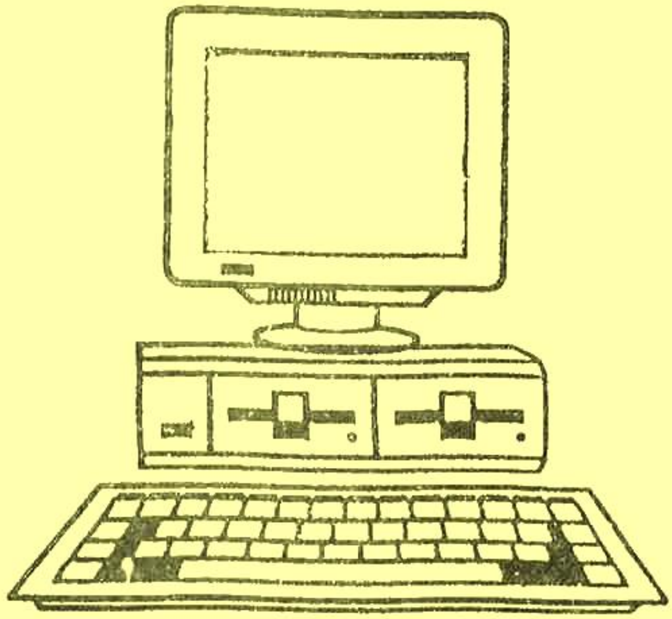
Memopad



Volume 3
Issue 9

Volume Three **MEMOPAD** Number 9

Editorial

Hello Readers

I am sure you are wondering what is going on, I think you will find the letter from Keith Hook on page one will answer all your questions.

Orion have been pulling out all the stops to offer members some new software and over the coming months we will be seeing many new additions to the software price list, in fact, this edition of the magazine contains many new releases.

We would like to apologise to those of you still awaiting their copy of "The Source". We felt under the circumstances that it was vital that we printed this edition of the Memopad before starting the second print of the book, but I can assure all of you who are still waiting that your name is on file and that you will receive your copy of the book shortly.

It is a very hectic time at present but we are endeavouring to ensure that our members are looked after and we are doing our upmost to keep things running as smoothly as possible.

As you will see from the content of the magazine I have had a great response from my plea to members for contributions to the mag and the general concensus of opinion is that the programs and articles are of a particularly high standard, I would like to take this opportunity to thank all of those who have sent me material and I hope that you will continue to do so.

Finally for those of you who are eagerly awaiting further instalments of "Football Pools Predictor' do not worry I intend to publish all the listings before the football season starts and if any of you win as a result of utilising the program Dave and I will expect our cut!

Sue

Number 9 MEMOPAD Volume Three

CONTENTS

What's In It For You!

NOTICE TO ALL MEMBERS

A RENEWAL SLIP WILL BE ENCLOSED WITH THE MAGAZINE OF ANYONE WHO IS DUE TO RENEW THEIR SUBSCRIPTION. PLEASE FILL THIS IN AND RETURN IT TO US AS SOON AS POSSIBLE SO THAT YOU DO NOT MISS AN ISSUE OF THE MAGAZINE.

Letter to all readers from Keith Hook

Well what's it all about ?

Many of you, who have called or spoken to me know that I have been very unhappy with the direction Syntaxsoft has been moving. When I decided to amalgamate my companies I did not forsee the pit-falls and dangers.

The Black Beauty has been my way of life right from it's inception and still remains my first love. My undying loyalty is still to this machine and to Geoff and Marlyn who had the desire and belief to re-incarnate the company from ashes.

Because my co-directors had different ideas from me I found that I was pushed into projects and avenues I did not even wish to pursue.

Obviously, a situation like this cannot be allowed to continue and when the magazine, repairs and technical services started to suffer, it was the final straw.

In order for me to regain control of the MTX side of the business and jettison my directors I was left with the simple choice of closing Syntaxsoft. Other events, which I cannot discuss at the moment, also lead to this decision.

Orion Software is the new company and we are operating as we did in the 'good old days'. MTX is first and I am sure you will see the difference within the next few months and I can promise you that Black Beauty is my number one concern and so is Memopad and all the users.

The transition, due to events I cannot yet disclose, is not easy and we are working every hour of the day to get things on the right track. The "Source" second edition will be out within the near future so please bear with us. Give us your support and we will support you - THAT'S A PROMISE.

HIGH SCORES

KNUCKLES	1,147,360	T. Erikson
CHAMBERIODS	Comp. 4 mins	T. Erikson
MISSION ALPHATRON	175,340	Matthew Moss
TAPEWORM	175,980	Richard Franks
TOADO	356,414	John Quinn
POT HOLE PETE	106,630	Richard Franks
MAXIMA	1,479,710	S. Olander
STAR COMMAND	140,430	Ian Nichols
OBLOIDS	62,400	M. Hurley
PHAID	26,000	Sally Street
KILOPEDE	82,253	Richard Nash
3D TACHYON FIGHTER	12,500	C. Walker
CONTINENTAL RAIDERS	106,240	Sean Haverty
BLOBBO	148,283	E. Mahon
QUANTUM	14	Ian Cartwright
QOGO 2	205,000	R. Siddall
MINEFIELD	2,100	C. Walker
FLUMMOX	251,510	C. Walker
TURBO	18,610	Michael Hunt
FATHOMS DEEP	3,450	Matthew Moss
AGROVATOR	675,000	P. Howard
FIREHOUSE FREDDIE	29,620	T. Erikson
QOGO	43,960	T. Erikson
ARCADIANS	25,900	Adrian Johnson
MISSILE COMMAND	27,580	Adrian Johnson
LITTLE DEVILS	34,320	Leslie Banks
FELIX IN THE FACTORY	14,740	Peter Crighton
HUNCHY	8,457	John Quin
SON OF PETE	17,233	T. Erikson
HAWKWARS	25,800	Gordon Hurd
ESCAPE FROM ZARCOS	76 Items	G. Hill
SALTY SAM	40,642	Andrew Johnson
MISSION OMEGA	10,850	A. Knott & S. Paine
ICEBURG	17,431	Alan Dobson
SNOWBALL	1,000	Victor Stepney
EMERALD 1SLE	768/1000	Victor Stepney
SUPERBIKE	23.9kms	A. Clark
ROLLA BEARING	27,000	Victor Stepney
DR. FRANKIE	65,435	J. Graham
TARGET ZONE	17,470	D.J. Chamberlain
MINER DICK	22,520	R. Siddall
JUMPING JACK	26,120	A. Miller
SURFACE SCANNER	72,060	T. Erikson
CAVES OF ORB	496/500	V. Stepney
SEPULCRI SCALERATI	8,000	Andrew Miller
SMG	105,400	Clare Townsend
RETURN TO EDEN	1,000	Andy Crick
QUAZZIA	26,660	Andrew Miller
OBLITERATION ZONE	32,670	Alan Dobson
ASTORMILLION	142,342	D.J. Chamberlain
CRYSTAL	32,425	Gordon Hurd (COMP)
DRIVE THE CEE?5	12,907	V. Stepney
HIGHWAY ENCOUNTER	123,120	
KARATE KING	4,570	G. Hill
DOWNSTREAM DANGER	8,976	G. Hill
DOODLEBUGS	4,340	A. Miller
THE WALL	53,500	A. Miller
COMBAT	47,690	A. Miller

Volume Three **MEMOPAD** Number 9

2 'C' Compilers

Since Hisoft's 'C++' and Manx Software Systems' 'Aztec C' Prime' share a high degree of mutual compatibility and are both at the lower end of the price range for 'C' compilers it seems reasonable to talk about both in the same article.

Both are available from Hisoft's retail outlet - 'The Software Toolshop';

The Old School,
Greenfield,Bedford MK45 5DE.

Incidentally,if you order from Hisoft,PLEASE take care to specify the disk format of your system MOST CAREFULLY!I say this with great feeling as Dave Nutkins of Hisoft and I were involved in a six-week long (and very expensive) correspondance over the fact that my system uses 'C:7' as the 'CONFIG' parameter and the new series-2 machines apparently use 'D:3'.BE WARNED!!

C++ costs £39.95 and is made and marketed by Hisoft.Aztec C' Prime is from Manx Software Systems and is retailed by Hisoft at £79.95.

C++ comes with a fully-interactive version of the well-known Hisoft editor ED.80,which returns you to the source-code at the point of the current mistake.C' can accept source-code written with any editor but,like most professional compilers,does not come with an editor.

Both compilers stick very closely to the language as defined by Kernighan and Ritchie,which has become a virtual standard for the language.Both make valiant attempts to make CP/M-80 behave like Unix.The Aztec C compiler is slightly more successful at this,but with a very considerable overhead in terms of program size.C++ is definitely faster - about as fast as Turbo-Pascal and a bit slower than Hisoft's Pascal-80 - and within its limitations produces more compact source code.

