FORMAT

FOR SPECTRUM AND SAM USERS

(Incorporating Sinclair & Sam Computing)

-JON-

Are You A Friendly Programmer?

CONTENTS

Vol.6 № 2.   October 1992.

SECRETARY REVELATION

First the good news - as they say - Revelation Software is now up and running again. The software division of Samco has been purchased from the liquidator and has started trading from a new address in Exeter (see their advert in this issue).

A program of re-releasing the existing (Ex SAMCO) titles is well underway and lots of new titles are already in the pipe-line - including games and serious software - for both SAM and Spectrum.

Their first new release is the long awaited 'The Secretary' word-processor for SAM. Released at the end of September the program retails for £14.95 but INDUG members can purchase it for just £12.95 including UK postage.

ATARI LOSSES

Atari are in a defensive mood following the announcement of a second quarter loss of $39.8 million based on a turnover of only $23 million. Last years turnover for the same period was over $49 million.

Atari blames the loss on the cost of internal restructuring and the write-down of inventory.

Their European division still shows a profit, although this is down 80% on last years figure.

In an effort to avoid losing further ST sales to the Commodore Amiga, Atari have started cutting prices of their 520STE and 1024STE bundles in the UK.

ZERO ZEROED

Multi-format games magazine ZERO has closed down. Publishers - Dennis Publishing - said that the market had changed since ZERO was first launched and, despite a recent relaunch, sales had been falling.

ZERO had twice been voted 'Magazine Of The Year' by the industry but falling sales had led to falling advertising revenue.

Many of ZERO's staff have been transferred to a new magazine due for launch in November.

PHILIPS DROP MONITOR PRICES

The highly successful CM8833/II colour monitor from Philips has been cut in price to £199, that's a £30 cut of the old list price. As adverts could already be found offering the monitor for under £200 they should soon appear even cheaper.

Philips say the price drop is to encourage home computer users away from a shared TV and onto a dedicated monitor. The CM8833 has already proved very popular with SAM owners because it comes with scart socket as standard.

STAR COLOUR

Star has announced the launch of a new entry level colour printer called the LC-100 Colour. The nine pin printer is priced at £219 and includes features like paper parking, electronic dip switches and eight resident fonts.

Also launched at the same time is a low cost 24 pin printer called the LC24-100 which will sell for £239. This mono printer comes complete with 16K buffer and 10 letter quality fonts.

Supplies of both printers should be in the shops within the next few weeks.

COMPUTING IN THE SKY

SKY Television, the satellite broadcasting company, have leaked plans for a daily (well Monday to Friday) computer program aimed at the teenage market. While the exact format of the show is still not fixed it will most certainly concentrate on games of course, but it is hoped it will also spot-light education and the serious use of home computers.

Meanwhile Channel 4's GamesMaster program returns for it's second series on October 1st. This is the start of an amazing 26 week run which means that if they instantly repeat the series, like they did the last, it will never be off our screens. The first series was very well received despite the shows presenter being a bit weird to say the least.

More over-the-top acting from Patrick Moore and more over-rating of games consols are in store, but is is worth a watch once in a while.

SPECTRUM EMULATOR FOR PC

A shareware Spectrum emulator for the PC (286 and above) is now available from B.G.Services. The program is available on disc at £3.50 with full registration costing an extra £10.

Registering the program gets you extra utility programs including some for DISCiPLE/ PLUS D disc reading and converting snapshot files. You also receive instructions for building a tape interface which plugs into the centronics port on the PC.

BG Services can now also supply most Spectrum spares and are selling a range of Star printers. For prices ring them on 081 397 0763.

ALL FORMATS SHOW.

A new agreement between several companies, including Format; SD Software and Fred Publishing, should lead to a SAM & Spectrum presence at most All Formats Show venues from now on. The only exceptions from the present list (see advert in this issue) are Donington; Washington; and the Friday 6th November show in Birmingham.

Ring us a few days before a show and we will confirm who will be there.

News Credits: Ken Elston, Barry Davis.

The Editor Speaks

The comments we have received on last months FORMAT have been very good, I'm glad you liked our trips down memory lane.

Now I know many regular colums have been missing in the last few months, mostly due to people going off on holiday and not making sure they were far enough ahead so I wouldn't run out of copy to print. Still, summer is over (at least it looks like it outside) so things should return to normal quite quickly now.

As I write there is still no final word on the SAM takeover by West Coast Computers, the matter is still with the receivers and they don't seem to understand the urgency involved. In the meantime it is nice to see Revelation Software up-and- running, we wish them all the best.

Over the last few months we have received a number of complaints about one of our advertisers - PBT Electronics. While the number has been quite small considering the size of PBT's business, it has nevertheless caused us some concern. Since the end of July PBT's Port Talbot number has been disconnected - according to PBT because they were moving premises - but there seems to be no sign of any change. We have therefore taken the precautionary step of suspending PBT's adverts until the situation is clarified. I sincerely hope that PBT will soon sort out their problems as over the years they have given good service to a lot of readers.

Finally this month, a warm welcome to the readers of Sinclair & Sam Computing for whom this is their first issue. I'm sure you will enjoy the wide mix of articles FORMAT has to offer and I also hope you will benefit from your membership of INDUG.

Bob Brenchley. Editor.

SHORT·SPOT

Edited By:- John Wase.

As I sit here in my little string vest, typing away, I marvel at all our readers, beavering away to send such a varied selection of contributions, month by month. For instance, here's a letter from Chris Ingram from Limbe, Malawi (always a particular pleasure to hear from far-flung places), who writes some very nice things about FORMAT. He then goes on to give some useful bits and additions to some of our earlier items. For instance, the piece from Mr O'Connell in August's FORMAT for turning a Basic program into a SCREEN$ only works if the Basic program is very short. Chris pondered on the origin of the code length; 8500 bytes. Curious. Experiments showed that long programs ended up short of lines. Chris eventually deduced that one really needs 7371 bytes plus the length of the Basic program to cover the area of memory between the screen display and the start of Basic (not allowing for Microdrive maps). Chris took the length of the Basic program from the DISCiPLE catalogue as disc sectors multiplied by 512. He writes that this seems to work well, and it doesn't seem to matter if there is a little bit of spare code at the end.

Chris also mentions the next article which instructed one on how to load a silly message instead of the program name. He tells us that s$ must not be more than five characters, or the name will overrun the ten characters provided for the original name.

Finally, he says that some months, he despairs, after turning over page upon page of SAM articles which could, with very little adaptation, be made to work in Beta basic on a Spectrum. Chris, I am just not able. Putting "Short Spot" together won't pay the bills, and I therefore have a full-time professional job which is demanding in time. I estimate that it takes about 18 - 24 hours a month to get Short Spot together, and I really haven't time to work out conversions of other programs - you, yourself, must know how difficult that is. On the other hand, it ought to be possible to put together a complete table of SAM MasterBasic, SAMBasic, BetaBasic 4.0, BetaBasic 3.0 and SpectrumBasic commands, so that one could see if there was an equivalent for a command merely by looking it up (all right; it's not quite as simple as that, but you know what I mean). If anyone is able to tackle this monumental task, would they please contact me through these columns.

Let's stay with the Spectrum (well, more or less) for a while. Our helpmate, Alan Cox of St. Clears, Dyfed, has sent me a lovely column full of snippets. Poor Alan has temporarily lost his DISCiPLE interface, and therefore had to send me a tape. Cheer up, Alan, it reads fine. The first program has a lot to it - it's quite long. It takes a Spectrum SCREEN$ and deforms it to lie on the surface of a sphere. At the moment, I'm 150 miles from home, typing this on a PC laptop (hiss, hiss) in the intervals between lectures at a symposium, for Bob's deadline is upon me, so a screen dump is a bit difficult, and you'll just have to imagine it. The program was originally written by Geoff Wearmouth.

```
1 REM ************************
2 REM *       DEFORM         *
3 REM *    G. Wearmouth.     *
4 REM ************************
5 REM Popular Computing Weekly
6 REM 30 October 1986
10 IF PEEK 26618<>201 THEN CLEAR 264
   75: PRINT "Installation": GOSUB 9
   000
20 PRINT #0;AT 1,0; PAPER 4;"Load Sa
   ve Print Border Circle Go"
```

7

```
  30 LET a=CODE INKEY$: IF a>97 THEN L
     ET a=a-32
  40 GOSUB 100+a: GOTO 20
 165 RETURN
 166 PLOT 0,0: DRAW 255,0: DRAW 0,175:
     DRAW -255,0: DRAW 0,-175: RETURN
 167 CIRCLE 127.5,87.5,PEEK 26489
 170 RETURN
 171 INPUT "Radius (10-88) ";r: POKE 2
     6489,r: CLS : PRINT #0;AT 0,10;"r
     adius - ";r: RANDOMIZE USR 26488
 175 RETURN
 176 INPUT "Title ";t$: PRINT #0;" Sta
     rt tape": LOAD t$SCREEN$ : RANDOM
     IZE USR 26476
 179 RETURN
 180 REM copy
 182 RETURN
 183 INPUT "Title ";t$: SAVE t$SCREEN$
 999 RETURN
9000 REM ** Machine Code **
9001 REM
9010 LET a=26476: LET s=0
9020 FOR i=1 TO 18: READ t$
9030 FOR j=1 TO 8
9040 LET b=16*(CODE t$-48-(7 AND t$>="
     A"))
9050 LET b=b+(CODE t$(2)-48-(7 AND t$(
     2)>="A"))
9060 POKE a,b: LET a=a+1: LET s=s+b
9070 LET t$=t$(3 TO )
9080 NEXT j
9090 NEXT i
9100 IF s<>13660 THEN PRINT "Data erro
     r": STOP
9110 CLS : RETURN
9200 DATA "2100401100680100"
9210 DATA "18EDB0C93E51CD28"
9220 DATA "2DEFA203C5A0C402"
9230 DATA "1B3134372F0F38CD"
9240 DATA "D52DFD77703A085C"
9250 DATA "FE20CAC516EF3131"
9260 DATA "E50522A305A10F34"
9270 DATA "372F04013104E531"
9280 DATA "04010328C30238CD"
9290 DATA "D52D4847CDAA223E"
9300 DATA "2884670620C50608"
9310 DATA "C5CB06301AE5EF34"
9320 DATA "377F31E40105A304"
9330 DATA "20E3040338CDA22D"
9340 DATA "FD4670CDE922E1FD"
9350 DATA "346EC110DB2CC110"
9360 DATA "D4EF31360090A103"
9370 DATA "3137008A0238C900"
```

This program's not just long; it's a
bit slow, too, taking some 8 minutes
on average to complete the job. So

here's the challenge - for those of
you with a SAM, firstly, see if you
can adapt it to work in mode 1
(Spectrum SCREEN$) really quickly. And
now, what about modes 2, 3 and 4....

