VOL. 1
NO. 6

STATUS 1500

JULY
1983

* Europe: add 30% * Outside Europe: add 60% *

6 issues £4.50 * 12 issues £8.50 * Single Copies £1.50 * Published Monthly *

* edited by RONALD COHEN, 62 BLENHEIM CRESCENT, LONDON W.11 *

SHAMBLES AT SHARP

Responibility for the PC 1500 has now been transferred from SHARPs Computer Division to their Calculator Division (except for office equipment purposes). So now instead of dealing with executives who know something about computers, but little about PC 1500, we shall be in the hands of those who know little about computers, and nothing about the PC 1500. This downgrading will be offensive to most users .One does not buy a pocket computer entirely to show off: but I doubt whether there are many users who do not get pleasure - when someone says (what a big calculator you have got) - out of replying that it is not a mere calculator, but a full-scale COMPUTER! It is only fair to say that the new team are making considerable effort to push the machine: but since they do not seem to have the vaguest idea of the real way to support it, I cannot see their efforts meeting with more than temporary success.

* * * * *

SHARP announce that their frequently-promised (Programming Manual), which should include machine-code, is to appear in October. No one who has had previous dealings with SHARP will have much confidence in this. Indeed, I have bet SHARPs (Southern Representative) 5 pounds to 5 pence that customers will not see it by the end of that month, and he has cheerfully accepted the bet. He has recently joined SHARP.

* * * * *

Meanwhile STATUS 1500 brings you on page 61 what we believe to be the complete op-codes for our machine. Explanations of what it all means will follow next month.

* * * * *

On page 56 we publish a brilliant and useful subroutine by SIMON COX, and a splendidly chauvinistic graphics program from JOHN MACK. Both programs are warmly recommended.

*** CONTENTS ***

Istanbul,May 31st,1983

Dear Mr.Cohen,

I have just received issue 5 and with much interest I read about your dispute with Mr.TRAYNOR whether or not the SAFE-CRACKER works on his EASI-Programs.

I think you both are right!

Just adding the SAFECRACKER by hand (as you suggest) results in that program getting 'cracked',no matter where you place it. I tried at line 2500 etc and at 60000 etc but in both cases:BLOW! The uncracked EASI-FILE was still in,SAFECRACKER had exploded and parts of programs long since erased partially appeared.

Maybe MERGEing does work,however,I did not try.
Instead with my APPEND facility I have put the SAFECRACKER as a second module behind EASI-FILE and let it run via an adress. Lovely how line numbers were run down until the flaw was detected!
Confidently I may tell Mr.TRAYNOR that EASI-FILE actually ends up with line number 2410.

For me a complete listing is essential since on arrival of 'TOOL2' I intend to substitute the INPUT#/PRINT# commands in lines 2320/2330 and 2410 with the new commands FSAVE/FLOAD in order to get a faster Data loading.

Presently my extended PC gives me 256 files of 80 characters each plus (acc. to STATUS 3 - STATUS 2) annother 2882 Bytes free for other uses. BUT SAVEING AND LOADING TAKES ABOUT 20 MINUTES EACH!

I am very hopeful to get the time down to under one minute with the new commands.Then the real value of EASI-FILE will become evident! I will immediately report on TOOL2 as soon as I have tried it out.

I very much appreciate your efforts to get some hints on PEEK,POKE etc. on to paper.Although I have both the HEXMONITOR and the brand new MACROASSEMBLER (both from Rasso von Schlichtegroll/Munich) at home and even loaded into my modified CE-159 I do not have the faintest idea how to operate them.Both instruction manuals definitively are excellent - in case you have the basic knowledge in mnemonics or whatever in order to understand the language.
Could you suggest an introduction into machine programming for people with an average IQ?

One final question: I very much liked your'letter' of May 14th written ROTATEd on the CE150.Do you have a program printout to spare?

Sincerely yours

Hans-Henning Heine

Boolean logic is not something uffish out of Alice in Wonderland, (although Lewis Carroll, under his real name of C.L.Dodgson, was in fact a noted mathematician and logician). It is, for our purposes, a way of dealing mathematically with logical statements.

