB8F
LD SP,HL	F9	: RES 1,C	CB89	: RES 7,B	CB88
LD SP,IX	DDF9	: RES 1,D	CB8A	: RES 7,C	CB89
LD SP,IY	FDF9	: RES 1,E	CB8B	: RES 7,D	CB8A
LDD	EDA8	: RES 1,H	CB8C	: RES 7,E	CB8B
LDDR	EDB8	: RES 1,L	CB8D	: RES 7,H	CB8C
LDI	EDA0	: RES 2,(HL)	CB96	: RES 7,L	CB8D
LDIR	EDB0	: RES 2,(IX+dis)	DDCBxx96	: RET	C9
NEG	ED44	: RES 2,(IY+dis)	FDCBxx96	: RET C	D8
NOP	00	: RES 2,A	CB97	: RET M	F8
OR (HL)	B6	: RES 2,B	CB90	: RET NC	D0
OR (IX+dis)	DDB6xx	: RES 2,C	CB91	: RET NZ	C0
OR (IY+dis)	FDB6xx	: RES 2,D	CB92	: RET P	F0
OR A	B7	: RES 2,E	CB93	: RET PE	E8
OR B	B0	: RES 2,H	CB94	: RET PO	E0
OR C	B1	: RES 2,L	CB95	: RET Z	C8
OR D	B2	: RES 3,(HL)	CB9E	: RETI	ED4D
OR n	F6xx	: RES 3,(IX+dis)	DDCBxx9E	: RETN	ED45
OR E	B3	: RES 3,(IY+dis)	FDCBxx9E	: RL (HL)	CB16
OR H	B4	: RES 3,A	CB9F	: RL (IX+dis)	DDCBxx16
OR L	B5	: RES 3,B	CB98	: RL (IY+dis)	FDCBxx16
OTDR	EDBB	: RES 3,C	CB99	: RL A	CB17
OTIR	EDB3	: RES 3,D	CB9A	: RL B	CB10
OUT (C),A	ED79	: RES 3,E	CB9B	: RL C	CB11
OUT (C),B	ED41	: RES 3,H	CB9C	: RL D	CB12
OUT (C),C	ED49	: RES 3,L	CB9D	: RL E	CB13
OUT (C),D	ED51	: RES 4,(HL)	CBA6	: RL H	CB14
OUT (C),E	ED59	: RES 4,(IX+dis)	DDCBxxA6	: RL L	CB15
OUT (C),H	ED61	: RES 4,(IY+dis)	FDCBxxA6	: RLA	17
OUT (C),L	ED69	: RES 4,A	CBA7	: RLC (HL)	CB06
OUT (C),n	D3xx	: RES 4,B	CBA0	: RLC (IX+dis)	DDCBxx06
OUTD	EDAB	: RES 4,C	CBA1	: RLC (IY+dis)	FDCBxx06
OUTI	EDA3	: RES 4,D	CBA2	: RLC A	CB07
POP AF	F1	: RES 4,E	CBA3	: RLC B	CB00
POP BC	C1	: RES 4,H	CBA4	: RLC C	CB01
POP DE	D1	: RES 4,L	CBA5	: RLC D	CB02
POP HL	E1	: RES 5,(HL)	CBAE	: RLC E	CB03
POP IX	DDE1	: RES 5,(IX+dis)	DDCBxxAE	: RLC H	CB04
POP IY	FDE1	: RES 5,(IY+dis)	FDCBxxAE	: RLC L	CB05
PUSH AF	F5	: RES 5,A	CBAF	: RLCA	07
PUSH BC	C5	: RES 5,B	CBA8	: RLD	ED6F
PUSH DE	I5	: RES 5,C	CBA9	: RR (HL)	CB1E
PUSH HL	E5	: RES 5,D	CBAA	: RR (IX+dis)	DDCBxx1E
PUSH IX	DDE5	: RES 5,E	CBAB	: RR (IY+dis)	FDCBxx1E
PUSH IY	FDE5	: RES 5,H	CBAC	: RR A	CB1F
RES 0,(HL)	CB86	: RES 5,L	CBAD	: RR B	CB18
RES 0,(IX+dis)	DDCBxx86	: RES 6,(HL)	CB86	: RR C	CB19
RES 0,(IY+dis)	FDCBxx86	: RES 6,(IX+dis)	DDCBxx86	: RR D	CB1A
RES 0,A	CB87	: RES 6,(IY+dis)	FDCBxx86	: RR E	CB1B
RES 0,B	CB80	: RES 6,A	CB87	: RR H	CB1B
RES 0,C	CB81	: RES 6,B	CB80	: RR L	CB1C
RES 0,D	CB82	: RES 6,C	CB81	: RRA	1F
RES 0,E	CB83	: RES 6,D	CB82	: RRC (HL)	CB0E
RES 0,H	CB84	: RES 6,E	CB83	: RRC (IX+dis)	DDCBxx0E
RES 0,L	CB85	: RES 6,H	CB84	: RRC (IY+dis)	FDCBxx0E