Next, Alan mentions a letter from
Anthony Lenaghan of Sharjah Soccer
Club, United Arab Emirates, to the
Peek and Poke column in Popular
Computing Weekly, December 1985,
containing the following superbly
trivial and fascinating snippet.

```
 10 BORDER 0
 20 BORDER 2: BORDER 5
 30 GOTO 20
```

Fine. Now, if you RUN it, you get a
border pattern of thin red stripes and
thicker cyan ones moving upwards. If
you hold down SPACE, then the stripes
move downwards fairly quickly. Any
other key except SYMBOL SHIFT still
causes the stripes to move down, but
more slowly. Finally, holding down
SYMBOL SHIFT pauses the movement of
the stripes. What? Not the same? Well,
you haven't been using the original
48k Sinclair rubber-keyed beermat that
Alan was using, for the effect varies
from model to model of Spectrum, or so
Alan understands. Three nice little
twists here. Firstly, if you type this
program into SAM and run it, nothing
moves. Why not? Secondly; how does it
work, anyway? Come on, there must be a
rational explanation. And thirdly, now
you know how it works, how about a SAM
equivalent?

And now Alan provides this month's
Really Useless Challenge. The program
below, "keydef", is intended as a
small subroutine for inclusion in
games programs to enable the player to
define keys for the obvious functions
"up", "down", etc. "Well", you might
say, "That's pretty boring". True, and
it would not have been included at all
except for the fact that it uses the
odd effect of POKEing 23681 ("Not
Used" according to the manual, but in
fact the high byte of PR CC, and
normally 91). This POKE, in
combination with LPRINT prints large
characters on the screen. Alan
mentions that he sent me something

similar some time ago, but that it is
interesting to see the effect used for
real. Does anyone know how it works?
Alan (and I) are baffled. I mean,
normally LPRINT shouldn't print to the
screen. And, in addition, you've got
these large characters. What on
earth's happening? Anyone care to
offer an explanation?

Anyway, here's the program...

```
  1 REM Key define program
  2 REM by A Watson
  3 REM Popular Computing Weekly
  4 REM December 15-21 1988
 10 CLS : FOR I=64 TO 71: POKE 23681,
    I: LPRINT "PRESS THE KEY FOR ":
    NEXT I
 20 READ I$: GOSUB 90: LET U$=I$
 30 READ I$: GOSUB 90: LET D$=I$
 40 READ I$: GOSUB 90: LET L$=I$
 50 READ I$: GOSUB 90: LET R$=I$
 60 READ I$: GOSUB 90: LET F$=I$
 70 READ I$: GOSUB 90: LET P$=I$
 80 READ I$: GOSUB 90: LET A$=I$: CLS
    : RETURN
 90 FOR I=72 TO 79: POKE 23681,I: LPR
    INT I$: NEXT I
100 LET I$="": PAUSE 10
110: PAUSE 0: BEEP .01,2: LET I$=INKE
    Y$: RETURN
120 DATA "....UP:",".DOWN:",".LEFT:",
    "RIGHT:","..FIRE:","PAUSE:","ABOR
    T"
```

Next, we've got some SAM offerings
from David Finch of Haxby, York. (He's
the man with the wonderful
letterheading - DAF done as a sort of
fancy design at the top). He tells me
he's enclosing some perfectly
pointless programs; well, pointless
unless you have a use for them. The
first is short and simple. It takes a
screen and produces a shrunk down
image in the bottom left hand corner.
Anyone got a use for this? What am I
offered? Anyway, here is "shrink", the
program to resize your screens.

```
 10 INPUT "Screen to load:";scr$
 20 INPUT "Screen to save:";nscr$
 30 LOAD scr$ SCREEN$
 40 LET yos=-18
 50 FOR x=0 TO 255 STEP 2
 60 FOR y=191 TO 0 STEP -2
 70 PLOT PEN POINT(x,y);x/2,y/2
 80 NEXT y
 90 NEXT x
100 PAUSE
110 SAVE nscr$ SCREEN$
120 GOTO 10
```

The next program, "SCREEN$+R" is
again one of those programs that is
only useful if you really need it.
Again, it takes a screen, and this
time searches for a certain pen,
replacing it with another. It was
intended for when you run out of
colours with FLASH!. Normally, there
are at least two almost identical
colours which you have used, and for
these you can in fact invariably
substitute just one pen without any
noticeable effects on the picture.
Filling in all the areas can be
fiddly, especially if it is patchy.
The program checks every pixel, so it
is slow but thorough. It wouldn't be
difficult to extend the program's uses
by defining an area in which to
search. Here is the program.

```
 10 CLS #
 20 LET yos=-18: CSIZE 8,8
 30 INPUT "Filename of screen:";fn$
 40 INPUT "Filename to save:";sfn$
 50 LOAD fn$ SCREEN$
 60 GRAB scr$,0,41,256,41
 70 PRINT AT 19,0;"Search pen (16=Sa
    ve) Replace Pen"
 80 PRINT ,,,,
 90 PRINT AT 21,0;"Load:";fn$; AT 21
    ,16;"Save:";sfn$
100 INPUT "Search pen:";sp
110 PRINT AT 20,5;sp
120 IF sp>16 THEN GOTO 100
130 INPUT "Replace pen:";rp
140 PRINT AT 20,25;rp
150 IF rp>16 THEN GOTO 130
160 IF rp=sp THEN GOTO 100
170 IF sp=16 OR rp=16 THEN PUT 0,41,
    scr$: SAVE OVER sfn$ SCREEN$
180 PAUSE 20
190 PUT 0,41,scr$
200 BORDER sp: PALETTE sp,127: PALET
    TE sp,0 LINE 96: PALETTE sp,127
    LINE 95
210 FOR y=1 TO 192
220 FOR x=0 TO 255
230 IF POINT(x,96)=sp THEN PLOT PEN
    rp;x,96
240 NEXT x
250 ROLL 2,1
```

```
260 NEXT y
270 BORDER 0: PALETTE sp,127
280 GOTO 60
```

Finally, there is "SCRSTRETCH", another pointless program to play about with, as David puts it. All it does is to stretch the screen vertically. Keys Q, A and Z then display the top half, the second and third quarters and the bottom half, respectively. I guess someone has a use for this program, but in spite of the fact that David's written it, he can't currently find an application for it. Can you?

```
 10 SCREEN 1: INPUT "Screen filename
    :";fn$
 20 LOAD fn$ SCREEN$
 30 LET yos=-18
 40 FOR a=2 TO 5
 50 CLOSE SCREEN a
 60 OPEN SCREEN a,4
 70 NEXT a
 75 DISPLAY 1
 80 DIM scr$(192,130)
 90 FOR a=1 TO 192
100 GRAB scr$(a),0,a-1,256,1
110 NEXT a
120 SCREEN 2
130 FOR a=1 TO 96
140 putline a,0
150 NEXT a
160 SCREEN 3
170 FOR a=49 TO 144
180 putline a,48
190 NEXT a
200 SCREEN 4
210 FOR a=97 TO 192
220 putline a,96
230 NEXT a
240 LET a$="a"
250 IF a$="z" THEN LET s=2: ELSE : I
    F a$="a" THEN LET s=3: ELSE : IF
    a$="q" THEN LET s=4: ELSE LET s
    =1
260 DISPLAY s
270 GET a$
275 IF a$="s" THEN SCREEN 5: DISPLAY
    5: INPUT "Filename:";fn$: SCREE
    N s: DISPLAY s: SAVE fn$ SCREEN$
    : GOTO 270
280 GOTO 250
290 DEF PROC putline x,d
300 PUT 0,2*(x-d)-1,scr$(x)
310 PUT 0,2*(x-d)-2,scr$(x)
320 END PROC
```

While I'm dealing with SAM, it's worth mentioning that Chris Dodd of Thornbury, Bristol, has sent in a whole clutch of programs, many being SAM specific; some working on the Spectrum as well. One of them is the "Flag" program: sorry; can't do; it's still in Amstrads' instruction book which is copyright, and Amstrad are still here! Otherwise, there's quite a lot of good stuff. I haven't room for it all this month, but how about "sunset", which does just that.

```
  1 MODE 4
  5 BORDER 0: PAPER 0: PEN 6: CLS
 10 FOR n=1 TO 80
 20 PLOT 127,75
 30 DRAW INT (RND*250)-125,INT (RND*
    97)
 40 NEXT n
 50 FOR n=75 TO 0 STEP -15
 60 PLOT 127,75
 70 DRAW -127,-n: PLOT 127,75: DRAW
    127,-n
 80 NEXT n
100 FOR n=-127 TO 127 STEP 20
110 PLOT 127,75
120 DRAW n,-75
130 NEXT n
```