If a statement is true, it returns 1. If false, it returns 0. As you may be tired of hearing, this is how the machine 'thinks'. It only recognises, ultimately, 1s and 0s. Try this from the keyboard:

a) A=7 b) (A=7) [DON'T FORGET THE BRACKETS]

and the display will show 1 . If you now enter *c)(A=8)* the display will show 0 . The = sign is dual purpose in the sense that, according to context, it is sometimes an *assignment* statement, and at other times it is a *comparison* instruction.

In a multi-statement line, when the IF condition is not met, the remaining statements in the line are ignored. This can at times be most useful: occasionally it can be a nuisance. Often the remaining statements are required, regardless, and cannot conveniently be transferred to the following line. The use of IF can often be reduced or avoided by the form described above. Compare the two routines below:

```
a) 10: IF A=7 LET AA=47: BEEP 1
   20: IF B=9 LET BB=58: BEEP 1
   30: IF C=4 LET CC=33: BEEP 1
   40: etc., etc.,

b) 10: AA=47*(A=7): BB=58*(B=9): CC=33*(C=4): BEEP (A=7)+(B=9)+(C=4)
```

Again, much time may be saved by a judicious use of line numbers:

```
a) 10: IF A=30 LET A$="ARIES" : GOTO 9999
   20: IF A=60 LET A$="TAURUS": GOTO 9999
   30: IF A=90 LET A$="GEMINI": GOTO 9999
                    and so on

b)  5: GOSUB A: GOTO 9999
   30: A$="ARIES":RETURN
   60: A$="TAURUS":RETURN
   90: A$="GEMINI": RETURN
```

In routine a) up to 12 IF statements must be examined. In routine b) only one statement must be dealt with.

The Boolean form is often useful as substitute for AND and OR. For instance, there is no XOR (exclusive OR) in the BASIC provided. How would you deal with the following, using OR and AND ?

"If A or B is true, but not both!"

The simplest way is by:

```
10: IF (A=1)+(B=1) = 1  ..........
```

You could even have *"If two out of three statements are true then..."* by:

```
10: IF (A=1)+(B=1)+(C=1) = 2 THEN ............
```

Finally:

```
10: IF A$="THIS ARTICLE IS HELPFUL TO YOU" LET Z$="GOOD!"
```

DISTRESS SIGNALS

Several readers have been displeased by some unusual mnemonics in IAN TRAYNOR's *INTRODUCTION TO MACHINE CODE*, and by certain statements therein which purists find not quite accurate.

If one of the complainants had submitted a superior version it would have been printed. This is a monthly newsletter, and while every effort is made to maintain complete accuracy, this can only be in the light of present knowledge and actual submissions: I personally have found IAN TRAYNOR's introduction extremely helpful, even if parts of it must be amended later in the light of further revelations.

JOHN KERR reproves me for using the "^" operator in my 'HEX & DEC' program last month. He points out that this sign makes use of logarithms, and is thus slow and inaccurate. Challenged to do better, he offers the 2 superior versions below.

```
1Ø: INPUT "dec "; D
2Ø: LPRINT D; " &";
3Ø: K$=""
4Ø: L=D-16*INT (D/16-3)
5Ø: D=INT (D/16)
6Ø: K$=CHR$ (L+7*(L>57))+K$
7Ø: IF D GOTO 4Ø
8Ø: LPRINT K$
9Ø: CLEAR: GOTO 1Ø
```

```
1Ø: INPUT "dec "; D: X=D
2Ø: LPRINT D; " &";
3Ø: ON 1+(D>255) GOSUB 7Ø,6Ø
4Ø: LPRINT
5Ø: CLEAR: GOTO 1Ø
6Ø: X=INT (D/256): GOSUB 7Ø: X=D-256*X
7Ø: Y=X/16: GOSUB 8Ø: Y=15 AND X
8Ø: LPRINT CHR$(Y+48+7*(Y>9));
9Ø: RETURN
```

INT(X/16)

While I will not admit the superiority in the present case, it is certainly true that logs are responsible for all those irritating cases where 'the number you first thought of' has .00001 tagged on the end. It is most interesting to examine the very different method of approaching the problem, and repays study; there is no one correct way of writing a program or solving a problem.

A.W.SLADE, in Abu Dhabi, has very kindly solved the CE 153 problem posed last month by R.J.COURT of New Zealand.