MNEMONIC	HEX	MNEMONIC	HEX	MNEMONIC	HEX
RRC A	CBOF	: SET 2,D	CBD2	: SLA C	CB21
RRC B	CBOB	: SET 2,E	CBD3	: SLA D	CB22
RRC C	CBO9	: SET 2,H	CBD4	: SLA E	CB23
RRC D	CBOA	: SET 2,L	CBD5	: SLA H	CB24
RRC E	CBOB	: SET 3,(HL)	CBDE	: SLA L	CB25
RRC H	CB0C	: SET 3,(IX+dis)	DDCBxxDE	: SRA (HL)	CB2E
RRC L	CB0D	: SET 3,(IY+dis)	FDCBxxDE	: SRA (IX+dis)	DDCBxx2E
RRCA	OF	: SET 3,A	CBDF	: SRA (IY+dis)	FDCBxx2E
RRD	ED67	: SET 3,B	CBD8	: SRA A	CB2F
RST 00	C7	: SET 3,C	CBD9	: SRA B	CB28
RST 08	CF	: SET 3,D	CBD9	: SRA C	CB28
RST 10	D7	: SET 3,E	CBD8	: SRA D	CB29
RST 18	DF	: SET 3,H	CBDC	: SRA E	CB2A
RST 20	E7	: SET 3,L	CBDD	: SRA H	CB2B
RST 28	EF	: SET 4,(HL)	CBE6	: SRA L	CB2C
RST 30	F7	: SET 4,(IX+dis)	DDCBxxE6	: SRL (HL)	CB2D
RST 38	FF	: SET 4,(IY+dis)	FDCBxxE6	: SRL (IX+dis)	DDCBxx3E
SBC A,(HL)	9E	: SET 4,A	CBE7	: SRL (IY+dis)	FDCBxx3E
SBC A,(IX+dis)	DD9Exx	: SET 4,B	CBE0	: SRL A	CB3F
SBC A,(IY+dis)	FD9Exx	: SET 4,C	CBE1	: SRL B	CB38
SBC A,A	9F	: SET 4,D	CBE2	: SRL C	CB39
SBC A,B	9B	: SET 4,E	CBE3	: SRL D	CB3A
SBC A,C	99	: SET 4,H	CBE4	: SRL E	CB3B
SBC A,D	9A	: SET 4,L	CBE5	: SRL H	CB3C
SBC A,n	DExx	: SET 5,(HL)	CBEE	: SRL L	CB3D
SBC A,E	9B	: SET 5,(IX+dis)	DDCBxxEE	: SUB (HL)	96
SBC A,H	9C	: SET 5,(IY+dis)	FDCBxxEE	: SUB (IX+dis)	DD96xx
SBC A,L	9D	: SET 5,A	CBEF	: SUB (IY+dis)	FD96xx
SBC HL,BC	ED42	: SET 5,B	CBE8	: SUB A	97
SBC HL,DE	ED52	: SET 5,C	CBE9	: SUB B	90
SBC HL,HL	ED62	: SET 5,D	CBEA	: SUB C	91
SBC HL,SP	ED72	: SET 5,E	CBE8	: SUB D	92
SCF	37	: SET 5,H	CBEC	: SUB E	93
SET 0,(HL)	CBC6	: SET 5,L	CBED	: SUB H	94
SET 0,(IX+dis)	DDCBxxC6	: SET 6,(HL)	CBF6	: SUB L	95
SET 0,(IY+dis)	FDCBxxC6	: SET 6,(IX+dis)	DDCBxxF6	: XOR (HL)	AE
SET 0,A	CBC7	: SET 6,(IY+dis)	FDCBxxF6	: XOR (IX+dis)	DDAExx
SET 0,B	CBC0	: SET 6,A	CBF7	: XOR (IY+dis)	FDAExx
SET 0,C	CBC1	: SET 6,B	CBF0	: XOR A	AF
SET 0,D	CBC2	: SET 6,C	CBF1	: XOR B	A8
SET 0,E	CBC3	: SET 6,D	CBF2	: XOR C	A9
SET 0,H	CBC4	: SET 6,E	CBF3	: XOR D	AA
SET 0,L	CBC5	: SET 6,H	CBF4	: XOR n	EExx
SET 1,(HL)	CBCE	: SET 6,L	CBF5	: XOR E	AB
SET 1,(IX+dis)	DDCBxxCE	: SET 7,(HL)	CBFE	: XOR H	AC
SET 1,(IY+dis)	FDCBxxCE	: SET 7,(IX+dis)	DDCBxxFE	: XOR L	AD
SET 1,A	CBCF	: SET 7,(IY+dis)	FDCBxxFE		
SET 1,B	CBC8	: SET 7,A	CBFF		
SET 1,C	CBC9	: SET 7,B	CBF8		
SET 1,D	CBCA	: SET 7,C	CBF9		
SET 1,E	CBCB	: SET 7,D	CBFA		
SET 1,H	CBCC	: SET 7,E	CBFB		
SET 1,L	CBCD	: SET 7,H	CBFC		
SET 2,(HL)	CBD6	: SET 7,L	CBFD		
SET 2,(IX+dis)	DDCBxxD6	: SLA (HL)	CB26		
SET 2,(IY+dis)	FDCBxxD6	: SLA (IX+dis)	DDCBxx26		
SET 2,A	CBD7	: SLA (IY+dis)	FDCBxx26		
SET 2,B	CBDO	: SLA A	CB27		
SET 2,C	CBD1	: SLA B	CB20		