Or, for that matter, some embroidery..

```
  1 MODE 4
  2 PALETTE
 10 FOR n=0 TO 255
 15 PEN RND*8
 20 PLOT 0,0: DRAW n,173
 21 PLOT 0,173: DRAW n,-173
 30 PLOT 255,0: DRAW -n,173
 50 PLOT 255,173: DRAW -n,-173
 60 NEXT n
 70 GOTO 10
```

Or even this interesting little snippet, called appropriately enough, "Square".

```
 10 MODE 1
 20 PALETTE
 30 PAPER 0: BORDER 0: PEN 7
 40 CLS #
 50 LIST FORMAT 2
 60 PLOT 0,175
 70 LET t=255: LET r=-175: LET b=-25
    5: LET l=169
 80 DRAW t,0: DRAW 0,r: DRAW b,0: DR
```

```
   AW 0,1
 90 LET t=t-6
100 LET r=r+12: LET b=b+12: LET l=1-
    12
110 IF R=185 THEN PAUSE : STOP
120 DRAW t,0: DRAW 0,r: DRAW b,0: DR
    AW 0,1
130 LET t=t-6
140 GOTO 90
```

We've had a more complicated version of this before, haven't we... Anyway, here's the simple one - a little version of "password"...

```
 10 LET p$=""
 20 PRINT "enter password"
 30 PAUSE 0
 40 LET k$=INKEY$: IF k$="" THEN GOT
    O 40
 50 LET p$=p$+k$
 60 IF LEN p$<>5 THEN GOTO 30: REM L
    ENGTH OF PASSWORD
 70 IF p$<>"HELLO" THEN STOP : REM P
    ASSWORD
 80 PRINT "ok"
 90 REM rest of prog
```

Regular readers of this column will know that in the June Short Spot, I challenged you to find a short, crisp Universal Calendar program. Regular readers will also have come across Ettrick Thomson (from Aldeburgh, Suffolk), before. Well, I knew he would come up with something. Here's his solution, working on SAM...

```
 10 REM weekday - Ettrick Thomson
 20 LET d$="Sunday  Monday  Tuesda
    y WednesdayThursday Friday  Sa
    turday "
 30 LET m$="January February March
       April    May     June    Ju
    ly   August  SeptemberOctober
    November December "
 40 DO : INPUT #2;"day:";dy;", month
    (1-12):";mh;", year(eg 1992):"yr
 50 LET lp=NOT yr MOD 4 AND yr MOD 1
    00 OR NOT yr MOD 400
 60 LET dm=31-(mh MOD 2<>(mh<8))-((1
    +NOT lp) AND mh=2)
 70 LET valid=1<=dy AND dy<=dm AND 1
    <=mh AND mh<=12 AND (yr>1752 OR
    yr=1752 AND mh>9)
 80 IF NOT valid THEN PRINT "date no
    t valid: please re-enter"
 90 LOOP UNTIL valid
```

```
100 LET y=(yr-(mh<3)) MOD 400,m=mh-2
    +(12 AND mh<3)
110 LET wd=1+(INT (1.25*y)-y DIV 100
    +INT (2.59*m-2.07)+dy+2) MOD 7
120 LET u=2*dy MOD 10 AND dy DIV 10<
    >1 AND dy MOD 10<4
130 PRINT TRUNC$ d$(9*wd-8 TO 9*wd);
    ", ";dy;"thstndrd"(u+1 TO u+2);"
    ";TRUNC$ m$(9*mh-8 TO 9*mh);" "
    ;yr
140 GOTO 40
20000 DEF PROC Lstprg L1,L2,m,w,n
20010 DEFAULT m=0,w=40,n=0
20020 LOCAL e$: POKE SVAR 14,w-1
20030 LET e$=CHR$ 27: OPEN #5,"b"
20040 PRINT #5,e$;"M";e$;"~";"1";e$;"R
      ";CHR$ n;e$;"1";CHR$ m
20050 LLIST L1-1 TO L2
20060 PRINT #5,e$;"~";"0";e$;"@": CLOS
      E #5
20070 POKE SVAR 14,79
20080 END PROC
```

Ettrick originally had a program adapted from a further, already published, Spectrum program. To adapt this further would mean a thoroughly cobbled job, so he started again and wrote this from first principles, for SAM at first, with modifications for the Spectrum at the end. This program handles any date in the Gregorian calendar, not just the 20th century. So, in Lines 20 and 30, each name is padded out with spaces if necessary to make 9 characters. Lines 40-90 accept and check the input, rejecting, for example, 31 April, or 29 February, 1900, and any date before the introduction of the Gregorian calander in England (September 1752). In Line 50, lp=1 for a leap year, 0 for an ordinary year [try not to use lowercase L in listings]. In Line 60, dm is the number of days for the given month and year combination; it depends on the fact that from Jan to Jul, the odd-numbered months have 31 days; from Aug to Dec, it's the even ones. For other months, one is knocked off 31, making 30, and, for February, a further one for a leap year, and two for an ordinary year.

Once all this has been sorted out, the manipulation starts. To simplify matters, Line 100 changes the start of the year to 1st March; m=1 up to 12

for February, which requires the year to be decremented for a January or February date. This change means that 29 February is the last day of the year, which simplifies certain calculations. The second change is the "MOD 400". There are 400 years in the leap-year cycle, and, by a happy chance, there are 146097 days, exactly 20871 weeks in it. Thus the dayname/dayno./month combination repeats every 400 years. So whatever day, month, year is specified at Line 40, it becomes a date dy:m:y at Line 100 with 0<=y<399, and leap years arise in years 3,7,11,...,395,399, except that years 99,199,299 are not leap years.

Line 110 gets wd, weekday number - 1 for Sunday up to 7 for Saturday. The expression INT(1.25*y)-y DIV 100 gives the number of effective days in years 0 to y-1. Since a normal year is 365=52*7+1 days, in calculating the day of the week, it has one effective day; a leap year has 2; INT(1.25*y) produces leap years at 3, 7,...and -y DIV 100 eliminates the non-leap years 99, 199, 299. The expression INT(2.59*m-2.07) gives the number of effective days from the beginning of the year up to month m-1; a 31-day month has 3 effective days, a 30-day month has has 2; the constants 2.59, 2.07 were determined by trial and error. Note that the number of days in February is not needed; if the change of Line 100 to a year with March the first month had not been made, the expression would have been (3*(mh-1) AND mh>3)+(INT (2.62*mh-4.47+lp) AND mh>2).

In the PRINT of LINE 130, ornamenting the day number by "th", "st", and so on is achieved by selecting the right pair of characters from the string "thstndrd" according to the value of u,worked out at Line 120; u=0,2,4,6 if the day unit digit (dy MOD 10) is 0,1, 2,3 and u=0 for other values of the unit digit; u is also zero if the 10s digit (dy DIV 10) is unity.

A Spectrum version is also possible with the following modifications. AT Line 40, a plain INPUT would have to

be used, followed by, say, PRINT "Date ";dy;"/";mh;"/";yr. The DO of Line 40, and the whole of Line 90 would disappear, and "GO TO 40" would have to be added to Line 80. For the functions MOD and DIV, define a function with:-

    DEF FN M(n,m)=n-m*INT(n/m)

DIV m=INT(n/m). The SAM TRUNC$, used in Line 130 is awkward. A possibility is LET d$="6Sunday 6Monday ...", each name being preceded with the number of letters in it: similarly for m$. Then, at Line 110, make wd ten times as large; i.e.

LET wd=10*(1+FN M(INT(1.25...+dy+2,7)

For Line 30, PRINT d$(wd-8 TO wd-9+VAL d$(wd-9));","... with the slicer for m$ being the same, but with wd replaced by 10*mh.

It all takes a lot longer to explain than the program, but I hope that Spectrum owners can also get this to work.

Many thanks, Ettrick...

And that's all we can fit in this month. Please keep the bits and pieces coming to John Wase, Green Leys Cottage, Bishampton, Pershore, Worcs, WR10 2LX. Many thanks.

- oOo -

'It's amazing how the fish always take their holidays at the same time as you'

SPELL MASTER

Reviewed By:- Carol Brooksbank.

I have always prided myself on being quite a good speller, and therefore as having no need of a spell checker. However, I am not a good typist, and had not realised until I received SPELLMASTER - FRED's new spell checker and puzzle solver for the SAM Coupé - that a spell checker will also sniff out typing errors. It has already saved me hours of 'proof reading'. Usually a couple of typing errors get past me - none get past SPELLMASTER.

The usefulness of a spell checker depends on the size of its dictionary. The dictionary for the 512K SAM is splendid, 80,000 words. I was quite surprised at the number of words which I would think of as reasonably uncommon which are included. AARDVARK is not a word I use every day. And when I came to write one of my theological pieces, I found it could cope quite happily with ISRAELITE, INDIGENOUS and MONARCHIST. The dictionary selected if you are using a 256K SAM is smaller, but I have not checked its exact contents, because with my 512K machine the 512K dictionary is loaded automatically.

There is also a user dictionary, empty until you start using it, which is loaded automatically too. If you are checking through a document and the program stops to query a word which is not in either dictionary, pressing key A will add it to the user dictionary. The main dictionary holds no plurals - it will accept NARRATOR but query NARRATORS. I find it handy to add the plurals to the user dictionary, so that less and less words are queried as your user dictionary grows. And if you are in the habit of using words like ESCHATOLOGICAL, they can go in the user dictionary too. If you like, words can be added to the main dictionary too, and you can browse

through either dictionary and delete from either. In fact, there is nothing to stop you from maintaining several user dictionaries, each containing words you meet in a particular field, and loading the one you want. You have to save the user and main dictionaries at the end of a session if you have made any changes to them - changes are made in memory, not on the disc.