Thank you!

PETE ELDRIDGE is writing a 'password' program, and would like to know the ASCII code for the ON key.

Experimenting with

```
10: ARUN: A$=INKEY$:LPRINT ASC A$; Z
20: Z=999: END
```

[RUN/ Turn OFF/ Turn ON] leads me to believe that the ON key returns the ASCII code 0 . Strictly speaking, this is the NULL code. You may find the routine on page 55 useful. It might also be interesting to experiment with a Time Lock of some sort.

H-H HEINE is interested in buying a printer, and connecting to PC 1500 via CE 158 interface, without the intervention of another computer. He would like to be sure that the set-up will work before spending vast sums. He does not say whether he wants 'daisywheel' or 'dotmatrix'.

If any reader has successfully made such a connection, full details would be most welcome. Please help!

COLIN STONE wonders whether the RENUMBER program on page 17 could not be adapted to automatically renumber all GOTO and GOSUB statements.

It is a question of space and time. It would - it seems to me - be necessary to store all the new and old line numbers and re-match them. However perhaps a machine-code program stored in the String-Variable locations could do it? Any offers?

PLEASE WILL ALL THOSE KIND READERS WHO SEND ME PROGRAMS AND SUBROUTINES PUT THE TITLE AND THEIR NAME ON THE BACK OF EACH ONE! I HAVE A MASS OF VERY INTERESTING LITTLE BITS OF PAPER FLOATING AROUND, AND ONLY WISH I KNEW WHAT THEY WERE!

PEEK, POKE & MEMORY - VI

I have been surprised to find that several readers have had great difficulty with this series: though they have not responded to my invitation to inform me what the specific difficulties are. Perhaps it is just fear of what appears to be a difficult subject. But it really is not as hard as it appears, *provided you work through the articles with concentration*. No instructions ever make much sense without the machine beside you to try the instructions on.

Perhaps this - the sixth - article in the series is the right place for a brief recap of what we have dealt with so far.

The computer's memory consists of a series of 'addresses', like electronic pigeonholes, from 0 to 65535 (256*256 -1). When a program is executed, information is retrieved from some of these in turn, or as specified. The potential program area runs from 0 to 28K, but some of the space is blank, depending on your memory. See memory map on page 38 for details. All this information is held in the form of numbers from 0 to 255, whether line numbers, ASCII codes, or whatever. POKE allows you to change these numbers: and PEEK to inspect them. From address 28672 to 32767 lie various pointers and counters. By POKEing numbers into some of these you can directly change the execution of a program, and in ways that the strict BASIC does not permit. The most important are the main system pointers - 30821 to 30826 - and these can be used to alter the start and finish addresses of programs, and to restore programs which have been deleted or MERGEd: but usually only in conjunction with other code within the program. By POKEing you can prime your RESERVE keys, starting at (STATUS 2 - STATUS 1 - 111), and your RESERVEtemplate area, starting at (STATUS 2 - STATUS 1 - 189). Note that to get a true value from STATUS 2 - STATUS 1 you *must* have at least one line of program in the machine. Finally it is important to remember that POKE is a verb, the form is POKE nnnnn,n : whereas PEEK nnnnn is a noun which can be added or multiplied, just like STATUS n, or X, or any other number or number value.

Please do not be daunted! It is all most useful, and not as hard as it looks at first sight.

* * * * *

Protection and counter-protection form a never-ending game of 'cops and robbers'. There is little one can do to deter really determined and intelligent and patient pirates. But the following will prevent casual prying hands from tinkering with your PC 1500 , as long as you make sure that you turn it OFF in RUN mode .

```
  1: ARUN: LOCK: POKE# 61453, 128
  2: ON ERROR GOTO 100
  3: A$ = INKEY$: GOTO ASC A$
 15: POKE# 61453, 0: UNLOCK: END
100: CALL 52593
```

The effect of this is that you can only turn the computer ON if during the preliminary linefeed you keep your finger on the OFF key ! (You can alter any line numbers in the above, *except* 15).

PEEK, POKE & MEMORY may not appear next month, in which case the series will be resumed in the September issue. If you have any queries do please write in: I cannot help you if you remain silent.