#### ABBREVIATIONS

n number 0-255  
nn number 0-65535  
xx hex number  
xxxx two hex numbers  
dis displacement

MNEMONIC	HEX	MNEMONIC	HEX	MNEMONIC	HEX
ADC A,(HL)	8E	: BIT 1,(HL)	CB4E	: BIT 6,L	CB75
ADC A,(IX+dis)	DD8Exx	: BIT 1,(IX+dis)	DDCBxx4E	: BIT 7,(HL)	CB7E
ADC A,(IY+dis)	FD8Exx	: BIT 1,(IY+dis)	FDCBxx4E	: BIT 7,(IX+dis)	DDCBxx7E
ADC A,A	8F	: BIT 1,A	CB4F	: BIT 7,(IY+dis)	FDCBxx7E
ADC A,B	88	: BIT 1,B	CB48	: BIT 7,A	CB7F
ADC A,C	89	: BIT 1,C	CB49	: BIT 7,B	CB78
ADC A,D	8A	: BIT 1,D	CB4A	: BIT 7,C	CB79
ADC A,n	CExx	: BIT 1,E	CB4B	: BIT 7,D	CB7A
ADC A,E	8B	: BIT 1,H	CB4C	: BIT 7,E	CB7B
ADC A,H	8C	: BIT 1,L	CB4D	: BIT 7,H	CB7C
ADC A,L	8D	: BIT 2,(HL)	CB 56	: BIT 7,L	CB7D
ADC HL,BC	ED4A	: BIT 2,(IX+dis)	DDCBxx56	: CALL addr	CDxxxx
ADC HL,DE	ED5A	: BIT 2,(IY+dis)	FDCBxx56	: CALL C,addr	DCxxxx
ADC HL,HL	ED6A	: BIT 2,A	CB57	: CALL M,addr	FCxxxx
ADC HL,SP	ED7A	: BIT 2,B	CB50	: CALL NC,addr	D4xxxx
ADD A,(HL)	86	: BIT 2,C	CB51	: CALL NZ,addr	C4xxxx
ADD A,(IX+dis)	DD86xx	: BIT 2,D	CB52	: CALL P,addr	F4xxxx
ADD A,(IY+dis)	FD86xx	: BIT 2,E	CB53	: CALL PE,addr	ECxxxx
ADD A,A	87	: BIT 2,H	CB54	: CALL PO,addr	E4xxxx
ADD A,B	80	: BIT 2,L	CB55	: CALL Z,addr	CCxxxx
ADD A,C	81	: BIT 3,(HL)	CB5E	: CCF	3F
ADD A,D	82	: BIT 3,(IX+dis)	DDCBxx5E	: CP (HL)	BE
ADD A,n	C6xx	: BIT 3,(IY+dis)	FDCBxx5E	: CP (IX+dis)	DD8Exx
ADD A,E	83	: BIT 3,A	CB5F	: CP (IY+dis)	FDBExx
ADD A,H	84	: BIT 3,B	CB58	: CP A	BF
ADD A,L	85	: BIT 3,C	CB59	: CP B	B8