I am glad that SPELLMASTER does not just change a word, as some spellcheckers do, or offer you words it thinks you might mean. It simply highlights the word it does not recognise, and you can opt to ignore it - i.e. leave it alone - add it to the dictionary, or correct it. You have to type in the correction which is then inserted and the justification, if any, adjusted.

If you are not sure what the correct spelling should be, you can exit checking mode and browse through the dictionaries till you find it. You can browse word by word if you have a day or two to spare. Alternatively, there are facilities for jumping to the next letter of the alphabet - from words starting with A to those starting with B; to the next letter of the alphabet in the second letter of the word - from AA to AB; or jumping 50 words at a time; so it takes very little time to locate a particular word. All the browse facilities are available for moving backwards too.

SPELLMASTER is also a simple word processor. Although it lacks things like printer controls, block moving and word finding, it does cope with wordwrapping and justification, and is perfectly adequate for running off short notes. And, you can set it to spell check as you type.

But the program is not, of course,

limited to checking text files written using its own word processor. I use it with files written with THE SECRETARY. Even after alterations which involved reformatting the text, I could import it back into THE SECRETARY, delete a line which had been added at the end, and which appears as a line of question marks in THE SECRETARY , and re-save it as a SECRETARY file. The embedded printer controls were unaffected by the transfer to SPELLMASTER, though the file had to be imported, rather than re-loaded, and then saved. It was more difficult to use with WORDMASTER files, but this is because WORDMASTER text is not saved in lines, but as one continuous file. The justification is worked out at the printing stage, and is not shown on screen. This meant that the wordwrap was upset when the file was loaded into SPELLMASTER, which expects the file to be divided into lines with the justification spaces already entered. I have not tried it with any others, but the publishers say it can cope with files from TASWORD and OUTWRITE.

There are two bonus facilities with SPELLMASTER. One is an anagram solver. Enter a group of letters, and the program will list all the words in its dictionary which contain only those letters. I entered LVIE, and the following list was offered:-

        EVIL LEVI LIVE VEIL VILE

The other bonus is a crossword solver. You enter the letters you have, with # to represent the missing ones. E#E#T##LL# produced: EVENTUALLY EVENTFULLY

In crossword mode, * can be used as a "wild card", indicating that you want to see all the words of any length which contain the given letter pattern at the beginning. A#* will get you all the words starting with A.

At £14.99 SPELLMASTER is excellent value for money. Whether you are a serious writer who can't spell, a good speller who can't type, or a crossword/anagram addict, this program will make your life easier.

FRIENDLY PROGRAMMING
Or How Write A Good User Interface

By:- Alan Davis.

When someone else's program crashes on you its "bad programming" but when your own program crashes who can you blame? User friendliness begins at home.....

Let's begin with a tall story, but one with a moral. Its called "The Shape of Things to Come?".....

Sid had just bought a book, and on arriving home he curled up in his favorite armchair to enjoy it. Unfortunately it was the one of those new-fangled books fitted with a complicated copyright-protection device to defeat photo-copiers, and so it took him a quarter of an hour just to get it open. Still, he did manage it in the end, and he settled down to read in comfort. After he'd read a couple of chapters, his fingers accidentally turned two pages over instead of one. Now the designer of the book hadn't anticipated that anyone could do such a silly thing - so the page-opening mechanism jammed, and the book snapped shut. "Oh well, my fault" thought Sid, philosophically. (It took him another 14 minutes to get it open again). This time he took great care not to turn two pages at a time, but after a while he just couldn't resist trying to sneak a look at the last page to see if it really was the butler who did it - and at that point all the pages fell out......

Now I know that this is an odd way to begin an article on programming. But substitute "software " for "book", and I think you'll see what I'm getting at. We've all bought (and perhaps - heaven forbid! - even written) programs which were poorly error-trapped, crashing without warning as the result of an injudicious key-press; we're all distressingly familiar with those

barbaric "protection" methods which hang-up the machine when you press BREAK; and of course the monstrous LENSLOK has found no difficulty at all in achieving a place in the top ten list of contributors to the misery of mankind.

Only one person suffers from all this: the user of the program. In principle, there's no reason why he shouldn't find using a computer program almost as straightforward as reading a book, but the fact is that many programs give him a rough ride. So I thought it might be a good idea if we tried to find a few ways of making life easier for him when writing our own programs. The keyword, then, is "friendly", and friendliness is always worth striving for, no matter what kind of program you're writing. it doesn't even matter if the program is a utility being written only for your own use - because we all make mistakes; and the last thing you want is a poorly crash-proofed utility which leaves you in a mess after maybe hours of work, just because you pressed the wrong key by accident.

Usually, the friendliness of program isn't determined by ingenious programming. Rather, it depends on the programmer painstakingly searching for possible weak points and anticipating potential errors in such a way that they're rendered harmless to the program - and by implication, harmless to the user. From our present point of view, the danger points will occur at places in a program where some kind of input is needed from the user and it s predominantly this area that we'll be looking at in this article.

On the whole, the friendliest way of getting input from the user is probably to present him with a menu of options, and ask him for a single prod

at the keyboard to make his selection.

This automatically puts a limit to the silly things he might try to do, and has the great advantage of being easy to understand. it also means that our task of error-trapping is made very straightforward.

My first listing below is the sort of routine one might use here. it presents a choice of three actions (pointless ones, here - but this is only an example) determined by pressing key 1,2, or 3 - and is about as simple to operate as any program could be. In fact, short of pressing BREAK, the user simply can't crash the program - because line 40 rejects every keypress except the three allowable ones.

```
1 REM *** Simple menu routine ***
10 CLS : PRINT AT 8,11;"OPTIONS";AT
   10,5;"1: Do something"' TAB 5; "2
   : Do something else"' TAB 5;"3: D
   o something different"
20 PRINT #1;AT 0,8; INVERSE 1;"PLEAS
   E SELECT..."
30 PAUSE 0: LET I$=INKEY$
40 IF I$<"1" OR I$>"3" THEN GOTO 30
50 BEEP .1,30
60 GOSUB 100*VAL I$
70 GOTO 10
100 CLS : PRINT "I've done something"
    : GOTO 400
200 CLS : PRINT "I've done something
    else": GOTO 400
300 CLS : PRINT  "I've done something
    different"
400 PRINT #1;AT 0,8; INVERSE 1;"PRESS
    A KEY....."
410 PAUSE 0
420 RETURN
```

Whenever it's appropriate a menu-driven utility program gets my vote every time. But programming (like life, alas) often presents us with circumstances which can't be tackled in quite the way we might like. Sometimes a single key-press menu just won't do, and this is generally the point at which our program's "friendliness" can start to acquire rough edges. I can't cover all eventualities, of course - it would take a lifetime! But we can learn a

good deal by taking one specific example and delving into it thoroughly - because it's really the thinking process underlying this which is important, rather than the example itself.

The example I've chosen is one which commonly arises in programs of many types, namely, where the program requires a number (which may be several digits long) to be entered by the user. We'll restrict the discussion to integer, here - and let's also add the arbitrary condition that for some reason peculiar to the situation the number mustn't exceed 50. This is just the kind of thing which could arise in a typical programming situation.

On the face of it, the next listing would appear to be a straightforward answer to the problem. it uses the simple Basic "INPUT" command to assign a value to N, checks to see whether the value entered is permissible (N mustn't exceed 50 remember), and prints an appropriate comment. It works, of course; but as a piece of "friendly programming" it is a dead loss because there are so many possible ways of crashing it. Just for starters, try entering a letter, or several letters, or even (since there's nothing to stop you) something like VAL z$!) Gruesome, isn't it?

```
1 REM *** Simple INPUT routine ***
10 INPUT "Give me a number ";N
20 IF N>50 THEN PRINT "Sorry, that n
   umber is too big.": GOTO 10
30 PRINT "That will do, thanks."
40 PAUSE 100
50 GOTO 10
```

What can were do to improve matters? Well, a solution which sorts out the major difficulties is given in my third listing below. We're still using the INPUT command - but now we're picking up the entry not as a numeric variable (N) but as a string (I$). This gives more power to our elbow, because we can now add a little error checking subroutine (lines 500-550). This rejects the input if the entry is either an empty string, or if any of

the characters are not pure numbers between 0 and 9, and returns to the user with an appropriate comment and a repeat request. On the other hand, if all is well on return from the subroutine line 30 assigns the correct value to N, and Bob's your uncle (or editor in my case).

```
1 REM *** Error trapped input ***
10 INPUT "Give me a number "; LINE I
   $: GOSUB 500
20 IF FAIL THEN PRINT "Numbers only
   please": PAUSE 100: GOTO 10
30 LET N=VAL I$
40 IF N>50 THEN PRINT "Number too bi
   g.": PAUSE 100: GOTO 10
50 PRINT N;" will do.": PAUSE 100: S
   TOP
500 LET FAIL=0:IF I$=""THEN LET FAIL=
    1: RETURN
510 FOR I=1 TO LEN I$
520 IF I$(I)<"0" OR I$(I)>"9" THEN LE
    T FAIL=1: RETURN
530 NEXT I
540 RETURN
```

This isn't a bad solution to our problem, in fact - and you may well consider it good enough. but it's by no means perfect, because the INPUT command still has a couple of nasty tricks up its sleeve. Try pressing CAPS SHIFT and 6 at the same time, for example.