TOOLKIT "INSTANT RESCUE" by SIMON COX

This is a machinecode subroutine, BUT YOU DO NOT NEED TO UNDERSTAND MACHINE-CODE IN ORDER TO OPERATE IT. JUST FOLLOW THE INSTRUCTIONS. It will enable you to recover a deleted program at the touch of a single key!

It occupies part of the RESERVE template area.

FROM THE KEYBOARD:

a) T = STATUS 2 - STATUS 1 - 189 *[ENTER]*

b) POKE T, 72, PEEK 30821, 74, 197, 68, 181, 0, 1, 153, 6 *[ENTER]*

c) POKE T + 10, 174, PEEK 30821, 197, 132, 174, 120, 103, 4, 174 *[ENTER]*

d) POKE T + 19, 120, 104, 154, 0 *[ENTER]*

e) *Program a RESERVE key as* CALL 14344@
(This figure is for 8K extension: add 2048 to it for unexpanded machine: or subtract 6K from it for 8Kexpansion with battery)

NOTE: you must have at least one line of program in memory for line a) to give a true value.

Now when you have deleted a program by writing NEW, it can immediately be fully restored, without restriction*, merely by pressing your chosen RESERVE key. Your Reserve template area is no longer available for templates. The facility remains unaffected by changing programs, but is deleted by writing NEW in RESERVE mode.

**except that if your first line number is more than 255 it may be changed: and if your first line number is an exact multiple of 256 difficulties could occur in running this line.*

U.J. by JOHN MACK

[Initiate execution by keying: RUN 10]

```
 1:LINE (97,Y)-(1
   19,Y):RETURN
 2:A$=A$+CHR$ (A+
   52+Y):RETURN
10:LF 9:LF -9:
   GRAPH :A=5/3:B
   =47:C=91:D=125
   :FOR Y=0TO B:
   IF Y<36LINE (0
   ,Y)-(A*(35-Y),
   Y),0,1
20:LINE (C-A*Y,Y)
   -(C,Y):IF Y>4
   LINE (D,Y)-(D+
   A*(Y-4),Y)
30:E=216:IF Y<39
   LINE (E-A*(39-
   Y),Y)-(E,Y)
50:NEXT Y:
   GLCURSOR (0,8)
   :SORGN :LINE -
   (E,0),,3:FOR Y
   =-1TO -8STEP -
   1:IF Y<-8LINE
   (A*(-Y-8),Y)-(
   A*-Y,Y)
60:IF Y>-9LINE (0
   ,Y)-(A*-Y,Y)
70:GOSUB 1:LINE (
   D+A*(B+Y),Y)-(
   E+A*Y,Y):NEXT
   Y:FOR Y=-48TO
   -52STEP -1:
   GOSUB 1:NEXT Y
90:FOR Y=-53TO -7
   6STEP -1:LINE
   (0,Y)-(E,Y):
   NEXT Y:FOR Y=-
   77TO -81STEP -
   1:GOSUB 1:NEXT
   Y
110:GLCURSOR (0,-8
   2):SORGN :FOR
   Y=0TO -46STEP
   -1:F=A*(B+Y):
   LINE (F,Y)-(C+
   A*Y,Y):GOSUB 1
120:IF Y>-40LINE (
   E-F,Y)-(E-A*(3
   9+Y),Y)
130:IF Y<-39LINE (
   E-F,Y)-(E,Y)
150:NEXT Y:
   GLCURSOR (0,-8
   ):SORGN :LINE
   -(E,0):LINE -(
   E,129):LINE (0
   ,129)-(0,0)
170:FOR Y=0TO B:IF
   Y>7LINE (0,Y)-
   (A*(Y-8),Y)
180:IF Y<43LINE (C
   -A*(43-Y),Y)-(
   C,Y),,1
190:LINE (D,Y)-(D+
   A*(B-Y),Y):IF
   Y>12LINE (E-A*
   (Y-12),Y)-(E,Y
   )
210:NEXT Y:DIM A(2
   ):A(0)=4.42419
   2017E22:A(1)=1
   .4162626E20:A(
   2)=2.5424223E2
   3:A$=""
220:FOR Y=0TO 2:A=
   INT LOG A(Y):
   GOSUB 2:A(Y)=A
   (Y)*10^-(A-1):
   NEXT Y
230:FOR X=0TO 4:
   FOR Y=0TO 2:A=
   INT A(Y):GOSUB
   2:A(Y)=100*(A(
   Y)-A):NEXT Y:
   NEXT X:
   GLCURSOR (12,-
   20):COLOR 0:
   LPRINT A$
```

It is not easy to write a review of this Statistical Forecasting Program, for the simple reason that I can find little about it that does not meet with my unqualified approbation. I am seldom so enthusiastic, but I have no doubt that this is the most useful and the best-produced piece of software that has been developed for the PC 1500 until now.