The main limitations of C++ are:

floating-point not supported;
no long integers (the word 'long' is accepted but ignored);
limited preprocessor macro expansion;
bit-fields not supported;
comma operator not supported;
some limitations on the format characters accepted by printf and scanf.

It is,however,a very good compiler which,since it supports absolutely standard C conventions,would be an ideal choice for anyone new to the language and can be used for quite serious program development.The manual is very thorough - although MINUTELY printed.

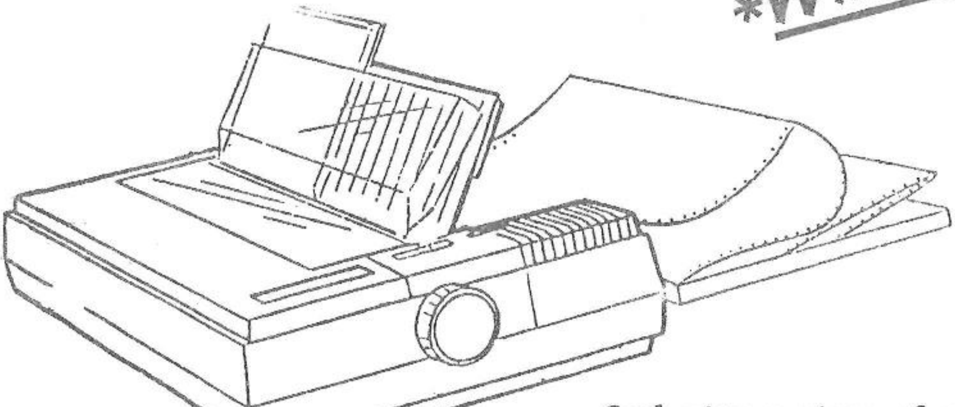
C' Prime for twice as much money gives you the FULL 'K & R' specification of the language,except that 'short' (8-bit) integers are in practice 16-bit.Floats doubles and longs are all supported,and backed up by full 32-bit arithmetic.A consquence of this is a VERY large run-time overhead - the minimum size of a program doing anything useful is likely to be around 16k to 20k.An extensive series of chain options are provided however,and these enable you to pass various kinds of data between the chained programs.Two supplementary files (which,I'm afraid,have to be bought separately as part of an upgrade package) enable you to create and load overlays.A set of support programs is provided including the assembler and linker and a 'librarian' program with which you can create your own *.LIB files or list or change the contents of existing ones.The documentation is thorough and complete,illustrated by a number of short demo programs.

Dr. B.L. Houghton.

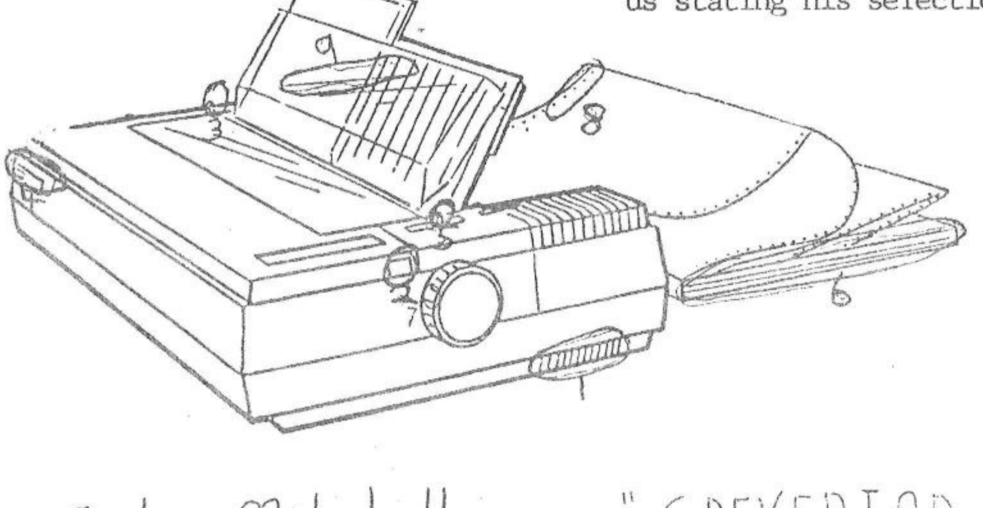
For The Under Tens

Spot The Differences

Winner



Carl wins a piece of software of his own choice and we would be grateful if he could contact us stating his selection.



Name....Carl....Mitchell.. Address..."GREYFRIAR"....
HORNSEA...RD....SKIPSEA....N..HUMBERSIDE..... M/PAD No C787
YO25 8ST

SOUND ROUTINES

Introduction

In a recent PCWeekly survey, 9 out of 10 people expect their computer to be able to make reasonable sound, with two-thirds of these expecting at least three channel sound.

The two main programmable sound generator (PSG) integrated chips (ie sound chips) used in todays modern home computers are General Instruments AY-8910 and the Texas Instruments SN 76489A. The AY-8910 is the most commonly used PSG as it has stereo output over 8 octaves and is used in the following micros: MSX, Amstrad CPC, Einstein range, Spectrum 128 and plus 2 and in the Atari ST range. The SN 76489A is less common but is used on two of the more sophisticated micros, the powerful 6502 cpu BBC micro and on the Z80 cpu Memotech MTX series. The SN 76489A is less powerful as only 4 octaves of mono sound can be generated.

This article is written for MTX series 1 & 2 micros. The PSG is a sound microprocessor which can produce three seperate voices and one noise channel, this allows harmonies to be created. As the MTX series has no onboard speaker, the sound is directed through the TV/monitor speaker or through the standard HiFi socket at the rear of the MTX. The latter has the advantage of giving sound output of high quality and is very handy for recording any masterpieces composed.

CPU - PSG Communication

To write data to the PSG, the Z80 sends valid data via 8 data lines D0-D7 (in parallel) to output port 6 on the Z80. The data waits here until a dummy read to port 3, sends this data to the PSG, see listing 1. This is analogous to waiting for a train on the platform. You have to wait there until the correct train arrives and then you leave the platform and go onto the train.

```
cputopsg:  OUT (6),A       ;load register A with data and
                           ;store at port 6
           IN A,(3)        ;This dummy read sends the data
                           ;from the Z80 to the PSG.
           RET             ;Return to calling routine.
```

There is one point to beware of and that is that 32 clock cycles or T-states must elapse before another dummy read can be performed. More information on this can be found on page 243 of the MTX manual, Technical section.

Volume Control

Three tone generators, 0-2, are available in MTX basic. In order to create music we must specify the music using the following MTX basic comand:

```
SOUND   c,f,v
```

```
where c = tone generator or sound channel, range 0-2
      f = frequency of the note, range 10-1020,$*
      v = volume of the note, range 0 (min) to 15 (max)
```

$*, note that this is a pseudo frequency range, the actual frequency range is 12500 Hz to 122 Hz respectively. The actual frequency is calculated from equation (1) section 4.0. A list of pseudo frequencies and their equivalent actual frequencies are given on page 185 to 187 of the MTX manual.

Unfortunately the MTX machine code programmer cannot use this format as the PSG is configured differently and the extra code required to mimic this command is more complex and leads to longer execution times. The PSG controls the volume of the three available tone generators via three dedicated volume registers 1,3 and 5. The PSG requires only one byte of information to select the register and the volume to be outputted. The upper nibble or upper 4-bits are used define the register and the lower nibble defines the volume of the note. I emphasise at this point that the volume range used by the PSG is the reverse of basic, ie 0 (max) and 15 is minimum. The upper nibble patterns give the volume register, see table 1.

Table 1: Volume register select

Sound channel	PSG register	Upper Nibble pattern	Data to select PSG register
0	1	0 0 0 1	16
1	3	0 0 1 1	48
2	5	0 1 0 1	80
3*	7	0 1 1 1	112

*, This is the noise volume register which is setup as the tone generators.

I have arranged my assembly code to be as simple and as easily understood as possible. Listing 2, needs two inputs, ie the PSG volume 15 to 0 and the PSG register, this is selected from column 4 of table 1. These two bytes are added together and bit 7 is set and the data is sent using the technique in cputopsg.