Alternatively, type in lots of numbers - say a couple of rows, Oops! So unless you have a quaint fondness for the "STOP in INPUT" and "Number too big" error reports. it looks as though we'll have to continue our search for the ultimate in friendly input routines.

Obviously, to improve matters further, we'll have to abandon the INPUT command altogether, and simulate a similar command of our own which allows the program to intercept every character as it's typed. This could be tackled in several ways, and the next listing shows one of them.

```
1 *** Using INKEY$ ***
10 LET N$="": LET I=0: PRINT #1;AT 0
   ,0;"Give me a number"
20 PRINT #1;AT 1,I;">":: PAUSE 0: LE
```

```
T I$=INKEY$: LET A=CODE I$: IF (A
   <48 OR A>57) AND A<>13 AND A<>12
   THEN GOTO 20
30 IF NOT I AND (A=13 OR A=12) THEN
   GOTO 20
40 BEEP .05,30
50 IF A=12 THEN GOSUB 500: GOTO 20
60 IF A=13 THEN PRINT #1;AT 0,0,,,,:
   GOTO 80
70 IF I=5 THEN GOTO 20
80 PRINT #1;AT 1,I;I$;:: LET N$=N$+I$
   :LET I=I+1:GOTO 20
90 LET N=VAL N$: IF N>50 THEN PRINT
   "Sorry, that number is too big."
   : GOTO 110
100 PRINT "OK, the number ";N;" will
    do for me."
110 PAUSE 100: GOTO 10
500 LET I=I-1: LET N$=N$( TO LEN N$-1
    ): PRINT #1;AT 1,I;" ": RETURN
```

The main bulk of the error-trapping is done in line 20; because we're reading the keyboard using INKEY$, we can examine each keypress as it comes, and ignore it if it isn't either a number (CHR$ 48 - CHR$ 57), ENTER (CHR$ 13) or DELETE (CHR$ 12). This still leaves quite a lot to be done, though. The variable "i" keeps count of the number of valid characters typed, so that line 30 can prevent the user attempting to ENTER or DELETE a non-existent number. Line 70 solves the problem of the "Number too big" error by limiting the number of digits that can be typed to five. Line 80 builds up the string N$ one character at a time, and when ENTER is pressed, line 90 extract the value of N (i.e. VAL N$) that we need. (line 500, by the way, is the DELETE subroutine).

Although there are alternative lines of approach, this seems to be about as far as you could go in BASIC. It's as crash-proof as most menu routines, in that only BREAK will defeat it. if you type it in and try it, though, you may notice a marginal sluggishness in the keyboard response. it's only very slight, but should you wish to add extra checks between each keypress (for some other specific application) it could become irritating. The problem arises of course because the more work you ask the program to do between each keypress, the longer it

will take to do it.

The only comprehensive solution to this is to read the keyboard and do all the "between keypresses" error checking in machine code, returning to BASIC only when the input is ENTERed. Here is one way of tackling this:-

```
        ORG 65000
;
LAST_K EQU 23560
;
        XOR A ;Clear A and flags
        LD (POINT),A ;Reset pointer
        CALL 5633 ;Open lower screen
START   CALL PROMPT
        XOR A
        LD (LAST_K),A
INPUT   LD A,(LAST_K)
        CP 0
        JP Z,INPUT ;Wait for key
        CP 12
        JP Z,DELETE
        CP 13
        JP Z,ENTER
        CP 58
        JP NC,START
        CP 48
        JP C,START ;Only numbers now
        LD (CHR),A ;Temp store
        LD A,POINT
        CP 5
        JP NC,START ;Abort if >5
        LD HL,STORE
        LD DE,(POINT)
        ADD HL,DE ;Point to store
        LD A,(CHR)
        LD (HL),A ;Store digit
        LD A,E
        INC A
        LD (POINT),A ;Store point
        JP START
PROMPT  CALL SETPOS
        LD A,62
        RST 16 ;Print prompt
        RET
PRINT   CALL SETPOS
        LD A,(CHR)
        RST 16 ;Print digit
        RET
SETPOS  LD A,22
        RST 16
        LD A,1
        RST 16
        LD A,(POINT)
        RST 16
        RET
```

```
DELETE LD A,(POINT)
        CP 0
        JP Z,START; If 0 len
        CALL SETPOS
        LD A,32
        RST 16 ;Print space
        LD A,(POINT)
        DEC A ;Move point back 1
        LD (POINT),A
        JP START
ENTER  LD A,(POINT)
        CP 0
        JP Z,START
        RET ;Back to basic
;
CHR DEFB 0
POINT DEFB 0
STORE DEFS 5
;
END EQU $
```

This works by reading the system variable LAST-K, which stores the code of the last newly pressed key. if this is non-zero, indicating a keypress, then the error-trapping checks are called one after another so that only a valid keypress is accepted. The routine also looks after the printing of digits to the screen, and the deleting process. The actual input is stored in a series of up to five bytes starting at address STORE.

A simple machine code program like this will need a short BASIC subroutine to drive it - such as this one:-

```
 1 REM *** Machine code input ***
 2 REM ** Spectrum Only **
10 LET POINT=65138: LET STORE=65139
20 PRINT #1;AT 0,0;"Enter a number "
30 LET M=USR 65000
40 PRINT #1;AT 0,0,,,,
50 LET Z$="": FOR I=STORE TO STORE-1
   +PEEK (POINT): NEXT I: LET N=VAL
   (Z$)
60 IF N>50 THEN PRINT "TOO BIG!": GO
   TO 80
70 PRINT "OK, ";N;" will do"
80 PAUSE 100: GOTO 20
```

If you want to try this out for yourself you'll need the machine code residing in memory at 65000 - don't forget to CLEAR 64999 beforehand. This new BASIC/machine code combination will behave in exactly the same way as the last basic routine I showed you, except that all trace of keyboard sluggishness has disappeared, together with the added bonus that even BREAK is disabled during the period where an input is being requested. There is absolutely nothing at all that the user can do which will cause a crash. Error-trapping is complete, and the only improvement in the way of friendliness would be to include more detailed prompt messages on screen.

As I said earlier, it's the general process involved which is important, rather than the details; and I hope that the method of progressively isolating the problems and then solving them is clear from the examples I've used here. The extent to which you go will largely depend on how likely it is that others will use your program, of course - and it will also depend on how familiar with computers they're likely to be! People can do very odd things when they're desperate ..... and it's up to the programmer to safeguard the inexperienced user from himself.

I can't leave the subject of friendliness without saying a word about "protection". Some people do seem to get obsessive about this sometimes, and don't regard a program as finished until they've incorporated all the BREAK disabling tricks they can muster. If the result simply means that BREAK is ignored, then fine; this in itself protects the user from stopping the program and getting into a mess. But if pressing BREAK causes the machine to hang up, or a system reset, then surely this is the very opposite of friendly programming! Furthermore, such methods are actually quite pointless. They certainly won't make your program uncopyable, and if all you're worried about is to keep prying eyes from your code, then it stands to reason that anyone who's capable of understanding your program in the first place will have no difficulty in making short work of your protection scheme! The moral? Think twice before you plant a bomb in your program.....

RECURSION
Again & Again

By:- Jeremy Cook.

This is the first of a small number of articles on recursion. I shall try to explain a powerful and often elegant technique, which you may not have come across before (unless you read Thought Spot). There are many places where recursion is useful, sometimes even essential, and it is worthwhile adding it to your programming armoury. It is readily available to SAM users, but involves some tricky DIY on the Spectrum.

The concept itself is not difficult. Basically, a recursive procedure or routine (or definition or whatever) is one which contains a call to itself (a "call" to a procedure is the command that causes the program to run that procedure). What can be difficult is understanding how a recursive routine works, or how to write your own.

To see why a self calling procedure would do anything useful I'll start with a fairly simple example (that you will find in almost every book explaining recursion!). We are going to look at a routine that calculates the factorial of a number (which tells you how many ways there are of arranging objects). Factorial n, where n is an integer, is written n!, and is usually defined thus:-

$$n! = \left\{ \begin{array}{ll} 1 & \text{if } n=0 \\ 1*2* \ \dots \ *(n-1)*n & \text{if } n>0 \end{array} \right.$$

(where the dots stand for any numbers needed in between). So 0!=1, 1!=1, 3!=6, 4!=24, 5!=120 etc. It may not look it, but this function is useful in many applications. A consequence of this definition is that 5!=4!*5, 4!=3!*4, etc. Generalising this gives a second, recursive, definition for n!:-

$$n! = \left\{ \begin{array}{ll} 1 & \text{if } n=0 \\ (n-1)!*n & \text{if } n>0 \end{array} \right.$$

A routine to calculate this could be as below. The first one is for Spectrum. Try it.

```
10 LET F=0
20 INPUT "Enter number";NUM
30 LET N=NUM: GOSUB 1000
40 PRINT NUM;"! = ";F
50 GOTO 10
1000 REM n factorial
1010 IF N=0 THEN LET F=1: GOTO 1040
1020 LET N=N-1: GOSUB 1000
1030 LET F=F*N
1040 LET N=N+1: RETURN
```

For SAM change to the following lines, which are numbered to correspond approximately to the actions of the above program.

```
30 FACTORIAL NUM
1000 DEF PROC FACTORIAL N
1010 IF N=0: LET F=1
1020 ELSE FACTORIAL N-1
1030 LET F=F*N: END IF
1040 END PROC
```

You'll notice that the Spectrum version does some extras that SAM BASIC does automatically. At this point I digress quite a lot before explaining the program; please bear with me. To be able to understand and use recursion it is often useful to know how procedures work. I'm afraid most of the explanations are for SAM BASIC, but Spectrum users should read it also. It will help considerably if you can understand how procedures and recursion work on SAM, since that is what you must emulate.