The manual is a model of clarity and information. The explanation, of what a statistical curve is, simplifies a subject which professional statisticians usually take pains to obscure. The program will in fact, when you have entered your data, *select* the type of curve which gives the best fit, and gives the measure of correlation: the other types are available for comparison. The selection is made from no less than *eight* types of linear, or linear transformation, curves, - and polynomials are available as well! Only 4K is required; printer optional.

Not content with providing all the information you could require, EASITREND will display it graphically. The author has for good reasons decided against providing a background, as on a conventional graph, so the the graphics are an illustration, rather than a graph from which measurements can be taken. Obviously the smallness of the scale, the thickness of the lines, and the well-known 'graphic slip' could make this misleading. Nevertheless I regret the absence of some sort of a grid, even if only on aesthetic grounds.

It is possible both to project a curve; and to find the corresponding value for any value you choose within range. You may if you wish insist that the program gives information for a particular type of curve: you are not bound by 'best fit'. Editing facilities are excellent, and most types of error in entering material can be corrected without having to start over. The range of secondary and tertiary menus is a little long-winded for my own tastes: but readers will have learnt to discount this particular fad of mine. In fact in order to find some adverse comment, I am reduced to pointing out a totally unimportant spelling mistake on page 23 of the manual.

I have of course discarded my own confused statistics program: with EASITREND the projection is merely a readership of 4018.73659 by August 1985. If 0.73659 of a reader will present himself at the Editorial Offices, he will be offered a proportional rebate on his subscription.

EASITREND by Ian Traynor, is distributed by ELKAN ELECTRONICS, £19.95

ON ERROR

ON ERROR GOTO is a very useful instruction. It can allow you to have more READ statements than DATA: or to approach a section of program *either* directly *or* as a subroutine. The ON ERROR GOTO will take care, in the first instance, of the superfluous RETURN. But later in the program, this ON ERROR GOTO will direct the pointer wrongly: indeed while developing the program you may want your ERRORs to be displayed. The book does not tell you how to cancel the instruction. It can be done simply by inserting a line:

```
ON ERROR GOTO Ø
```

"SUPERTEXT! "

Software review
by C.P.UNDERWOOD

This is a good little program, and I find it useful. Really I should start off by admitting that this review is not unbiased, since I was concerned with its development, and have an interest in its success. All the same, it does have quite a few good features. You enter a word or phrase or even a whole sentence at a time. I could write with it as fast as I can type - about 20 to 25 w.p.m - but printing out was slow. The editing facilities are superb, and easy to handle. You can set the line length, and need not worry about going over it, since the excess is automatically transferred to the start of the new line, *without splitting words*, and superfluous spaces are also taken care of. You can center titles, or move a name and address to the righthand edge. For the sake of appearances, you can "justify" the whole text, which means that the right margin is straight, as in printing,not like normal typing. I do not know that this helps as much as it might, since it is done by adding extra spaces, whereas in printing all the characters are minutely spaced to give the effect. I did not find it hard to remember , after a bit of practice, all the various commands, or at any rate the most commonly used ones. Not being menu-driven, you can go from any command to any other at any time, without being interrogated. Once or twice I pressed a wrong sequence of keys, and got ERROR: but no harm was done, since DEF F restarted the program where I had got to, without loss. The small characters are not attractive, nor are they all properly spaced; this is not the fault of the author, who has plans to redesign the whole lower-case character set. There is also a 'miniprintout'in CSIZE 1, which is quicker than the normal in CSIZE 2, but not so easy to read.It seems all pleasantly flexible, and you can with care add special commands of your own , if you have enough space. Some of the coding is very ingenious, but there are a few duplications and waste of space. The author claims that this is to allow further development; but the program shows clearly that it has been developed piecemeal rather than planned as a whole. However it works quite nicely, and one could write and print out about a thousand words, which should be enough: the program is just over 4 K in length. A useful addition would be provision to use it for "form" letters, with a series of names as DATA. It would not be hard to add this oneself. The price includes a listing, and future improvements. The latter could perhaps get too complicated to be of much use: the program is , by and large, alright as it is, and is reasonably good value for writing memos and so on.