Listing 2

```
10  GOTO 100

20 CODE

VSTART:PUSH AF              ;SAVE ANY REGISTERS CORRUPTED
       PUSH BC              ;
       LD A,(VOL)           ;GET VOL
```

```
         AND 15                    ;GET RID OF UPPER NIBBLE
         LD B,A                    ;SAVE IT
         LD A,(REG)                ;GET REG
         AND 240                   ;GET RID OF LOWER NIBBLE
         ADD A,B                   ;VOL + REG
         OR #80                    ;SET BIT 7
         CALL CPUTOPSG             ;SEND TO PSG, see listing 1
         POP BC                    ;RESTORE REGISTERS
         POP AF                    ;
         RET                       ;RETURN TO BASIC
VOL:   DS 1                        ;POKE VOLUME HERE
REG:   DS 1                        ;POKE REG HERE
```

```
100  SOUND 0,256,0   :REM ** initialise frequency as not
                            defined yet, until 4.2.  **

110  POKE   VOL,10   :REM ** volume of 10 **
120  POKE   REG,16   :REM ** select register 1 **
130  RAND USR (VSTART)
140  STOP
```

Remember to substitute the decimal equivalents of VOL & REG into lines 110 and 120.

You should be able to control the volume of a sound now. The above code and other listings will form a suite of sound utilities which I hope someone can develop into a music composer editor.

Frequency Synthesis

As already stated the MTX uses a 'pseudo' frequency range, 0 to 1024. The actual frequency can be calculated from equation (1).

Actual frequency = N / (32 * f)   ....(1)

where N = the reference clock frequency,4,000,000 Hz
and   f = the 'pseudo' frequency

For example, a frequency of 256 Hz gives a pseudo frequency, f, of:

f = 4,000,000 / (32 * actual frequency of 256) = 488

Therefore in basic the programmer would use a value of 488 to get a frequency of 256 Hz.

The PSG requires 10 bits on information to define the half period of the desired frequency. This 10 bit frequency, F0 to F9, is loaded into a ten stage tone counter which is decremented at a rate of N/16, where N is the clock speed of the Z80, ie 4,000,000. When the tone counter reaches zero, a borrow signal is produced. This borrow signal toggles the frequency, via flipping over and reloading the tone counter. Therefore the period of the desired frequency is twice the value of the period register.

The PSG has three dedicated tone generator registers 0,2 and 4. The register and frequency are sent to the PSG as two bytes:

```
          MSB                                 LSB
7  6  5  4  3  2  1  0           7  6  5  4  3  2  1  0

1  `-REG-"  F3 F2 F1 F0          0  x F9 F8 F7 F6 F5 F4
```

To use this format directly is very confusing, and the extra programming is a pain. The code used to define the frequency and its register has been simplified. I have used the same technique as in basic, ie input the frequency register, ie 0,32,64 or 96 (noise register), and then input a pseudo frequency value. Note that two bytes are required to define the frequency, ie

```
          MSB                                 LSB
7  6  5  4  3  2  1  0            7  6  5  4  3  2  1  0

x  x  x  x  x  x F9 F8           F7 F6 F5 F4 F3 F2 F1 F0
```

LISTING 3, to select a pseudo frequency of 488 and a volume of 10 setup the following:

```
10  GOTO 100
20  CODE

VSTART:    see LISTING 2
CPUTOPSG:  see LISTING 1

FSTART:    PUSH AF
           PUSH BC
           PUSH HL
           LD HL,(FREQ)
           LD A,L                ;GET F3-F0
           AND 15                ;GET RID OF THE UPPER BITS
           LD B,A                ;SAVE IT
           LD A,(REG)            ;GET THE FREQUENCY REGISTER
           AND 240               ;GET RID OF UNWANTED BITS
           OR #80                ;SET BIT 7
           ADD A,B               ;NOW IN PSG MSB FORMAT
           CALL CPUTOPSG         ;SEND IT
           LD A,L                ;GET F7-F4
           AND 240               ;GET RID OF UNWANTED BITS
           LD L,A                ;SAVE IT
           LD A,H                ;GET F9 AND F8
           AND 15                ;GET RID OF UNWANTED BITS
           LD H,A                ;SAVE IT
           LD B,4                ;SET COUNTER
FLOOP:     SRL H                 ;SHIFT H LEFT
```

```
          RR L                  ;MOVE INTO L AND MOVE LEFT
          DJNZ FLOOP            ;REPEAT UNTIL IN PSG LSB FORMAT
          LD A,L
          CALL CPUTOPSG
          POP HL
          POP BC
          POP AF
          RET
FREQ:     DS 2                  ;STORE PSEUDO FREQUENCY

100 POKE REG,16                 :REM SELECT VOL REGISTER
110 POKE VOL,10                 :REM SET VOLUME
120 RAND USR(VSTART)            :REM CALL VSTART
130 POKE REG,0                  :REM SELECT FREQUENCY REGISTER
140 POKE FREQ,232               :REM LSB OF FREQUENCY
150 POKE FREQ+1,1               :REM MSB
160 RAND USR(FSTART)            :REM CALL FSTART
170 STOP
```

The above listing demonstrates how to select frequency and volume and is equivalent to SOUND 0,488,5 in basic.

Noise Generation

Noise is a random mixture of frequencies which can be used to provide special sound effects like waves, drumbeats, etc. The noise generator consists of a noise source and an attenuator. The noise attenuator is setup as shown in section 4.0. The noise source is actually a shift register with an exclusive OR feedback network. Note that the network has provisions to protect the shift register from locked in the zero state.

Two noise configurations are possible, "periodic" and "white". "Periodic" noise as suggested by the name has a period associated with it, unlike "white" noise which is completely random. To select either noise configuration the Feedback, FB, bit must be either one or zero respectively. The FB bit is bit 2. The PSG requires one byte of information to select the register and the FB and the actual noise selected. This combined bit is sent to the PSG. The PSG format is:

7	6	5	4	3	2	1	0
1	1	1	0	x	FB	NF1	NF0

The upper nibble selects register 6, and doesn't need to be specified as this is automatically selected on calling the noise code, see later. NF1 and NF0 define the shift register and are selected from table 2.

Table 2: NF patterns and shift rates

NF1	NF0	Shift Rate
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	tone 2 output

Therefore the fixed sift rates are derived from the Z80 clock speed. The shift register will only shift at one of the 3 rates as determined by the two NF bits. Note that whenever the noise control register is changed the shift register is cleared.

In one special case though when both NF bits are set, the noise output is directed through tone generator channel 2. This will allow us to envelope and modulate noise as if it were pure sound. This is necessary to produce drum sounds like the bass drum, etc.

Listing 4, sets up white noise with a shift rate of N/512

```
10 GOTO 100

20 CODE

VSTART:  SEE LISTING 2
CPUTOPSG:  SEE LISTING 1

NSTART:  PUSH AF
         PUSH BC
         LD A,(SHRATE)           ;GET SHIFT RATE
         AND 3                   ;GET RID OF UNWANTED BITS
         LD B,A                  ;SAVE IT
         LD A,(PORW)             ;PERIODIC OR WHITE NOISE
         AND 4                   ;GET RID OF UNWATED BITS
         ADD A,B                 ;FB + NF
         OR 224                  ;SELECT REGISTER 6
         CALL CPUTOPSG           ;SEND IT
         POP BC
         POP AF
         RET
SHRATE:  DS 1
PORW:    DS 1

100 POKE REG,112                 :REM SELECT VOLUME REGISTER
110 POKE VOL,10
120 RAND USR(VSTART)
130 POKE SHRATE,0                :REM SHRATE SETUP AS N/512
140 POKE PORW,1                  :REM WHITE NOISE
150 RAND USR(NSTART)
160 STOP
```

This code simulates SOUND 3,4,5.

Sound Off

This last section shows you how to switch off all channels.

```
10 CODE

VOFF:  LD A,15                 ;VOL=OFF
       LD (VOL),A              ;SAVE IT
       LD A,16                 ;VOLUME REGISTER 1
VOFF1: LD (REG),A              ;SAVE IT
       CALL VSTART             ;UPDATE VOLUME
       ADD A,32                ;UPDATE VOLUME REGISTER
       DJNZ VOFF1
       RET                     ;RETURN TO BASIC
VSTART:SEE LISTING 2
CPUTOPSG: SEE LISTING 1

20 STOP
```

(c) A.F. Wilson 1986.