ADD HL,BC	09	: BIT 3,D	CB5A	: CP C	B9
ADD HL,DE	19	: BIT 3,E	CB5B	: CP D	BA
ADD HL,HL	29	: BIT 3,H	CB5C	: CP n	FExx
ADD HL,SP	39	: BIT 3,L	CB5D	: CP E	BB
ADD IX,BC	DD09	: BIT 4,(HL)	CB66	: CP H	BC
ADD IX,DE	DD19	: BIT 4,(IX+dis)	DDCBxx66	: CP L	BD
ADD IX,IX	DD29	: BIT 4,(IY+dis)	FDCBxx66	: CPD	EDA9
ADD IX,SP	DD39	: BIT 4,A	CB67	: CPDR	EDB9
ADD IY,BC	FD09	: BIT 4,B	CB60	: CPI	EDA1
ADD IY,DE	FD19	: BIT 4,C	CB61	: CPIR	EDB1
ADD IY,IY	FD29	: BIT 4,D	CB62	: CPL	2F
ADD IY,SP	FD39	: BIT 4,E	CB63	: DDA	27
AND (HL)	A6	: BIT 4,H	CB64	: DEC (HL)	35
AND (IX+dis)	DDA6xx	: BIT 4,L	CB65	: DEC (IX+dis)	DD35xx
AND (IY+dis)	FDA6xx	: BIT 5,(HL)	CB6E	: DEC (IY+dis)	FD35xx
AND A	A7	: BIT 5,(IX+dis)	DDCBxx6E	: DEC A	3D
AND B	A0	: BIT 5,(IY+dis)	FDCBxx6E	: DEC B	05
AND C	A1	: BIT 5,A	CB6F	: DEC BC	0B
AND D	A2	: BIT 5,B	CB68	: DEC C	0D
AND n	E6xx	: BIT 5,C	CB69	: DEC D	15
AND E	A3	: BIT 5,D	CB6A	: DEC DE	1B
AND H	A4	: BIT 5,E	CB6B	: DEC E	1D
AND L	A5	: BIT 5,H	CB6C	: DEC H	25
BIT 0,(HL)	CB46	: BIT 5,L	CB6D	: DEC HL	2B
BIT 0,(IX+dis)	DDCBxx46	: BIT 6,(HL)	CB76	: DEC IX	DD2B
BIT 0,(IY+dis)	FDCBxx46	: BIT 6,(IX+dis)	DDCBxx76	: DEC IY	FD2B
BIT 0,A	CB47	: BIT 6,(IY+dis)	FDCBxx76	: DEC L	2D
BIT 0,B	CB40	: BIT 6,A	CB77	: DEC SP	3B
BIT 0,C	CB41	: BIT 6,B	CB70	: DI	F3
BIT 0,D	CB42	: BIT 6,C	CB71	: DJNZ,dis	10xx
BIT 0,E	CB43	: BIT 6,D	CB72	: EI	FB
BIT 0,H	CB44	: BIT 6,E	CB73	: EX (SP),HL	E3
BIT 0,L	CB45	: BIT 6,H	CB74	: EX (SP),IX	DDE3