SUPERTEXT! by Ronald Cohen is distributed by STATUS SOFTWARE, £7.95 to subscribers, and by ELKAN ELECTRONICS at £19.95 to non-subscribers.

MINDBOGGLE CORNER

It has been pointed out to me that this series is excessively orientated towards graphics: so here is one for the aristocracy.i.e. those superhuman beings who program in machinecode. A CONCISE SUBROUTINE FOR A "TOOLKIT SUPERMERGE" , on the lines of the TOOLKIT INSTANT RESCUE by Simon Cox on page 56 . It should tuck away neatly in the Reserve Template Area. *Useful prize, closing date August 15th .*

Only 2 correct solutions received so far for the JUNE Mindboggler, and not a single entry for the MAY competition. There is still time left until July 15th, and August 1st, respectively.

THE DEMONSTRATION PROGRAM (on page 48) EXPLAINED by IAN F. TRAYNOR

all numbers are HEX, except those in italics, thus *[31664]*

Address	Op-codes	Mnemonic	Explanation
[30912]	48 7B	LDI XH, 7B	Set X-register to start of display buffer: 7BB0 *[31664]*
[30914]	4A B0	LDI XL, B0	
[30916]	6A 20	LDI UL, 20	Set U-lower register to lowest ASCII code to be displayed i.e. *[32]* which is 'space'
[30918]	68 1A	LDI UH, 1A	Set U-higher register to number of characters to be displayed i.e. *[26]*
[30920]	24	LDA UL	Load UL into Accumulator A
[30921]	41	SIN X	Store Accumulator in X-register, then increase X-register by 1
[30922]	60	INC UL	Increase UL register by 1
[30923]	FD 62	DEC UH	Decrease UH register by 1
[30925] *[30926]*	99 07	BNZ 07	Branch back 7 bytes to *[30920]* if the result of the last operation (DEC UH) does not= *0*.
[30927]	BE E8 CA	JSR E8CA	... otherwise jump to subroutine at address E8CA *[59594]*. This is a ROM subroutineto display contents of display buffer on LCD
[30930]	9A	RTS	End of machinecode subroutine: return to BASIC at address from which it was called.

NOTES: a) *BNZ is properly known as BZS : for the 6502 chip it is BNE*

b) *JSR is properly known as SJP*

c) *RTS is properly known as RTN*

"WEIRD NOISES"

Variations on a Theme by MALCOLM RAY

```
1:REM ABCDEFGHIJ
  KLMNOPQR
2:S=STATUS 2-
  STATUS 1+4
3:DATA &5A,&02,&
  6A,&04,&48,&01
  ,&4A,&02,&BE,&
  E6,&6F,&60,&99
  ,&06,&52,&99,&
  0F,&9A
5:FOR I=1TO 18:
  READ A:POKE S+
  I,A:NEXT I
7:CALL S+1
9:POKE S+2,RND 2
  :POKE S+4,RND
  8
10:POKE S+8,RND 1
  6
11:GOTO 7
```

Varying the 6th byte, &01, also produces interesting sounds.

MACHINE CODE - more background

More headaches! but the following notes may help to make the concept of programming in machinecode slightly less obscure. They will not actually teach you how to program, if you have not met machinecode before: to do so is beyond the scope of this magazine.

***

The Accumulator A is, so to speak, the "current account". It is the only register that can be flexibly manipulated . This is the reason for much of the tedious shuffling of registers in and out of the Accumulator.

***

The mnemonics, like LDA X, i are a language of which there are many dialects, one for each chip, and sub-dialects for each machine. If, as in our case, the machine is the sole user of the particular chip, and the manufacturers obdurately refuse to release the machinecode, then everyone tends to invent their own, which causes confusion. This newsletter will attempt to set up a standard system, mainly but not entirely derived from TANDY. If SHARP ever publish the long-awaited "programming manual", and if it is comprehensible, then the standard may have to be changed.