Volume Three **MEMOPAD** Number 9

FOOTBALL POOLS PREDICTOR

PART TWO

```
10 REM ************************************************************************
20 REM ******************** FOOTBALL POOLS FORECASTING ************************
30 REM ********************   PROFESSOR FRANK GEORGE   ************************
40 REM ********************       COMMODORE 64         ************************
50 REM ********************    ADAPTED FOR MEMOTECH    ************************
60 REM ********************      BY DAVE WEMYSS        ************************
70 REM ********************         JAN 1987           ************************
80 REM ********************       FORECAST.BAS         ************************
90 REM ************************************************************************
100 DISC SAVE "FORECAST.BAS"
110 CLEAR : VS 5: CLS
120 DIM RECORD$(25,11,16),F(12,2),LOA$(2,12),DIV$(25),A$(1),HOME$(11,16),AWAY$(1
1,16),FORECAST$(12,4),WEEK$(9),DATE$(9),EMP$(4)
130 LET Y=0: LET Z=0: LET EMP$="    "
210 CLS : PLOD "PROG1"
220 GOSUB 2200: IF A<128 OR A>134 THEN GOTO 220
230 GOSUB 2250
240 CLS : CSR 25,0: PRINT DIV$: PRINT : GOSUB 2350
250 GOSUB 2180: PRINT "Loading fixture number ";Y
260 DISC OPEN £1,LOA$(1),"I"
265 DISC INPUT £1,WEEK$
270 FOR X=1 TO 25
280 DISC EOF £1,330
290 LET Y=Y+1: CSR 43,7: PRINT Y
300 DISC INPUT £1,F(X,1)
310 DISC INPUT £1,F(X,2)
320 NEXT X
330 DISC CLOSE £1
340 DISC OPEN £1,LOA$(2),"I"
350 CLS : CSR 20,0: PRINT DIV$: GOSUB 2350: GOSUB 2180: CSR 20,7: PRINT "Loading
 team number ";Z: DISC INPUT £1,DATE$
355 LET Z=0
360 FOR X=1 TO 25
370 DISC EOF £1,420
380 LET Z=Z+1: CSR 40,7: PRINT Z
390 FOR I=1 TO 11
400 DISC INPUT £1,RECORD$(X,I)
410 NEXT I: NEXT X
420 DISC CLOSE £1
430 CLS : CSR 25,0: PRINT DIV$: GOSUB 2350: GOSUB 2180: PRINT "Forecasting resul
t for match number ": FOR X=1 TO Y: LET FORECAST$(X)=EMP$: NEXT X
440 FOR X=1 TO Y: CSR 56,7: PRINT X
450 IF F(X,1)=0 OR F(X,2)=0 THEN LET FORECAST$(X)="VOID": GOTO 2010
460 FOR V=1 TO 11: LET HOME$(V)="                ": LET AWAY$(V)="
  ": NEXT V
465 FOR V=1 TO 11
470 LET HOME$(V)=RECORD$(F(X,1),V): LET AWAY$(V)=RECORD$(F(X,2),V)
480 NEXT V
490 IF VAL(HOME$(3))=0 THEN LET HOME$(3)="1"
500 IF VAL(AWAY$(3))=0 THEN LET AWAY$(3)="1"
510 LET HOME2=VAL(HOME$(2))/VAL(HOME$(3))
```

```
520 LET AWAY2=VAL(AWAY$(2))/VAL(AWAY$(3))
530 IF HOME2>=AWAY2 THEN GOTO 570
540 LET G=AWAY2-HOME2
550 IF G<0.4 THEN GOTO 970
560 IF G>=0.4 THEN GOTO 1280
570 IF HOME$(10)="H" THEN GOTO 590
580 IF HOME$(10)="A" THEN GOTO 730
590 IF HOME$(11)="W" THEN GOTO 620
600 IF HOME$(11)="D" THEN GOTO 650
610 IF HOME$(11)="L" THEN GOTO 670
620 IF AWAY$(11)="W" THEN GOTO 1850
630 IF AWAY$(11)<>"W" THEN GOTO 1800
650 IF AWAY$(11)<>"L" THEN GOTO 1850
660 IF AWAY$(11)="L" THEN GOTO 1800

670 IF AWAY$(10)="A" THEN GOTO 690
680 IF AWAY$(10)="H" THEN GOTO 1850
690 IF AWAY$(11)="W" THEN GOTO 1980
700 IF AWAY$(11)="D" THEN GOTO 1890
710 IF AWAY$(11)="L" THEN GOTO 1850
730 IF HOME$(11)="W" THEN GOTO 760
740 IF HOME$(11)="D" THEN GOTO 840
750 IF HOME$(11)="L" THEN GOTO 920
760 IF AWAY$(10)="H" THEN GOTO 780
770 IF AWAY$(10)="A" THEN GOTO 810
780 IF AWAY$(11)="W" THEN GOTO 1850
800 IF AWAY$(11)<>"W" THEN GOTO 1800
810 IF AWAY$(11)="W" THEN GOTO 1800
830 IF AWAY$(11)<>"W" THEN GOTO 1890
840 IF AWAY$(10)="H" THEN GOTO 860
850 IF AWAY$(10)="A" THEN GOTO 890
860 IF AWAY$(11)="W" THEN GOTO 1890
880 IF AWAY$(11)<>"W" THEN GOTO 1800
890 IF AWAY$(11)="W" THEN GOTO 1800
910 IF AWAY$(11)<>"W" THEN GOTO 1890
920 IF AWAY$(10)="A" THEN GOTO 930
925 IF AWAY$(10)="H" THEN GOTO 1800
930 IF AWAY$(11)="W" THEN GOTO 1850
940 IF AWAY$(11)="D" THEN GOTO 1890
960 IF AWAY$(11)="L" THEN GOTO 1930
970 IF HOME$(10)="H" THEN GOTO 990
980 IF HOME$(10)="A" THEN GOTO 1040
990 IF HOME$(10)="W" THEN GOTO 1850
1000 IF HOME$(11)="L" THEN GOTO 1800
1010 IF AWAY$(11)="W" THEN GOTO 1850
1030 IF AWAY$(11)<>"W" THEN GOTO 1890
1040 IF HOME$(11)="W" THEN GOTO 1070
1050 IF HOME$(11)="D" THEN GOTO 1150
1060 IF HOME$(11)="L" THEN GOTO 1190
1070 IF AWAY$(10)="H" THEN GOTO 1090
1080 IF AWAY$(10)="A" THEN GOTO 1120
1090 IF AWAY$(11)="W" THEN GOTO 1890
1110 IF AWAY$(11)<>"W" THEN GOTO 1800
1120 IF AWAY$(11)="W" THEN GOTO 1850
1140 IF AWAY$(11)<>"W" THEN GOTO 1800
1150 IF AWAY$(10)="A" THEN GOTO 1980
1160 IF AWAY$(11)="W" THEN GOTO 1850
1180 IF AWAY$(11)<>"W" THEN GOTO 1800
1190 IF AWAY$(10)="H" THEN GOTO 1210
1200 IF AWAY$(10)="A" THEN GOTO 1250 ↗
```

```
1210 IF AWAY$(11)="W" THEN GOTO 1800
1220 IF AWAY$(11)="D" THEN GOTO 1890
1240 IF AWAY$(11)="L" THEN GOTO 1930
1250 IF AWAY$(11)<>"L" THEN GOTO 1980
1270 IF AWAY$(11)="L" THEN GOTO 1890
1280 IF HOME$(10)="H" THEN GOTO 1300
1290 IF HOME$(10)="A" THEN GOTO 1420
1300 IF HOME$(11)="W" THEN GOTO 1330
1310 IF HOME$(11)="D" THEN GOTO 1360
1320 IF HOME$(11)="L" THEN GOTO 1390
1330 IF AWAY$(11)<>"L" THEN GOTO 1850
1350 IF AWAY$(11)="L" THEN GOTO 1890
1360 IF AWAY$(11)<>"D" THEN GOTO 1930
1380 IF AWAY$(11)="D" THEN GOTO 1890
1390 IF AWAY$(11)<>"L" THEN GOTO 1980
1410 IF AWAY$(11)="L" THEN GOTO 1930
1420 IF HOME$(11)="W" THEN GOTO 1450
1430 IF HOME$(11)="D" THEN GOTO 1540
1440 IF HOME$(11)="L" THEN GOTO 1640
1450 IF AWAY$(10)="H" THEN GOTO 1470

1460 IF AWAY$(10)="A" THEN GOTO 1500
1470 IF AWAY$(11)<>"D" THEN GOTO 1890