When a procedure is called, a temporary variables area for that procedure is set up. This area holds all the variables local to the procedure. This includes the parameters, which take their values from the procedure call (eg n takes its first value from NUM in line 30), and any variables specified in a LOCAL command. In the above example, the only local variable is the parameter N. Thus when line 30 is executed, an area is created for N, which is given the value of NUM.

This variables area is located on

the BASIC stack, which is a place in memory that the computer uses for temporary storage. Its action is like any other sort of stack: you can push things onto the top, or pop them off the top. The details are not important here, but in effect, when a procedure is called, a whole temporary variables area is pushed onto the top of the stack.

Once a procedure is finished, the associated variables area is removed from the stack. This is why they are local variables: outside the procedure they do not exist. In the example above, if SAM users replace line 50 with "50 PRINT N" and RUN, then they will get a variable not found error - N exists only in the procedure.

The important thing to note from all this is that every call, whether recursive or not, pushes a new area onto the stack, and that when an area is removed, all the variables are put back as they were before that particular call. (The call in line 1020 is recursive, because it is calling the procedure it is in, but the call in line 30 is not recursive).

I personally think of a procedure call as creating a new layer of variables, with each call placing a new layer above existing ones. These layers form a "tower" over the global variables (ie. the variables that aren't local to any procedure. In the example NUM and F are global variables). I also consider the layers to have holes in, through which the variables of lower layers can be accessed. But if a layer has a variable name the same as one below, then I think of the "higher" variable "covering" the lower. This means that a recursive call will create a layer that exactly covers the layer just below, since the local variables will all have the same name.

Having talked about the structure that allows recursive programming, I'll now try explaining the program in terms of some of the ideas I've just introduced. Try to follow line by line.

Line 10 initialises F to 0, and let us suppose that 4 is entered for NUM at line 20. Line 30 is the first (non-recursive) procedure call, which takes us from the bottom to our first layer. It is here that the value of NUM is passed to N.

Line 1000 just identifies the procedure (and parameter N in SAM version). Lines 1010 to 1030 correspond to our definition of factorial. But on the first layer N=4, so we go on to line 1020. This is where (N-1)! is calculated. Line 1020 creates a second layer above the first, and N-1=4-1=3 is passed to the N on this layer (remember, the N on layer one below is still 4).

Back at line 1000, but on the second layer. We go past line 1010 again because N=3, and find a third layer created on top at line 1020. On this new layer 3-1=2 is passed to N.

Back at line 1000 on the third layer. As above, a fourth layer, with N=1, and a fifth layer, with N=0, are created. So now, on the fifth layer, we find the IF at line 1010 is true. Thus F is given the value 1. The program now has to skip straight to 1040. This is the end of the procedure, so layer five is removed, and we find ourselves back on layer 4, at line 1030 (just after the call to layer five), where N=1 and we have F=1. Here F is assigned the new value F*N=1*1=1.

Now we proceed to line 1040, which is the end of the procedure again. Thus we find layer four is removed, and we're back on layer three, again at line 1030, with N=2 and F=1. Once again we change F to F*N=1*2=2. At 1040 layer three is removed, and we go back to layer two, where N=3. At line 1030 again, so F becomes 2*3=6.

Then of course the procedure of layer two ends at 1040 and layer two is cleared, just leaving layer one, where N=4. We are again at line 1030, so F becomes 6*4=24. Finally, layer one is removed and we drop back to the bottom, where F=24, NUM is still 4 (and N no longer exists, remember).

We're now ready to execute line 40 (just after the initial call). Hence on the screen up pops "4! = 24", and line 50 means the whole process can start again. I hope that has made things a bit clearer.

Now a few words for the Spectrum users. Speccy BASIC hasn't got procedures, so you have to have your own equivalent of a layer creating mechanism. In this program N is easy to keep track of because it changes predictably from layer to layer; hence the N=N-1 and N=N+1 in lines 1020 and 1040. Normally an array is required, but more of that next time.

That is probably enough for now. I am far from an expert on recursion and so I may have made some technical errors, but I believe that my layers concept is basically sound. If anybody has any comments or opinions on that, then send them to my address given in Thought Spot.

In the next part I intend to look at actually writing recursively, and give some criteria for successful recursion. I might also look at the advantages and disadvantages of recursion. So be here! Bye.
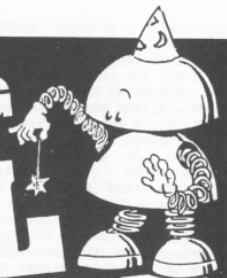
'If you're not using your wallet tonight Dad, can I borrow it?'

MACHINE CODE
WITHOUT THE TEARS

Part 13.   By:- Carol Brooksbank.

Today we will write a new version of our program which will make the tractor disappear off the edge of the screen on the right. Then we shall modify it so that the tractor will reappear on the left as it disappears off the right. Fig.1 summarizes the difference between the versions.

Last month, you deleted lines from the original tractor drawing program before we changed it, and I suggested you save a copy of the shortened program for future use. Load this copy into your assembler.

Up till now we have been thinking of the tractor as being on a 7-column frame. We moved it across the frame and then moved the frame reference up. Now, we are going to think of the tractor frame as the whole width of the screen. We shall still move a line of bytes one bit at a time, but now the line will be 32 bytes long, and we shall not have to move the frame reference. We shall write this in the form of another library routine called "SCROLL", which you will be able to use with any program. It will be possible to use this routine to scroll and clear a window of any size, in any position on screen, and even use it to clear the screen by scrolling everything off the right hand edge.

So, first we shall change our original program a little more. Ensure that COLUMN and ROW are set at 0, so that the tractor is drawn in the top left corner of the screen.

The section we deleted last month left us with:-

```
DJNZ CELLOOP
CALL SCREENOUT (Sam only)
RET
```

at the end of the main loop of the program. Insert, immediately after DJNZ CELLOOP:-

```
POP HL
CALL SCROLL
```

Our original program will now only draw the tractor and call the scroll routine to move it. If you like having the attributes set to PAPER green, INK black, you can insert the 5 lines from last month's program which start at the label SETATTR. They go between DJNZ CELLOOP and POP HL. If you decide to leave them out, the PAPER and INK colours will be those current when you load the code and run it.

THE DISAPPEARING TRACTOR
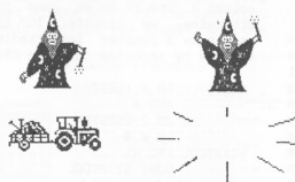
IF WE TREAT THE TRACTOR'S FRAME AS BEING THE WHOLE WIDTH OF THE SCREEN, AND RESET BIT 7 OF THE FIRST BYTE EVERY TIME WE SCROLL A LINE BY ONE PIXEL TO THE RIGHT, THE TRACTOR WILL MOVE ACROSS THE SCREEN AND DISAPPEAR OFF THE RIGHT HAND EDGE.

IF WE PRESERVE THE BIT STATUS OF BIT 0 OF THE LAST BYTE AND COPY IT INTO BIT 7 OF THE FIRST BYTE AFTER SCROLLING THE LINE, THE TRACTOR WILL REAPPEAR ON THE LEFT OF THE SCREEN AS IT DISAPPEARS OFF THE RIGHT.

Fig.1.

You can delete LASTCELL. We will use DEC HL in its place, because we are only ever moving backwards within a screen line, so straddling a third boundary will not be a problem. But remember when writing your own programs that you must be certain that there will be no third boundary problems before abandoning LASTCELL or NXCELL. We have used NXCELL in the original program, so we must keep that. We must also go on using NXDOWN, because this is to be an all-purpose window scrolling program, and it must be able to cope with any window, even if its rows cross a third boundary. So, when this routine is finished, if you want to save it as a library routine, you must save NXDOWN with it.

We need two other new subroutines. They require the size of the window involved to be stored in four variables. LEFT and RIGHT store the width boundaries, in column numbers, and may be 0-31. TOP and BOTTOM store the height boundaries, in row numbers, and may be 0-23.

```
FINDWID  LD A,(LEFT)
         LD B,A
         LD A,(RIGHT)
         SUB B
         LD B,A
         RET
```

The left column is fetched in A and transferred to B. The right column is fetched in A. SUB B deducts the left from the right leaving A holding (width-1). This is transferred to B before the routine exits. (Width-1) is a convenient dimension for us, because when we are scrolling a line we usually repeat (width-1) times. When we need the true width we must make INC B the first instruction after CALL FINDWID.

```
FINDEP   LD A,(TOP)
         LD B,A
         LD A,(BOTTOM)
         SUB B
         INC A
         LD B,A
         RET
```

In the same way, this subroutine finds the depth of the window we are scrolling. This time, we INC A before transferring the answer to B, because we always need the true depth.

```
TOP      DB 0
BOTTOM   DB 2
LEFT     DB 0
RIGHT    DB 31
```

The main scroll routine starts here, with the bytes storing the window dimensions. Our tractor is 3 cells deep, so the top and bottom rows are 0 and 2 respectively. We want to scroll it across the whole screen, so the window must be the full width, columns 0-31.

```
SCROLL   LD HL,SCREEN
         LD A,(TOP)
         AND A
         JR Z,STAY1
         LD B,A
STARTROW PUSH BC
         CALL NXDOWN
         POP BC
         DJNZ STARTROW
```

The routine is called from SCROLL. It begins by finding the screen file address for the top left byte of the window. The TOP dimension is fetched first. AND A checks whether it is column 0, and if so we jump forward. Otherwise, we transfer TOP to B and use it as a counter for finding the top row by repeated calls to NXDOWN.

```
STAY1    LD A,(LEFT)
         AND A
         JR Z,ONESTEP
         LD B,A
STARTCOL INC HL
         DJNZ STARTCOL
```

Similarly, the start column is found, so that HL holds the top left byte of the window.