For instance, take the Op-code 99 . Ian Traynor gives the mnemonic for this as BNZ [Branch if Not Zero]. This is more commonly known as BNE [Branch if Not Equal to zero]. In the table we publish TANDY give BZS as meaning "Branch if Z=1" and BZR as meaning "Branch if Z=0", and it is the latter that has the op-code 99. Z is a flag, occupying one bit of the Status Register S , which (I presume) shows 1 if the last operation resulted in 0 , and 0 if it resulted in anothervalue.

***

With R standing for "a register" it is important to note the difference between an expression like XYZ R and XYZ (R). The former means: "do something by with or from the register R". The latter, with the name of the register in brackets, means "do something with the contents of a memory location whose address is to be found in register R".

***

You cannot write machine-code directly into the program like BASIC. The computer is trained to read program lines as BASIC, and unless you have installed a "machine-code monitor" it will not treat your code as machine code. What you have to do is to find a chunk of memory where you will not interfere with normal running, and into that chunk you POKE your machinecode op-codes: to execute you CALL the address with which that chunk of memory starts.

There are 3 convenient ways of doing this. First, find the address of certain fixed variables of your choice, and use those locations, as Ian Traynor has done (being careful not to use or mention those variables in your BASIC program, or you will destroy the op-coding). Second , start your BASIC program with 1: REM ABCD.....XYZ and so on: and then in BASIC POKE your op-codes into the addresses following the REM (since the rest of a line following a REM is exempt from execution in BASIC). Make sure that you have enough letters etc following the REM for your op-codes to replace. You are of course limited to 80 bytes. Third , if you can spare your RESERVE program area , or better still your RESERVE template area, POKE op-codes direct into these addresses. Always end by 0 as the last POKE, even after the code "Return to Basic program" (&9A).

***

The chart on page 61 will not mean much without the full definitions of the mnemonics: these will follow in the AUGUST issue.

High Byte

Low Byte	0	1	2	3	4	5	6	7
0	SBC XL	SBC YL	SBC UL		INC XL *INC XH*	INC YL *INC YH*	INC UL *INC UH*	
1	SBC(X) *SBC#(X)*	SBC(Y) *SBC#(Y)*	SBC(U) *SBC#(U)*		SIN X	SIN Y	SIN U	
2	ADC XL	ADC YL	ADC UL		DEC XL *DEC XH*	DEC YL *DEC YH*	DEC UL *DEC UH*	
3	ADC(X) *ADC#(X)*	ADC(Y) *ADC#(Y)*	ADC(U) *ADC#(U)*		SDE X	SDE Y	SDE U	
4	LDA XL	LDA YL	LDA UL		INC X	INC Y	INC U	
5	LDA(X) *LDA#(X)*	LDA(Y) *LDA#(Y)*	LDA(U) *LDA#(U)*		LIN X	LIN Y	LIN U	
6	CPA XL	CPA YL	CPA UL		DEC X	DEC Y	DEC U	
7	CPA(X) *CPA#(X)*	CPA(Y) *CPA#(Y)*	CPA(U) *CPA#(U)*		LDE X	LDE Y	LDE U	
8	STA XH *LDX X*	STA YH *LDX Y*	STA UH *LDX U*	NOP	LDI XH,i *LDX S*	LDI YH,i *LDX P*	LDI UH,i	
9	AND(X) *AND#(X)*	AND(Y) *AND#(Y)*	AND(U) *AND#(U)*		ANI(X)i *ANI#(X)i*	ANI (Y)i *ANI#(Y)i*	ANI(U)i *ANI#(U)i*	
A	STA XL *POP X*	STA YL *POP Y*	STA UL *POP U*		LDI XL,i *STX X*	LDI YL,i *STX Y*	LDI UL,i *STX U*	
B	ORA(X) *ORA#(X)*	ORA(Y) *ORA#(Y)*	ORA(U) *ORA#(U)*		ORI(X)i *ORI#(X)i*	ORI(Y)i *ORI#(Y)i*	ORI(U)i *ORI#(U)i*	
C	DCS(X) *DCS#(X)*	DCS(Y) *DCS#(Y)*	DCS(U) *DCS#(U)*		CPI XH,i *OFF*	CPI YH,i	CPI UH,i	
D	EOR(X) *EOR#(X)*	EOR(Y) *EOR#(Y)*	EOR(U) *EOR#(U)*		BII(X)i *BII#(X)i*	BII(Y)i *BII#(Y)i*	BII(U)i *BII#(U)i*	
E	STA(X) *STA#(X)*	STA(Y) *STA#(Y)*	STA(U) *STA#(U)*		CPI XL,i *STX S*	CPI YL,i *STX P*	CPI UL,i	
F	BIT(X) *BIT#(X)*	BIT(Y) *BIT#(Y)*	BIT(U) *BIT#(U)*		ADI(X)i *ADI#(X)i*	ADI(Y)i *ADI#(Y)i*	ADI(U)i *ADI#(U)i*	