1490 IF AWAY$(11)="D" THEN GOTO 1980
1500 IF AWAY$(11)="W" THEN GOTO 1850
1510 IF AWAY$(11)="D" THEN GOTO 1930
1530 IF AWAY$(11)="L" THEN GOTO 1800
1540 IF AWAY$(10)="H" THEN GOTO 1560
1550 IF AWAY$(10)="A" THEN GOTO 1600
1560 IF AWAY$(11)="W" THEN GOTO 1930
1570 IF AWAY$(11)="D" THEN GOTO 1890
1590 IF AWAY$(11)="L" THEN GOTO 1850
1600 IF AWAY$(11)="W" THEN GOTO 1930
1610 IF AWAY$(11)="D" THEN GOTO 1850
1630 IF AWAY$(11)="L" THEN GOTO 1890
1640 IF AWAY$(10)="H" THEN GOTO 1660
1650 IF AWAY$(10)="A" THEN GOTO 1690
1660 IF AWAY$(11)<>"D" THEN GOTO 1980
1680 IF AWAY$(11)="D" THEN GOTO 1890
1690 IF AWAY$(11)="W" THEN GOTO 1980
1700 IF AWAY$(11)="D" THEN GOTO 1850
1720 IF AWAY$(11)="L" THEN GOTO 1890
1800 GOSUB 2500
1810 IF P>Q THEN LET FORECAST$(X)="H"
1820 IF Q-P<0.4 THEN LET FORECAST$(X)="H"
1830 IF Q-P>=0.4 THEN LET FORECAST$(X)="H/D"
1840 GOTO 2010
1850 GOSUB 2500
1860 IF P>Q THEN LET FORECAST$(X)="H"
1870 IF Q-P<0.4 THEN LET FORECAST$(X)="H/D"
1880 IF Q-P>=0.4 THEN LET FORECAST$(X)="D"
1885 GOTO 2010
1890 GOSUB 2500
1900 IF P-Q>=0.4 THEN LET FORECAST$(X)="H/D"
1910 IF Q-P>=0.4 THEM LET FORECAST$(X)="D/A"
1920 IF P-Q<0.4 THEN LET FORECAST$(X)="D"
1924 IF Q-P<0.4 THEN LET FORECASI$(X)="D"
1925 GOTO 1210
1930 GOSUB 2500
1940 IF P>=Q THEN LET FORECAST$(X)="D"
```

```
1950 IF Q-P<0.4 THEN LET FORECAST$(X)="D/A"
1960 IF Q-P>=0.4 THEN LET FORECAST$(X)="A"
1970 GOTO 2010
1980 GOSUB 2500
1990 IF P>=Q THEN LET FORECAST$(X)="D/A"
2000 IF P<Q THEN LET FORECAST$(X)="A"
2010 GOSUB 2600: NEXT X
2020 CLS : CSR 20,10: PRINT "Forecasts for ";DIV$;" now complete."
2030 CSR 20,15: PRINT "Press <P> for Printout"
2040 LET A$=INKEY$: IF A$<>"P" THEN GOTO 2030
2050 CLS : CSR 20,0: PRINT DIV$: LPRINT "                   ";DIV$: GOSUB 2350
2060 LPRINT CHR$(27);"D";CHR$(1);CHR$(17);CHR$(20);CHR$(37);CHR$(40);CHR$(0)
2070 FOR X=1 TO Y
2080 IF F(X,1)=0 OR F(X,2)=0 THEN CSR 20,X+3: PRINT FORECAST$(X): LPRINT FORECAS
T$(X): GOTO 2110
2090 CSR 20,X+3: PRINT RECORD$(F(X,1),1): CSR 37,X+3: PRINT "v": CSR 40,X+3: PRI
NT RECORD$(F(X,2),1): CSR 56,X+3: PRINT "=": CSR 58,X+3: PRINT FORECAST$(X)
2100 LPRINT RECORD$(F(X,1),1),"v",RECORD$(F(X,2),1),"=",FORECAST$(X)
2110 NEXT X
2120 LPRINT : GOSUB 2400: LPRINT : LPRINT
2130 CSR 20,20: PRINT "Forecasts for ";DIV$;" now printed": PAUSE 3000
2140 CSR 0,20: PRINT CHR$(5): CSR 20,20: PRINT "Any more forecasts to do? (Y/N)"
: GOSUB 2450
2150 IF A$="Y" OR A$="y" THEN GOTO 130
2160 CLS : GOSUB 2180: PRINT "Returning to Main Menu"
2170 DISC LOAD "POOLS.BAS"

2180 CSR 20,5: PRINT "Please wait..........": CSR 20,7
2190 RETURN
2200 LET A$=INKEY$: LET A=ASC(A$)
2210 RETURN
2220 RETURN
2250 IF A=128 THEN LET LOA$(1)="ENGDIVS1.FIX": LET LOA$(2)="ENGDIVS1.TMS": LET D
IV$="English First Division": GOTO 2320
2260 IF A=129 THEN LET LOA$(1)="ENGDIVS2.FIX": LET LOA$(2)="ENGDIVS2.TMS": LET D
IV$="English Second Division": GOTO 2320
2270 IF A=130 THEN LET LOA$(1)="ENGDIVS3.FIX": LET LOA$(2)="ENGDIVS3.TMS": LET D
IV$="English Third Division": GOTO 2320
2280 IF A=131 THEN LET LOA$(1)="ENGDIVS4.FIX": LET LOA$(2)="ENGDIVS4.TMS": LET D
IV$="English Fourth Division": GOTO 2320
2290 IF A=132 THEN LET LOA$(1)="SCOTPREM.FIX": LET LOA$(2)="SCOTPREM.TMS": LET D
IV$="Scottish Premier Division": GOTO 2320
2300 IF A=133 THEN LET LOA$(1)="SCOTDIV1.FIX": LET LOA$(2)="SCOTDIV1.TMS": LET D
IV$="Scottish First Division": GOTO 2320
2310 IF A=134 THEN LET LOA$(1)="SCOTDIV2.FIX": LET LOA$(2)="SCOTDIV2.TMS": LET D
IV$="Scottish Second Division"
2320 RETURN
2350 FOR X=0 TO 79: PRINT "-";: NEXT X
2360 RETURN
2400 FOR X=0 TO 79: LPRINT "-";: NEXT X
2410 RETURN
2450 LET A$=INKEY$: IF A$<>"" THEN GOTO 2450
2460 LET A$=INKEY$: IF A$="" THEN GOTO 2460
2470 IF A$<>"N" AND A$<>"n" AND A$<>"Y" AND A$<>"y" THEN GOTO 2450
2480 RETURN
2500 IF VAL(HOME$(8))=0 THEN LET HOME$(8)="1"
2510 IF VAL(AWAY$(9))=0 THEN LET AWAY$(9)="1"
2520 LET P=VAL(HOME$(4))/VAL(HOME$(8))
2530 LET Q=VAL(AWAY$(6))/VAL(AWAY$(9))
2550 LET FORECAST$(X)="   "
```

```
2560 RETURN
2600 IF FORECAST$(X)<>"H/D " AND FORECAST$(X)<>"D/A " THEN GOTO 2690
2610 IF FORECAST$(X)="D/A " THEN GOTO 2660
2620 LET FORECAST$(X)="    "
2630 LET P=(VAL(HOME$(5))/VAL(HOME$(8))): LET W=(VAL(AWAY$(6))/VAL(AWAY$(9)))
2635 LET D=(VAL(AWAY$(7))/VAL(AWAY$(9))): LET G=(VAL(HOME$(4))/VAL(HOME$(8)))
2640 IF P>W AND D>G THEN LET FORECAST$(X)="D": GOTO 2650
2643 IF P<W AND D<G THEN LET FORECAST$(X)="H": GOTO 2650
2645 LET FORECAST$(X)="H/D"
2650 GOTO 2690
2660 LET FORECAST$(X)="    "
2670 LET P=(VAL(HOME$(5))/VAL(HOME$(8))): LET W=(VAL(AWAY$(6))/VAL(AWAY$(9)))
2675 LET D=(VAL(AWAY$(7))/VAL(AWAY$(9))): LET G=(VAL(HOME$(4))/VAL(HOME$(8)))
2680 IF P>W AND D>G THEN LET FORECAST$(X)="A": GOTO 2690
2683 IF P<W AND D<G THEN LET FORECAST$(X)="D": GOTO 2690
2685 LET FORECAST$(X)="D/A"
2690 RETURN
```

```
                 CHOOSE DIVISION MENU
                 ====================

English First Division..............................F1

English Second Division.............................F2

English Third Division..............................F3

English Fourth Division.............................F4

Scottish Premier Division...........................F5

Scottish First Division.............................F6

Scottish Second Division............................F7
```