MMEMONIC	HEX	MMEMONIC	HEX	MMEMONIC	HEX
EX (SP),IY	FDE3	: LD (addr),SP	ED73xxxx	: LD C,A	4F
EX AF,AF'	08	: LD (BC),A	02	: LD C,B	48
EX DE,HL	EB	: LD (DE),A	12	: LD C,C	49
EXX	D9	: LD (HL),A	77	: LD C,D	4A
HALT	76	: LD (HL),B	70	: LD C,n	0Exx
IM 0	ED46	: LD (HL),C	71	: LD C,E	4B
IM 1	ED56	: LD (HL),D	72	: LD C,H	4C
IM 2	ED5E	: LD (HL),n	36xx	: LD C,L	4D
IN A,(C)	ED78	: LD (HL),E	73	: LD D,(HL)	56
IN A,port	DBxx	: LD (HL),H	74	: LD D,(IX+dis)	DD56xx
IN B,(C)	ED40	: LD (HL),L	75	: LD D,(IY+dis)	FD56xx
IN C,(C)	ED48	: LD (IX+dis),A	DD77xx	: LD D,A	57
IN D,(C)	ED50	: LD (IX+dis),B	DD70xx	: LD D,B	50
IN E,(C)	ED58	: LD (IX+dis),C	DD71xx	: LD D,C	51
IN H,(C)	ED60	: LD (IX+dis),D	DD72xx	: LD D,D	52
IN L,(C)	ED68	: LD (IX+dis),n	DD36xxxx	: LD D,n	16xx
INC (HL)	34	: LD (IX+dis),E	DD73xx	: LD D,E	53
INC (IX+dis)	DD34xx	: LD (IX+dis),H	DD74xx	: LD D,H	54
INC (IY+dis)	FD34xx	: LD (IX+dis),L	DD75xx	: LD D,L	55
INC A	3C	: LD (IY+dis),A	FD77xx	: LD DE,(addr)	ED5Bxxx
INC B	04	: LD (IY+dis),B	FD70xx	: LD DE,nn	11xxxx
INC BC	03	: LD (IY+dis),C	FD71xx	: LD E,(HL)	5E
INC C	0C	: LD (IY+dis),D	FD72xx	: LD E,(IX+dis)	DD5Exx
INC D	14	: LD (IY+dis),n	FD36xxxx	: LD E,(IY+dis)	FD5Exx
INC DE	13	: LD (IY+dis),E	FD73xx	: LD E,A	5F
INC E	1C	: LD (IY+dis),H	FD74xx	: LD E,B	58
INC H	24	: LD (IY+dis),L	FD75xx	: LD E,C	59
INC HL	23	: LD A,(addr)	3Axxxx	: LD E,D	5A
INC IX	DD23	: LD A,(BC)	0A	: LD E,n	1Exx
INC IY	FD23	: LD A,(DE)	1A	: LD E,E	5B
INC L	2C	: LD A,(HL)	7E	: LD E,H	5C
INC SP	33	: LD A,(IX+dis)	DD7Exx	: LD E,L	5D
IND	EDAA	: LD A,(IY+dis)	FD7Exx	: LD H,(HL)	66
INDR	EDBA	: LD A,A	7F	: LD H,(IX+dis)	DD66xx
INI	EDA2	: LD A,B	78	: LD H,(IY+dis)	FD66xx
INIR	EDB2	: LD A,C	79	: LD H,A	67
JP (HL)	E9	: LD A,D	7A	: LD H,B	60
JP (IX)	DDE9	: LD A,n	3Exx	: LD H,C	61
JP (IY)	FDE9	: LD A,E	7B	: LD H,D	62
JP addr	C3xxxx	: LD A,H	7C	: LD H,n	26xx
JP C,addr	DAxxxx	: LD A,I	ED57	: LD H,E	63
JP H,addr	FAxxxx	: LD A,L	7D	: LD H,H	64
JP NC,addr	D2xxxx	: LD A,R	ED5F	: LD H,L	65
JP NZ,addr	C2xxxx	: LD B,(HL)	46	: LD HL,(addr)	2Axxxx
JP P,addr	F2xxxx	: LD B,(IX+dis)	DD46xx	: LD HL,nn	21xxxx
JP PE,addr	EAxxxx	: LD B,(IY+dis)	FD46xx	: LD I,A	ED47
JP PO,addr	E2xxxx	: LD B,A	47	: LD IX,(addr)	DD2Axxx
JP Z,addr	CAxxxx	: LD B,B	40	: LD IX,nn	DD21xxx
JR C,dis	38xx	: LD B,C	41	: LD IY,(addr)	FD2Axxx
JR dis	18xx	: LD B,D	42	: LD IY,nn	FD21xxx
JR NC,dis	30xx	: LD B,n	06xx	: LD L,A	6F
JR NZ,dis	20xx	: LD B,E	43	: LD L,B	68
JR Z,dis	28xx	: LD B,H	44	: LD L,C	69
LD (addr),A	32xxxx	: LD B,L	45	: LD L,D	6A
LD (addr),BC	ED43	: LD BC,(addr)	ED4Bxxxx	: LD L,n	2Exx
LD (addr),DE	ED53	: LD BC,nn	01xxxx	: LD L,E	6B
LD (addr),HL	22xxxx	: LD C,(HL)	4E	: LD L,(HL)	6E
LD (addr),IX	DD22xxxx	: LD C,(IX+dis)	DD4Exx	: LD L,(IX+dis)	DD6Exx
LD (addr),IY	FD22xxxx	: LD C,(IY+dis)	FD4Exx	: LD L,(IY+dis)	FD6Exx