From here I have, wherever possible, used the same labels as were in last month's program, so that you can compare the two.
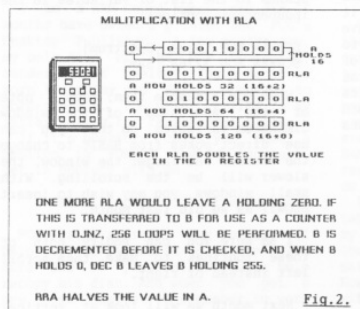
```
ONESTEP  CALL FINDWID
         INC B
         LD A,B
         RLA
         RLA
         RLA
         LD B,A
STEPLOOP PUSH BC
         CALL MOVESPR
         POP BC
         DJNZ STEPLOOP
         RET
```

Last month ONESTEP was a separate subroutine moved the whole sprite one complete cell to the right. It was called 26 times from the main loop of the routine. This month, we are using the whole width of our window as a frame, so ONESTEP's job is to move the sprite right across the window.

FINDWID is called, and INC B used because we need the true width of the window in cells. We need a counter, however, not in bytes but in bits. The number of bits involved will be 8*width.

The width is transferred to A, and RLA repeated three times multiplies it by 8. Fig.2 shows how this happens. We are quite safe in using RLA because the greatest possible number of bytes across a window is 32. RLA three times will produce zero when A holds 32, but used as a counter in B, zero will result in 256 loops. If the carry flag were set at the start of the operation it would corrupt the result, but the

MULTIPLICATION WITH RLA

```
0 ← 0 0 0 1 0 0 0 0   A HOLDS 16
0 0 0 1 0 0 0 0 0 RLA
A NOW HOLDS 32 (16+2)
0 0 1 0 0 0 0 0 0 RLA
A NOW HOLDS 64 (16+4)
0 1 0 0 0 0 0 0 0 RLA
A NOW HOLDS 128 (16+8)
```

EACH RLA DOUBLES THE VALUE IN THE A REGISTER

ONE MORE RLA WOULD LEAVE A HOLDING ZERO. IF THIS IS TRANSFERRED TO B FOR USE AS A COUNTER WITH DJNZ, 256 LOOPS WILL BE PERFORMED. B IS DECREMENTED BEFORE IT IS CHECKED, AND WHEN B HOLDS 0, DEC B LEAVES B HOLDING 255.

RRA HALVES THE VALUE IN A.   Fig.2.

carry flag is reset in FINDWID by SUB B, because LEFT will always be lower than RIGHT.

The rest of ONESTEP is exactly as it was last month, but the whole routine is now part of SCROLL's main loop.

```
MOVESPR  PUSH HL
         CALL FINDEP
SPRLOOP  PUSH BC
         PUSH HL
         CALL ROWRT
         POP HL
         CALL NXDOWN
         POP BC
         DJNZ SPRLOOP
         POP HL
         RET
```

The only change in MOVESPR is that FINDEP is called to find the number of times it is to repeat.

```
ROWRT    PUSH HL
         LD B,8
PIXLOOP  PUSH HL
         PUSH BC
         CALL FINDWID
FINDEND  INC HL
         DJNZ FINDEND
STARTMVE CALL FINDWID
```

The change at the beginning of ROWRT is similar. FINDWID finds the number of repeats - 1 less than the full width of the window because we are already at the first byte and adding (width-1) will move to the last. Between DJNZ FINDEND and STARTMVE we have removed the section which preserved the last pixel status in each row. In order to make the tractor disappear we shall always reset the first pixel. At FINDEND, INC HL is used in place of CALL NXCELL. FINDWID is called again to give the number of bytes to rotate - 1 less than the full width of the window because the first byte is treated separately after the loops.

```
MVELOOP  PUSH HL
         RR (HL)
         DEC HL
         LD A,(HL)
```

```
        EX DE,HL                    DONEIT  EX DE,HL
        POP HL                              DJNZ MVELOOP
        BIT 0,A                             RR (HL)
        JR Z,ITS0                           EX AF,AF'
        SET 7,(HL)                          AND A
        JR DONEIT                           JR Z,NOUGHT
ITS0    RES 7,(HL)                          SET 7,(HL)
DONEIT  EX DE,HL                            JR LINEDONE
        DJNZ MVELOOP                NOUGHT  RES 7,(HL)
                                   LINEDONE POP BC
                                            POP HL
                                            INC H
                                            DJNZ PIXLOOP
                                            POP HL
                                            RET
```

This section is exactly the same as last month's except that DEC HL is used instead of CALL LASTCELL.

```
        RR (HL)
        RES 7,(HL)
        POP BC
        POP HL
        INC H
        DJNZ PIXLOOP
        POP HL
        RET
```

The last byte in the line is rotated and its bit 7 reset. The section which retrieved the last bit's status and copied it to the first has been removed.

This completes SCROLL. If saved as a library routine it requires FINDEP, FINDWID and NXDOWN to be saved with it. It can be used to scroll any size or position of window by changing the values in TOP, BOTTOM, LEFT and RIGHT.

It is easily modified to produce another routine, ROLL, which will make our tractor reappear on the left as it disappears from the right. All we have to do is change the label SCROLL to ROLL, and change the last part of ROWRT to reinstate the instructions which preserved the last bit status and used it for the first bit of each line. Below are the two sections involved.

```
FINDEND   INC HL
          DJNZ FINDEND
          BIT 0,(HL)
          JR Z,BIT0
          LD A,1
          EX AF,AF'
          JR STARTMVE
BIT0      XOR A
          EX AF,AF'
STARTMVE  CALL FINDWID
```

SCROLL and ROLL can be used with any screens. You can even assemble them as separate programs, giving them their own ORG address and calling them, after loading any SCREEN$, to clear or rotate a window, with a few minor modifications.

If they are used independently Sam users must remember to insert CALL SCREENIN and CALL SCREENOUT. CALL SCREENIN will be at the labels SCROLL and ROLL, so that LD HL,SCREEN is the second instruction. CALL SCREENOUT will go between DJNZ STEPLOOP and RET. This is not necessary when they are used as subroutines with a program like the tractor program which pages the screen itself.

Sam and Spectrum users must also add SCREEN to the list of variables in the independent routines:

```
SCREEN EQU 16384 (Spectrum)
SCREEN EQU 32768 (Sam)
```

When you assemble them, keep a note of the addresses of the window dimension variables, so that you can use direct pokes from BASIC to change the window. The larger the window, the slower will be the scrolling. With small windows you may wish to insert SLOWIT as we did last month.

I am sure you can modify both of these routines to make them scroll left instead of right.

Next month we will look at vertical scrolling.

Dear Editor,
Just a few lines to say keep up the good work. I wish I could find more time to contribute and use your advice columns. Meanwhile could you recommend anyone that can repair PLUS D interfaces?

Also, has anyone out there managed to get Megabasic, Betabasic and Collossus 4 Chess working with a +2 and PLUS D setup?

Yours sincerely, David Burk.

Try Bruce Gordon on 0792-797770, he may be able to help on the PLUS D repair front and he is also doing SAM repairs if anyone is interested.

Betabasic is not a problem on the PLUS D but if you have an old version you may need to upgrade. As for the other two - I look to help from our readers - anyone with an answer?

Oh, in answer to your other question, your membership expires at the end of August 1993 - does that sound right? Ed.

Dear Editor,
I am a SAM Coupé owner, and for 18 months have owned a printer and PCG's Desktop Publisher. A few months ago, my only disc of the DTP corrupted, rendering it useless. For a month or more I used my own BASIC program to word-process. PCG never replied to my letter asking for replacement files.

If it's possible, could you make a mention in FORMAT, asking people to get in contact with me.

Yours sincerely, Martin Quested.

PCG strike again... Any offers of help will be passed on to Martin - I'm sure one of you out there can help recopy his disc. And when you get a new disc Martin - remember to work from a back-up copy not the main disc, keep that in a safe place. Ed.

Dear Editor,
Can you please confirm that my subscription runs from April 1992 to March 1993? I write this because next to my membership number you print the digits 0293. Shouldn't that be 0393?

By comparing back issues of FORMAT (Vol 2) with the current issue (Vol 5 No 11) I became a bit frustrated. In the current issue there are very few things about Spectrum or the PLUS D and a lot about SAM. Also, the article about adventures looks very similar to older ones and the article about Epson codes is useless since you don't provide the decimal values of the codes. Should Spectrum owners order all the back issues and then stop subscribing to FORMAT?

But enough of complaints. In Vol 2 Numbers 3 & 4 I found the codes for controlling the PLUS D from machine code. That was the main reason I subscribed to FORMAT. So thanks a lot. And your overseas exchange rate is a brilliant idea and should be adopted by other magazines as well.

Yours sincerely, Patrick Davy.

Well, back in the days of Vol 2 there was no SAM to write about, but our average issue size was around 28 to 32 pages. Now we normally manage 40 pages and pack even more onto each page. It is interesting to note that over any three or four issues we are averaging a fairly even split between SAM and Spectrum so there is a very real reason for Spectrum users to go on subscribing.

There was not room to give the full table of control codes in the article by Alan Cox but it would be simple to look up the decimal (or hex for that matter) code from what is given. However that was not the point, without many more details you wouldn't be able to use the codes anyway. You need the printer's manual to

---

understand how to use a code. The article was to give an overview - to help those looking for a printer in assessing the sales blurb.