Speeding It Up

The standard way to speed up a program which runs too slowly is to rewrite the thing in machine code or to insert blocks of machine code to do the slower bits.However,unless you actually enjoy using assembly-language (yes,I am told that some people do!) this is not always necessary,as programs can often be speeded up considerably - and without any loss of their portability - by careful attention to their syntax.

In the Stone-Age days of computing (back in the late '60's) a lot of time was devoted to methods of program optimisation,now often relegated to the last few lines of textbooks.It's a pity that they are so widely ignored,as a number of studies have shown that algorithms often thought to be too slow to run other than in machine-code are so because they are very inefficiently written.

In case this article causes me to be got by a hit-squad of Structured Programmers,most of these ideas are for emergency use only.I'm not suggesting that we should go back to the sort of mess found in the early TRS-80 programs,but some documentation and structuring techniques definitely carry a speed penalty.

SPEED OF ARITHMETIC OPERATIONS

The following table is now very old,and was written with mainframe systems in mind,but remains a generally reliable indication of relative speeds.

integer assignment	= 1
integer addition/subtraction	= 1.5
real assignment	= 2
real addition/subtraction	= 3
integer multiplication	= 5
integer to real conversion	= 6
real multiplication	= 8
division	= 9
integer power	= 35
real power	= 115
transcendental function	= 150+

These ratios are a good general guide but individual micros,languages and compilers vary,as you can see if you compare benchmark tests in reviews.

1.<u>Don't use non-integer values or fractional steps in loops.</u>Even in MTX basic,with no integer datatype,a loop will usually run a bit faster if it starts and finishes on an integer and has a step which is a multiple of 1.This means that,as in Pascal,you can only loop from 0.1 to 100 in steps of 0.1 by writing:

```
DO 10 I=1,1000
R = I * 0.1
..........
```

```
        10        CONTINUE
```

but even with the extra code it's usually still faster.

2.Don't multiply if you can add,or divide if you can multiply,and avoid exponentiating to small integer powers:N + N is faster than 2 * N and N * N is faster than NΔ2.0 or N ** 2.

3.Try to turn division by a constant value into multiplication by the reciprocal of that value.

4.Try to simplify arithmetic expressions as much as possible.We all hated algebraic factors at school,but 'X = A + B:Y = X * X' is up to 20 times faster than 'Y = AΔ2 + 2 * A * B + BΔ2'.All modern languages allow you to mix integer and real variables in the same statement,but in very complex expressions it may still be a little slower to do so.A trial run on your system will show whether you have anything to gain by separating them.

5.Don't put anything inside a loop which always evaluates to the same result.

&.Transcendental functions are horrendously slow:if you are,for instance,using sines & cosines of the same 30 angles repeatedly and can afford the extra variable space,then construct a Look-Up table of them at the start of the program run.

SUBROUTINES AND THEIR PARAMETERS

In interpreted Basic,there are two simple principles.GOSUBs are always faster than GOTOs,and doing either to a low line number is always much faster than going to a high one.As A.F.Wilson showed in a recent 'Memopad' article,it also saves storage space,so put frequently called routines at the START of the program and not at the end where one usually sees them.

Fortran is the only common compiled language which allows you this choice of position - it should not make any difference in this case,nor should it matter in what order you declare them in a Pascal program,but it does no harm to find out just once.

Unless you really must extract the very last millisecond of speed out of a program you should stick to the now usual principles of program structuring.Very occasionally it may be important to remember that it takes a very small time to call functions and subroutines,and also to pass parameters to and from them.Pascal VAR parameters pass more quickly than value parameters (and don't take up extra storage) and global variables are accessed more quickly still.If a loop calls a particular module many thousands of times,it may well be worth putting the entire loop into its own subprogram and writing the called procedure as a linear series of statements within the loop.

Incidentally,it is surprising how few books on Pascal or Fortran emphasise that these languages treat entire arrays as single variables - quite a few users who progress to one of these systems from

Basic waste a lot of time passing or exchanging arrays one element at a time.If you are using Fortran you can do a lot of things very rapidly with large numbers of variables by use of the COMMON and EQUIVALENCE statements or by incuding a repetition factor in DATA statements.These three terms are currently rather dirty words because of what happens if the facilities are misused,but they remain very useful emergency techniques.

If you have to store a lot of temporary data in a Pascal program,can afford the extra program and runtime space,and don't want random-access to the data then remember that dynamically-linked lists can be read much more quickly than arrays.This leads us naturally to:

INPUT AND OUTPUT

Read and Write statements are among the slowest operations in any programming language.There is not usually much you can do to reduce their number but you can put them in their most efficient form.

1.Don't put calculations and conversions into Read/Write statements.

2.Send your data to files of the appropriate type:in Pascal it is much quicker to write an integer to a variable declared as :FILE OF integer than to one declared as :FILE OF char even though the latter is allowed.In Fortran this means using free-format Read & Write statements to generate and access binary files.

3.Don't put FORMAT commands or field parameters into statements writing to files that only a computer will ever read unless these are absolutely essential to defining the file-type.

4.Disc I/O is the slowest thing that can happen inside a computer system.Try to do it only at the beginning and end of a program,transferring your datafile to or from a suitable RAM-based structure (in blocks if necessary) and letting your program operate on that.

5.Whether there is any significant speed difference between 'READ (A,B,C,D,E)' and:

```
READ (A)
.......
READ (E)
```

will depend on the language and your implementation.

ARRAYS

It always takes longer to access an array element than an unstructured variable.It's worth bearing in mind that on some systems it takes longer to read the last element of A(10,10,10) than the last element of A(1000).Mapping functions and 'unfolded' loops may then speed things up,but should only be used if there is a very considerable gain in exchange for the untidiness.

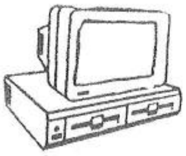
Different compilers and interpreters tend to handle arrays in slightly different ways which are not apparent to the user.If you know your system well you may be able to use this in run-time optimisation,but do not expect it to work in the same way on another machine,even if the source-code is fully portable (you may even slow it down)!

ODDS AND ENDS

Comment lines and very long variable names slow Basic up quite a bit,and multistatement lines speed it up a little.If necessary have two versions of your program;a readable one for your files and a fast one for actual use.In compiled languages this doesn't make any difference.

Study the 'options' section of your compiler manual carefully and use the ones that will give the most efficient runtime code.How much of an improvement you can make will depend on your compiler,but you can always make some.If you have written Pascal or Fortran source code carefully enough you may also have the chance to persuade someone to generate a .COM file for you using a faster compiler.

Dr. B.L. Houghton.



Volume Three **MEMOPAD** Number 9

CONNECT FOUR

This month we are adding three routines.

a] CLS
b] PRINT
c] PLOT

The CLS routine is fairly simple. Before clearing the screen it re-sets the Video Enable/disable bit so that the screen instantly blanks out. Of course, this in itself, does not clear the screen but it does save any messy bits floating around. The bit-mapped screen is cleared by clearing out the Pattern Generator and the Colour Table. The screen is then automatically cleared.

PRINT interfaces to KSUB1 by testing for FFH which means that the string to be printed has reached the end. HL must point to the string on entry to the routine.

The PLOT routine is a simple generic plotting facility that plots one point at a time. This can be upgraded to interface to general plotting routines that will plot from x,y to x1,y1 but we don't need it here. We shall deal with the plot routine again in the next edition.