## TURBO-PASCAL AND ITS USE.

### INTRODUCTION.

Turbopascal is a CP/M compatible programming language. It has the facilities of an editor, compiler, linker and debugger all within one program. This means that any Pascal programs that you wish to write can be written, run and debugged without exiting Turbopascal.

Turbopascal is configured to run the 80 column video card, don't worry if you do not have this as it will run quite happily without it. When running without the 80 column card fitted you are not limited by the 40 column screen as Albert tends to scroll the screen to the left so that you can use the full 80 columns that it supports.

### USING TURBOPASCAL.

To run Turbopascal place the disk containing the program in drive one and boot up as normal. Once the '0:' has appeared type 'TURBO' and this will load up Turbopascal. Once the program has loaded successfully the program will ask whether or not you wish to include error messages. You should answer yes to this question as this determines whether or not the error messages for the compiler are used. If you are like me you need these as non of my programs seem to run the first time. Once this stage has been completed the program will proceed to display the main menu.

### MAIN MENU.

Turbo is a menu driven program, that is to say that at every stage throughout the program you are presented with a list of available options. With the main menu as with any of the other menu's all you need to enter to use an option is its first letter ie, to use the edit facility just enter 'E'. Going back to the use of the program we see that the main menu is made up of the following options.

OPTION	LETTER	FUNCTION.
EDIT	E	THIS ALLOWS YOU TO EDIT THE CURRENT WORK FILE.
COMPILE	C	COMPILES THE CURRENT WORK FILE
RUN	R	RUNS THE CURRENT WORK FILE.
WORKFILE	W	THIS ALLOWS YOU TO CHANGE WORK FILES WITHOUT LEAVING TURBO.
SAVE	S	THIS SAVES THE WORK FILE TO THE SELECTED DISK DRIVE.
DIRECTORY	D	DISPLAYS THE SELECTED DISKS DIRECTORY.
COMPILER OPTIONS	O	COMPILER DESTINATION OPTION.
EXECUTE	X	ALLOWS EXECUTION OF ANOTHER CP/M PROGRAM OTHER THAN TURBO.
QUIT	Q	END TURBOPASCAL.