NOTE: XYZ R means do something to with or from register R
XYZ(R) means operate on the contents of a location whose *address* is held in register R.
In each box, the entry in italics means that the same code is preceded by FD . i.e 42 is the code for DEC XL, and FD 42 is the code for DEC XH.
i means "immediate value".

8	9	A	B	C	D	E	F
SBC XH	SBC YH	SBC UH		VEJ(C0) *RDP*	VEJ(D0)	VEJ(E0)	VEJ(F0)
BCR,i(+) *SIE*	BCR,i(-)	SBC(ab) *SBC#(ab)*	SBI A,i *HLT*	VCR,i *SDP*	ROR	SPU	AEX
ADC XH	ADC YH	ADC UH		VEJ(C2)	VEJ(D2)	VEJ(E2)	VEJ(F2)
BCS,i(+)	BCI,i(-)	ADC(ab) *ADC#(ab)*	ADI A,i	VCS,i	DRR(X) *DRR#(X)*	RPU	
LDA XH	LDA YH	LDA UH		VEJ(C4)	VEJ(D4)	VEJ(E4)	VEJ(F4)
BHR,i(+)	BHR,i(-)	LDA(ab) *LDA#(ab)*	LDI A,i	VHR,i	SHR		TIN
CPA XH	CPA YH	CPA UH		VEJ(C6)	VEJ(D6)	VEJ(E6)	VEJ(F6)
BHS,i(+)	BHS,i(-)	CPA(ab) *CPA#(ab)*	CPI A,i	VHS,i	DRL(X) *DRL#(X)*		CIN
LOP UL,i *PSH X*	*PSH Y*	SPV *PSH U*	RPV	VEJ(C8) *PSH A*	VEJ(D8)	VEJ(E8)	
BZR,i(+)	BZR,i(-)	AND(ab) *AND#(ab)*	ANI A,i	VZR,i	SHL	ANI(ab)i *ANI#(ab)i*	REC
RTI *POP A*	RTN	LDI S,ij *TTA*	JMP,ij *ITA*	VEJ(CA) *ADR X*	VEJ(DA) *ADR Y*	VEJ(EA) *ADR U*	
BZS,i(+)	BZS,i(-)	ORA(ab) *ORA#(ab)*	ORA A,i	VZS,i	ROL	ORI(ab)i *ORI#(ab)i*	SEC
DCA(X) *DCA#(X)*	DCA(Y) *DCA#(Y)*	DCA(U) *DCA#(U)*		VEJ(CC) *ATP*	VEJ(DC)	VEJ(EC) *ATT*	
BVR,i(+)	BVR,i(-)	EOR(ab) *EOR#(ab)*	EAI,i *XOR*	VMJ,i	INC A	BII(ab)i *BII#(ab)i*	FD
BCH,i(+) *CDV*	BCH,i(-)	STA(ab) *STA#(ab)*	SJP,ij *RIE*	VEJ(CE) *AM0*	VEJ(DE) *AM1*	VEJ(EE)	
BVS,i(+)	BVS,i(-)	BIT(ab) *BIT#(ab)*	BII A,i	VVS,i	DEC A	ADI(ab)i *ADI#(ab)i*	

NOTE: (ab) indicates an actual address. ij means 2 immediate values

(+) means "branch forward, and (-) means "branch back".

FULL EXPLANATIONS NEXT MONTH