Next month we shall put the board on the screen and then we will put a loop in the program so that we shall be able to print the board up on the screen and see how many mistakes we have made ... exciting isn't it ?

```
                                TITLE Connect Four Assembler version for magazines only  <c> K. Hook 1987

469C    F3                      START:  DI                                      ;DISABLE
469D    31 43B2                         LD      SP,STACK                        ;MAKE SURE STACK POINTER O.K.
46A0    CD 4171                         CALL    G2INIT                          ;INITIALISE SCREEN
46A3    CD 46A6                         CALL    CLS                             ;MAKE SURE ITS CLEARED


                                ;This routine cls the graphic screen after first blanking out the screen
                                ;buffer COL should be filled with the correct colour you wish the screen to
                                ;be cleared to.... in our case black
46A6                            CLS:
46A6    3E 80                           LD      A,80H                   ;THIS BLANKS SCREEN BY
46A8    D3 02                           OUT     (02),A                  ;RESETTING VDP REGISTER 1
46AA    3E 81                           LD      A,81H                   ;BIT 7 + 1
```

```
46AC    DD 36 00 00              LD      (IX+00H),0          ;RESET CSR POSITION
46B0    DD 36 01 01              LD      (IX+01H),1          ;Y POS
46B4    21 0000                  LD      HL,0000             ;NOW WE CAN CLEAR SCREEN
                                                             ;HL = PATT GENERATOR
46B7    CD 429C                  CALL    ADDOUT              ;SEND ADDRESS
46BA    01 1800                  LD      BC,1800H            ;NO OF BYTES TO CLEAR

46BD    AF              CLS1:    XOR     A                   ;CLEAR A=0
46BE    D3 01                    OUT     (01),A
46C0    0B                       DEC     BC
46C1    79                       LD      A,C
46C2    B0                       OR      B
46C3    20 F8                    JR      NZ,CLS1             ;IF BC <>0 DO IT AGAIN
46C5    21 2000                  LD      HL,2000H            ;COLOUR TABLE ADDRESS
46C8    CD 429C                  CALL    ADDOUT
46CB    01 1800                  LD      BC,1800H

46CE                    CLS2:
46CE    3A 46E1                  LD      A,(COL)             ;GET COLOUR TO CLEAR TO
46D1    D3 01                    OUT     (01),A
46D3    0B                       DEC     BC
46D4    78                       LD      A,B
46D5    B1                       OR      C
46D6    20 F6                    JR      NZ,CLS2
46D8    3E C0                    LD      A,0C0H              ;NOW SET UP VDP REG 1 CORRECTLY
46DA    D3 02                    OUT     (2),A               ;AND UN-BLANK THE SCREEN
46DC    3E 81                    LD      A,81H               ;WHICH SOULD NOW BE
46DE    D3 02                    OUT     (2),A               ;CLEARED
46E0    C9                       RET

46E1    11              COL:     DB      11H


                        ;Universal print routine using KSUB1.  The routine assumes that HL points
                        ;to data string on entry .... the data string must be terminated with FFh.

46E2                    PRINT:

46E2    F5                       PUSH    AF              ;PRESERVE A
46E3                    PRINT2:
46E3    7E                       LD      A,(HL)          ;GET A BYTE
46E4    FE FF                    CP      0FFH            ;IS IT THE END OF STRING
46E6    20 06                    JR      NZ,MESEND       ;IF IT IS GO BACK
46E8    CD 4199                  CALL    KSUB1           ;OTHERWISE GO PRINT IT
46EB    23                       INC     HL              ;BUMP TO NEX PLACE IN STRING
46EC    18 F5                    JR      PRINT2
46EE                    MESEND:
46EE    F1                       POP     AF
46EF    C9                       RET

                        ;This routine plots a point on bit=mapped screen at CO-RDS held in IX,IY
                        ;routine presumes that pattern generator is a #0000.

46F0                    PLOT:

46F0    E5                       PUSH    HL
46F1    D5                       PUSH    DE
46F2    C5                       PUSH    BC
46F3    F5                       PUSH    AF              ;MAKE SURE ALL REGS ARE PRESERVED
```

```
46F4   CD 471B            CALL   CALCAD         ;GO CALCULATE VRAM ADDRESS

46F7   CB FF              SET    7,A
46F9   1C                 INC    E
46FA               PLT2:
46FA   1D                 DEC    E              ;MAKE SURE IF ZERO .. IS ZERO
46FB   28 03              JR     Z,PLT3
46FD   0F                 RRCA
46FE   18 FA              JR     PLT2
4700               PLT3:
4700   CD 4708            CALL   SETCOL         ;GO SEND IT TO VRAM AND COLOUR TABLE
4703   F1                 POP    AF
4704   C1                 POP    BC
4705   D1                 POP    DE
4706   E1                 POP    HL
4707   C9                 RET

4708               SETCOL:
4708   C5                 PUSH   BC
4709   E1                 POP    HL             ;GET VRAM ADDRESS INTO HL
470A   CD 429C            CALL   ADDOUT         ;SEND ADDRESS
470D   D3 01              OUT    (1),A          ;SEND DOT
470F   CB EC              SET    5,H            ;ALIGN TO COLOUR VRAM ADDRESS
4711   CD 429C            CALL   ADDOUT
4714   3A 471A            LD     A,(PLTCOL)     ;GET COLOUT OF DOT
4717   D3 01              OUT    (1),A          ;SEND IT TO VRAM
4719   C9                 RET

471A               PLTCOL:
471A   00                 DB     00             ;PLOT COLOUR USED BY ABOVE ROUTINE


                   ;Following routine calculates Vram address from x,y pos held in ix,iy

                   ;Exits BC = Vram Address & A = dot mask to be used on return.

471B               CALCAD:
471B   DD E5              PUSH   IX             ;X CO-ORD
471D   C1                 POP    BC             ;INTO BC
471E   FD E5              PUSH   IY             ;Y CO-ORD
4720   D1                 POP    DE             ;INTO DE

4721   21 00BF            LD     HL,0BFH        ;GET TEST MASK 191
4724   ED 52              SBC    HL,DE          ;SUBTRACT Y CO-ORD
4726   D8                 RET    C              ;ERROR IF >192
4727   3E 07              LD     A,7            ;BIT COUNT
4729   91                 SUB    C              ;SUB XCO-ORD YES ALL OF IT!
472A   E6 07              AND    7              ;THIS GIVES BIT NO FOR DOT
472C   3C                 INC    A              ;INC IT BECAUSE MUST BE 1-8
472D   57                 LD     D,A            ;SAVE IT IN D (D NOT USED ONLY E FOR Y POS)
472E   79                 LD     A,C            ;GET X CO-ORD
472F   E6 F8              AND    0F8H           ;1111 1000B     GET THE PICTURE ?
4731   4F                 LD     C,A
4732   7D                 LD     A,L            ;RESULT OF SBC HL,DE
4733   E6 07              AND    7
4735   B1                 OR     C              ;ADD C TO A RESULT
4736   4F                 LD     C,A            ;LOW BYTE OF VRAM ADDRESS NOW IN C
```

```
4737    7D                              LD      A,L
4738    0F                              RRCA                    ;/2
4739    0F                              RRCA                    ;/4
473A    0F                              RRCA                    ;/8
473B    E6 1F                           AND     1FH             ;

473D    47                              LD      B,A             ;HIGH BYTE ADDRESS IN B
473E    E6 F8                           AND     0F8H
4740    5F                              LD      E,A
4741    78                              LD      A,B
4742    E6 07                           AND     7
4744    B3                              OR      E
4745    47                              LD      B,A
4746    79                              LD      A,C
4747    D3 02                           OUT     (02),A          ;SEND IT TO VRAM
4749    78                              LD      A,B
474A    E6 3F                           AND     3FH             ;MAKE SURE VDP KNOWS ITS A READ
474C    D3 02                           OUT     (02),A
474E    DB 01                           IN      A,(1)           ;GET VRAM PATTERN

4750    1E 08                           LD      E,8

4752                            CALC2:
4752    0F                              RRCA
4753    1D                              DEC     E
4754    15                              DEC     D
4755    20 FB                           JR      NZ,CALC2
4757    B7                              OR      A
4758    C9                              RET                     ;RETURN WITH BIT PATTERN IN A


                                END
Connect Four Assembler version for magazines only  <c> K. Hook 1987     MACRO-80 3.44   09-Dec-81
```

SPECIAL NOTICE!