Now that I have described the main menu to you I will try and describe the use of each option to you as best I can.

#### EDIT.

This option is used to enter into the part of the program that allows you to create or edit your program. If you are creating a fresh file then you will be asked for a file name, this is explained in the workfile command. A full explanation of the edit command is given latter.

#### COMPILE.

This option causes the current work file to be compiled to memory. If there is no current work file then you will be prompted to select one as in the workfile command. It is also possible to compile a program straight to disk, this produces a .COM file when the directory is examined. Compilation to disk is selected by the use of the 'compile options' option.

When ever errors are encountered when compiling the program an error message will be displayed. The system will then wait for you to enter an 'ESC' where upon the cursor will be positioned over the error it has just detected, clever little thing isn't it. A tip if the cursor falls at the start of a line is to check if you have forgotten the semicolon off the end of the last program line like I always do.

#### RUN.

This option allows the work file to be run, if the file has not been compiled then the compiler will take over, compile the program and then run it. So if you are like me and like to see what happens any way whether it works or not, you cant.

You should note that if your program doesn't look at the keyboard at any time and it enters a never ending loop your program will hang up. The only way that it would be possible to exit this is to hit the reset button, this resulting in the loss of your program if you have not saved it. So you should note that it is advisable to save your program before you attempt to run it.

#### WORKFILE.

This option allows you to enter the name of the file that you wish to edit or create. The length of the name is as standard CP/M, 8 letters and a file extension, if a file extension is not specified then a pascal extension is assumed ie, EXAMPLE.PAS. Care should be taken with file extensions, if you do not use them correctly then you may have the result of missing or overwritten files.

It should be noted that for editing, any file extension can be entered as long as the file you wish to edit is an ascii file. The file extension of the file should be quoted when entering the work file name otherwise a new file will be opened.

#### SAVE.

This command allows you to save your current work file to the designated disk. Usually you cannot quit Turbo without first saving the work file.

#### DIRECTORY.

The directory command allows you to examine the designated disk drives directory. On execution of this option you will be asked for a mask. This acts much as in copying and erasing on Albert, the '\*' and the '?' have the same functions. The '\*' shows all files with the appropriate match ie, \*.pas will display all pascal extension files, ?a.com will match any files which are com files and have an 'a' in there name. With turbo there is also another command which is the TYPE command 'T', this allows you to match any file with a name or letters the same as those specified in the mask ie, T\*.com will list all files with a com extension and whose name starts with a T.

#### COMPILE.

This option allows you to decide the destination of the compiled file of data. there are two options here, they are to compile to disk or to compile to memory. There are only two occasions when you would not compile to memory that is when memory is full, and secondly when you have completed and debugged the program and you wish to save it as a compiled .COM file.

#### EXECUTE.

This command allows you to load and run a program in Turbo that has a .COM extension, this would only normally be a program that you have created form Turbo. Note that not all .COM files can be run like this.

#### QUIT.

This command allows you to end TURBOPASCAL, if you have not saved your work file or have not saved it after up dating it you will be prompted to save it before the program quits. If you accidentally exit Turbo and you want to get back in without loosing your program execute a GO 103 command form MOS.

#### EDITOR COMMANDS.

The editor that is used in Turbo is a direct on screen editor, it has search and replace functions as well as most other standard editor functions. Also another handy feature is the automatic indentation, which automatically indents your program to make it more readable and more importantly more understandable. When the editor is first called up the cursor is positioned in the home position at the start of the work file. The editor works in either of two modes, those are the Insert mode and the Overwrite mode. These two modes operate as you would expect with the insert mode allowing you to insert data into the file, and the overwrite mode allowing you to overwrite existing data. Note that when ever the edit mode is called up the editor will be in the insert mode. This can be verified by looking at the status line at the top of the screen, as well as the position of the cursor relevant to the home position at the start of the program.