And the expiry date is that of your membership - which expires at the end of February 1993. But by that time we have printed the labels for the March issue so that is the last issue you receive, except it wont be - because you will want to keep renewing wont you. Remember the first five digits on the label are your membership number which is then followed by the four digit expiry field and then our internal sorting code. **Ed.**

Dear Editor,

I have picked up composite colour monitor and would like to use it with my issue 3 rubber keyboard Spectrum. Can any one give me any info on taking composite signal from my Spectrum. I have a PLUS D disc interface connected to my Spectrum any ideas would be welcome.

Yours sincerely, Peter Baillie.

The easy way to get the composite signal is to take it from the input to the TV modulator. But get a copy of "The Spectrum Hardware Manual" by Adrian Dickens, it is out of print but your library can get it for you. This will help you further. Ed.

Dear Editor,

I thank you so very much for keeping us up to date with the goings on regarding SAMCO. All of us at the Wirral Independent Sam Coupé User Group hope and pray that the rescue package you are working on with West Coast Computers will come to fruition very soon. I hope that this Cheque which is enclosed from us in the Group (it is a joint collection) will help. if every SAM User donated just £10 then there would be enough funds available.

I also hope that there will be some reward in the way of discounts etc for the Users of our group in the future. Some of us who are unemployed really cannot afford to part with the cash but have managed to give it for this good cause.

If you wish, you may publish this letter after having condensed the important parts of it, in FORMAT to help fill a corner. Anyone wishing to contact our user group can ring me on 051-643-8878.

Yours sincerely, Charles Williams.

Thanks Charles (and all your members too) Every little helps and money is still needed. You are right, if everyone gave even £5 there would be enough to rescue SAM and finance some improvements.

And I'm sure West Coast will find some way of rewarding your groups generosity once everything is sorted out. Ed.

Dear Editor,

As you can see from my headed paper, I took note of the review of SAMPRINT in the July edition of FORMAT and bought myself a copy. I like it very much and have found it of use in preparing bookplates, apart from the advertised applications.

However, I would like to extend its library of graphics. Carol Brooksbank's program in Short Spot of the same month lead me to wonder - would it be possible to do the reverse i.e. take files such as "Clipart" from Outlet and insert them into Samprint's "custom" libraries?

I'm sure such a program in FORMAT would be of interest to others.

Yours sincerely, Ron Fox.

It is much easier to extract something than to put it in. Yes it may be possible to do what you want but you would need to work out the detailed file format for SAMPRINT. Anyone who manages that will, I'm sure let us know. Ed.

Dear Editor,

I use a 256K Sam Coupé with one internal disc drive and I have a few questions that I think other FORMAT readers might like to have answered.

What are all the laws regarding backups of Spectrum software to disc and do you need to own a WORKING Spectrum to use the ROM on SAM?

I have a few old ZX Computing mags with a lot of BASIC program listings in them. I have typed in a program called "BLOCKBUSTER" from the Sept'85 issue on my SAM. Since ZXC is no longer with us, who now owns the copyright on this and other programs listed in the mag? The SAM version I have written uses the advanced graphic capabilities of the machine and I would like to know if I could sell the program!

I enclose a listing, and a copy on cassette, of a program I've written for a friend who's a bit of a golf fanatic. it is called golf card and it was written on SAM using SPECMAKER and BETABASIC (because it's easier to type the keywords).

Yours sincerely, Nigel French.

Copyright is a thorny question Nigel, but I will try to explain the situation in an article sometime soon, there certainly is not enough room here. As for the golf card program, I've sent that up to John Wase for the Short Spot. **Ed.**

Dear Editor,

I received today the latest copy of FORMAT, once again well up to its usual standard. There was however one thing that I was rather confused about, the last paragraph of your Editorial referring to the type of Cheque that you require to be submitted. It appears that you do NOT require cheques to have the 'ACCOUNT PAYEE' endorsement between the crossing lines, when the Banks are asking us to do just this to ensure cheque safety.

This could cause considerable problems for both you and your Contributors as the latest new cheque book that I have just received from Barclays has 'ACCOUNT PAYEE' already preprinted between the crossing lines and the literature that accompanied it informed me that 'all cheques for Personal Account Customers' would be so printed in future. Presumably this will also apply to the other major banks in the near future.

It would be helpful if you could clear up this point in the next issue as your Editorial was rather ambiguous in that the first part you say that you are UNABLE to accept such cheques, and in the second sentence you say that 'any cheque bearing these works is NOT liable to be rejected'; and then you say that you only require cheques to be crossed - somewhat difficult for BARCLAYS customers at least!

Yours sincerely, John Redfern.

OK, so I didn't proof read last months editorial quite well enough. I should have said, as so many of you have pointed out, that cheques endorsed "A/C PAYEE ONLY" is liable to be rejected (the not should not be there).

Now I have had quite a few phone calls about this. The reason for my request was because our building society has warned us that new rules imposed by the banking world mean they cannot process cheques bearing that endorsement. If you think about it, the endorsement tells the bank that that cheque must only be paid into the bank account of the person named on it. Not his building society account with the bank, not his brothers, in fact only into an account that bears the exact name that is written on the cheque. We get cheques made out to FORMAT, FORMAT Publishing (sometimes with a LTD added), INDUG and several other variations. We can't be expected to open an account in each possible name, and given the appalling service banks give these days we prefer to use a building society account (cos they pay us to look after our money.

If you are a Barclays customer all you have to do is ask for a normal cheque book - we wont be the only ones you will need it for.

Oh, just before I go, it looks like from January the rules will not allow you to make ANY alteration to a cheque - not even if you initial the alteration. Strikes me the banks are getting a little to big for their boots. **Ed.**

Dear Editor,

I have been reading the 'I remember' articles with great interest. I used to help out with the ZX80/ZX81 User

group. Better known as the publishers 'Interface' or Tim Hartnell. Starting with the second ZX Microfair I did about a score of shows on the trot, and am still amazed at seeing the home computer market launch itself from the grass roots.

My clearest memory of these times is really nothing to do with computing, probably sticking out because of its bizarre nature. That is Toni Baker, nee Tony Baker, last known as Gillian Michelle 'Coming out' at a microfair, complete with handbag and high heels. A memory perhaps not for public consumption. [Oh yes it is. Ed] With regard to the real business at the first home computer show at Earls Court (April 82) I remember not being able to take money fast enough. There were three or four on a small stall and we sold almost everything on the first day, (Friday). I also remember probably in '82 as well and I think in Birmingham someone being stunned at the progress of computing. It was the way the ZX81(1K) so totally outclassed the computer he had bought four years before - A Sinclair Mk14! Then again .... Enough of this I am running the risk of getting maudlin, having been out of touch for some years now and suddenly feeling somewhat older than I was a few minutes ago.

Yours sincerely, Ian Beardsmore.

Dear Editor,

As with many other people I was dismayed by the demise of Samco but pleased to hear that something may rise from the ashes. I must admit that I wondered if they were running into problems when the mouse and game I'd ordered did not arrive, despite several phone calls, even after 4 months.

However, I must say that a copy of 'The Secretary' and 'Little Genius' arrived just before the collapse despite being ordered after the mouse. I must say that overall I've been very happy with the computer and the service and friendly attitude of the company. My own story of the company concerns the day I turned up at one of the 'fairs' at Leeds with little cash and no cheque book. Wanting a printer

interface and lead I was given the "gear" and asked to forward a cheque to cover the amount when I got home. People like that deserve to be in jobs and I hope that they soon will be.

Please find enclosed a cheque to add to the fighting fund. I hope it succeeds.

Yours sincerely, Colin Smith.

Dear Editor,

For some strange reason I thought I would send you the attached photocopies of an article from The Model Engineer and Electrician of 1906. It is, I hasten to add, a little before my time and maybe even yours!

It has absolutely nothing to do with computers but if you read the parts about the petrol motor skate it just has a very familiar ring to it. Shades of an early relation to Sir Clive? @ Can you see yourself off to post the October issue of FORMAT with 1 H.P. strapped to each foot. --- Having the benefit of steering (albeit by moving your heels) you gently accelerate (without the problem of gears) up to 38 miles per hour. With the accumulators strapped to your waist (I bet they weren't Duracell), happy that you have a range of 50 miles. As you pass the post office it starts to dawn on you the fatal flaws in your purchase. No clutch and no brakes and a waist full of battery acid!!!

Sorry to ramble on. Please feel free(and I now you will) to bin this rubbish. I just needed to get it out of the system.

Thank you for FORMAT. Matthew my son and I fight to be the first to read it each month and as neither of us can remember on which day of the month FORMAT is delivered (the first week or so of each month tend to get tactical with the postman). Luckily he has just gone back to the sixth form and has to leave before the post.

I enjoyed your nostalgia issue (must be getting old) which brought back hours of fun?, typing in those terrible listings from magazines and more hours trying to make them work. I obviously enjoyed them so much that I bought a book of them called Sixty Programs for the Sinclair ZX Spectrum and one listing in there became a

legend in the family. it was called Scuba and I must have tried about six times to get the thing to work but never did. I have a faint recollection of a diver and a shark making a brief appearance on the screen but I am not sure to this day if after all that effort if it was just my mind playing tricks. Does any one out there have a working copy?

I felt very sad at the demise of Samco, they are a smashing group of people who I have always found very helpful. With the infectious enthusiasm of Alan Miles (I don't think I have ever met him without buying something) I could see the receiver ending up buying a SAM.

Yours Sincerely, Jim Collins.

Any resemblence between the above picture and the editor is purely coincidental as he clearly states he was not around in 1906. Jenny.

Letters may be shortened or edited to fit on these pages.

This is YOUR letters page so it is up to you, our readers, to fill it. Keep letters as short as you can so we can fit in as many as possible.