Cut and paste facilities are also provided by turbopascal, to us mere mortals that is to say block copy and block delete. These facilities are shown in the table that follows.



Automatic indentation is a valuable asset to the language as it allows the correct format to be arranged. The indentation of the program doesn't make any difference to the compiler it is purely for user convenience. I have found that by using this method that debugging of procedures is made much simpler. The effect that automatic indentation has is to place each successive line under each starting at the same place. The way to change the position of the automatic indentation is to use the delete or space key to move the cursor once it has appeared after a enter. I have found that for most purposes that an indentation of 5 spaces is practical.

The following list of commands shows the functions of the editor in edit mode.

FUNCTION	KEY.
Move character forward	Right cursor key.
Move character backwards	Left cursor key.
Move line upwards	Up cursor key.
Move line downwards	Down cursor key.
Move word forward	CTRL-F.
Move word backwards	CTRL-A.
Move screen upwards	CTRL-R.
Move screen downwards	CTRL-C.
Move to start of line	CTRL-Q CTRL-S.
Move to end of line	CTRL-Q CTRL-D.
Move to start of file	CTRL-Q CTRL-R.
Move to end of file	CTRL-Q CTRL-C.
Scroll line upwards	CTRL-W.
Scroll line downwards	CTRL-Z.
Delete character at cursor	CTRL-G.
Delete character before cursor	DEL.
Delete word after cursor	CTRL-T.
Delete current line	CTRL-X.
Insert a line	CTRL-N.
Search for a string	CTRL-Q CTRL-F.
Search and replace	CTRL-Q CTRL-A.
Repeat previous search/find	CTRL-L.
Mark start of block	CTRL-E CTRL-B.
Mark end of block	CTRL-E CTRL-K.
Move block to cursor position	CTRL-E CTRL-V.
Delete block	CTRL-E CTRL-Y.
Copy block to cursor position	CTRL-E CTRL-C.
Change between insert and overwrite mode	CTRL-V.
Finish editing	ESC ESC.

Where two control characters have been given, for example delete block CTRL-E and CTRL-Y, firstly press CTRL and E and then still holding down the CTRL key press the Y key. This is the method for using the editor commands.

HAPPY PROGRAMMING.

P.S.SYMONDS.

\* \* \* \* \* SUPASOFT \* \* \* \* \*

NEW

THE I CHING...Ancient Chinese system of divination brought to the Einstein via disk and 96 page book. If you are ever perplexed, or ever uncertain what to do; if you are faced with a big decision and do not know which way to jump; then this is the program for you. The choice is still yours, but you wont be jumping in the dark.....£ 9.00

\* \* \* \* \*

CASH TRADER...A multi-choice, multi-screen trading game for 2 to 4 players, any, or all, of whom may be the computer. Buy wholesale, sell retail. Trade from shop or stall. Three game lengths. VAT collected every thirteen weeks. Get your prices right, and survive - don't, and you wont.....£ 7.00

EAZIDRAW.....Fully machine coded printed circuit board designer of the highest standard, comes complete with written instructions and on board help screens. Single or double sided boards, load, save and print (Epson compatible, 9 pin) options and a wealth of other features make this a real bargain at only.....£12.00

SUPADISC.....SUPAMATH, SUPAMECH and BANKMAN. 200K of mathematics, mechanics and financial tutorials and utilities all on one disk for only.....£15.00

(for further information and prices of these three SUPADISC programs sold seperately, or in different combinations, please write to the address given below.)

SUPAPLAY.....Still SUPASOFTS biggest seller. Nine games on one disk to suit all ages and abilities. Nothing too complex but great for a rainy afternoon, or ten minutes relaxation.....£ 7.00

\* \* \* \* \*

*Please make all cheques payable to SUPASOFT and include 50p per disk towards postage and packing.*

Orders to, and further details from, Chris. Pickles, 474 Hertford Road, Edmonton, London, N9 8AD.