TO ALL 3.5″ SYSTEM OWNERS

SEND YOUR OWN 3.5" BLANK DISC

Pay 5.25′ Prices

MEMOTEXT

```
47B1            CALL LRAM
47B4            JP GETCHR
47B7 NEWSIZE:LD A,(DE)
47B8            OUT (1),A
47BA            LD BC,MES4
47BD            CALL SCR2
47C0            JR SIZE2
47C2 SIZE:      LD BC,MES3
47C5            CALL SCR2
47C8 SIZE2:     LD BC,MES5
47CB            CALL SCR3
47CE            LD HL,£1F93
47D1            CALL LRAM
47D4            CALL CSR
47D7            CALL DEC
47DA            LD HL,RES
47DD            LD A,(HL)
47DE            LD (COL),A
47E1 SIZE3:     LD BC,MES6
47E4            CALL SCR3
47E7            LD HL,£1F93
47EA            CALL LRAM
47ED            CALL CSR
47F0            CALL DEC
47F3            LD BC,MES7
47F6            CALL SCR3
47F9            LD HL,RES
47FC            LD A,(HL)
47FD            LD (ROW),A
4800 CHECK:     CALL SCREEN
4803            LD A,(ROW)
4806            CP 241
4808            JP NC,SIZE
480B            CP 21
480D            JP C,SIZE
4810            LD A,(COL)
4813            CP 241
4815            JP NC,SIZE
4818            CP 41
481A            JP C,SIZE
481D            LD E,A
481E            LD D,0
4820            LD (MPR),A
4823            LD A,(ROW)
4826            ADD A,2
4828            LD (MPD),A
482B            CALL MULT
482E            LD BC,(RES)
4832            LD A,B
4833            CP £4C
4835            JR C,CHECK2
4837            JR Z,CHECK1
4839            JP SIZE
```

```
483C CHECK1:  LD A,C
483D          CP £E0
483F          JR C,CHECK2
4841          JR Z,CHECK2
4843          JP SIZE
4846 CHECK2:  LD HL,STORE
4849          ADD HL,BC
484A          LD (STOREND),HL
484D          XOR A
484E          SBC HL,DE
4850          LD (XTRAROW),HL
4853          XOR A
4854          SBC HL,DE
4856          LD (LASTROW),HL
4859          LD BC,MES0
485C          CALL SCR2
485F          LD BC,MES14
4862          CALL SCR3
4865          LD BC,£4CE0
4868 OLDNEW:  CALL £79
486B          CP "n"
486D          JP Z,CLEAR2
4870          CP "o"
4872          JP Z,RESET
4875          JR OLDNEW
4877 SCREEN:  DI
4878          LD HL,£1F48
487B          CALL LRAM
487E          LD B,40
4880          LD A,"^"
4882 SCR1:    OUT (1),A
4884          DJNZ SCR1
4886          EI
4887          LD BC,MES1
488A SCR2:    LD HL,£1F70
488D          JR SCR5
488F SCR3:    LD HL,£1F84
4892          JR SCR5
4894 SCR4:    LD HL,£1F7A
4897 SCR5:    DI
4898          CALL LRAM
489B          LD H,B
489C          LD L,C
489D          LD B,20
489F SCR6:    LD A,(HL)
48A0          OUT (1),A
48A2          INC HL
48A3          DJNZ SCR6
48A5          LD HL,£1F98
48A8          CALL LRAM
48AB          LD B,40
48AD          LD A," "
48AF SCR7:    OUT (1),A
```

```
48B1          DJNZ SCR7
48B3          EI
48B4          RET
48B5 MES0:    DB "  *** MEMOTEXT ***  "
48C9 MES1:    DB " Row      Column    "
48DD MES2:    DB "  Are you sure y/n ?"
48F1 MES3:    DB " Screen size error  "
4905 MES4:    DB " Alter screen size  "
4919 MES5:    DB " Enter columns      "
492D MES6:    DB " Enter rows         "
4941 MES7:    DB "                    "
4955 MES8:    DB "      Escape ?      "
4969 MES9:    DB "     Fast scroll    "
497D MES10:   DB "  Ink (1-15)        "
4991 MES11:   DB "  Paper (1-15)      "
49A5 MES12:   DB "   Tabulation off   "
49B9 MES13:   DB "     Printing       "
49CD MES14:   DB " New or Old  (n/o) ?"
49E1 MES15:   DB " Enter document name"
49F5 MES16:   DB "  Please start tape "
4A09 MES17:   DB "then press any key. "
4A1D MES18:   DB " Saving:            "
4A31 MES19:   DB " Located:           "
4A45 MES20:   DB " Loading:           "
4A59 MES21:   DB " Verify:            "
4A6D MES22:   DB "       Insert       "
4A81 MES23:   DB "*** Break or Error i"
4A95 MES24:   DB "n tape routines. ***"
4AA9          NOP
4AAA ESCAPE:  LD A,32
4AAC          LD (£FA91),A
4AAF          PUSH HL
4AB0          LD BC,MES8
4AB3          CALL SCR3
4AB6          POP HL
4AB7 ESCAPE1:CALL £79
4ABA          JR Z,ESCAPE1
4ABC          PUSH AF
4ABD          PUSH HL
4ABE          CALL LRAM
4AC1          LD A,(DE)
4AC2          OUT (1),A
4AC4          LD BC,MES7
4AC7          CALL SCR3
4ACA          POP HL
4ACB          POP AF
4ACC          CALL LRAM
4ACF          CP 25
4AD1          JP Z,FAST
4AD4          CP 8
4AD6          JP Z,FAST
4AD9          CP 10
4ADB          JP Z,FAST
4ADE          CP 11
4AE0          JP Z,FAST
4AE3          CP "c"
4AE5          JP Z,COLOURS
4AE8          CP "i"
```

```
4AEA          JR Z,ESCI
4AEC          CP "j"
4AEE          JR Z,ESCJ
4AF0          CP "k"
4AF2          JP Z,ESCK
4AF5          CP "l"
4AF7          JP Z,LOAD
4AFA          CP "p"
4AFC          JP Z,PRINT1
4AFF          CP "s"
4B01          JP Z,SAVE
4B04          CP "v"
4B06          JP Z,VERIFY
4B09          CP "x"
4B0B          JR Z,EXIT
4B0D          CP 3
4B0F          JP Z,UPDATE
4B12          CP 28
4B14          JP Z,NEWSIZE
4B17          CP 29
4B19          JP Z,NEWSIZE
4B1C          JR ESCAPE
4B1E EXIT:    LD A,160
4B20          LD (£FA91),A
4B23          LD BC,960
4B26          LD HL,£1C00
4B29          CALL LRAM
4B2C          CALL CLS1
4B2F          CALL LRAM
4B32          LD A,32
4B34          OUT (1),A
4B36          LD HL,£18AF
4B39          LD (£FD55),HL
4B3C          LD A,£C3
4B3E          LD (£FD54),A
4B41          LD A,15
4B43          LD (£FD5E),A
4B46          RST 38
4B47 ESCI:    CALL LINE
4B4A          PUSH HL
4B4B          CALL ESC3
4B4E          LD HL,(STOREND)
4B51          XOR A
4B52          SBC HL,DE
4B54          LD B,H
4B55          LD C,L
4B56          LD DE,(STOREND)
4B5A          DEC DE
4B5B          LD HL,(XTRAROW)
4B5E          DEC HL
4B5F          LDDR
4B61          POP HL
4B62          LD A,(COL)
4B65          LD B,A
4B66          JR ESC2
4B68 ESCJ:    CALL LINE
4B6B          CALL ESC3
4B6E          EX DE,HL
```

```
4B6F          PUSH HL
4B70          LD B,H
4B71          LD C,L
4B72          LD HL,(STOREND)
4B75          XOR A
4B76          SBC HL,BC
4B78          LD B,H
4B79          LD C,L
4B7A          POP HL
4B7B          LDIR
4B7D          LD HL,(LASTROW)
4B80          LD A,(COL)
4B83          LD B,A
4B84 ESC2:    LD (HL),32
4B86          INC HL
4B87          DJNZ ESC2
4B89          LD HL,(TEMP2)
4B8C          LD DE,(TEMP3)
4B90          CALL SCROLL
4B93          JP GETCHR
4B96 ESCK:    CALL LINE
4B99          CALL ESC3
4B9C          LD A,(COL)
4B9F          LD C,A
4BA0          LD B,0
4BA2          LDIR
4BA4          LD HL,(TEMP2)
4BA7          LD DE,(TEMP3)
4BAB          CALL SCROLL
4BAE          JP DOWN
4BB1 ESC3:    PUSH HL
4BB2          LD A,(COL)
4BB5          LD E,A
4BB6          LD D,0
4BB8          ADD HL,DE
4BB9          EX DE,HL
4BBA          POP HL
4BBB          RET
4BBC COLOURS:PUSH DE
4BBD          PUSH HL
4BBE          LD A,(DE)
4BBF          OUT (1),A
4BC1 COL1:    LD BC,MES10
4BC4          CALL SCR3
4BC7          LD HL,£1F93
4BCA          CALL LRAM
4BCD          CALL CSR
4BD0          CALL DEC
4BD3          LD A,(RES)
4BD6          CP 16
4BD8          JR NC,COL1
4BDA          CP 0
4BDC          JR Z,COL1
4BDE          LD (ACC1),A
4BE1 COL2:    LD BC,MES11
4BE4          CALL SCR3
4BE7          LD HL,£1F93
4BEA          CALL LRAM
4BED          CALL CSR
```

```
4BF0          CALL DEC
4BF3          LD A,(RES)
4BF6          CP 16
4BF8          JR NC,COL2
4BFA          CP 0
4BFC          JR Z,COL2
4BFE          LD B,A
4BFF          LD A,(ACC1)
4C02          CP B
4C03          JR Z,COL2
4C05          LD A,B
4C06          LD HL,ACC1
4C09          RLD
4C0B          LD A,(HL)
4C0C          OUT (2),A
4C0E          LD A,7
4C10          OR £80
4C12          OUT (2),A
4C14          LD BC,MES7
4C17          CALL SCR3
4C1A          POP HL
4C1B          POP DE
4C1C          CALL LRAM
4C1F          CALL CSR
4C22          CALL LRAM
4C25          JP GETCHR
4C28 PRINT1:  PUSH AF
4C29          PUSH BC
4C2A          PUSH DE
4C2B          PUSH HL
4C2C          LD A,(COL)
4C2F          CP 80
4C31          JR NZ,PRINT3
4C33          LD A,(ROW)
4C36          CP 56
4C38          JR NZ,PRINT3
4C3A          LD BC,MES13
4C3D          CALL SCR3
4C40          DI
4C41          LD B,27
4C43          CALL £0CE3
4C46          LD B,"Q"
4C48          CALL £0CE3
4C4B          LD B,80
4C4D          CALL £0CE3
4C50          LD HL,STORE
4C53          LD DE,4480
4C56 PRINT2:  LD B,(HL)
4C57          CALL £0CE3
4C5A          INC HL
4C5B          DEC DE
4C5C          LD A,D
4C5D          OR E
4C5E          JR NZ,PRINT2
4C60          EI
4C61          LD BC,MES7
4C64          CALL SCR3
4C67 PRINT3:  POP HL
4C68          POP DE
```

```
4C69          POP BC
4C6A          POP AF
4C6B          CALL LRAM
4C6E          JP GETCHR
4C71 SAVE:    LD (TEMP2),HL
4C74          LD (TEMP3),DE
4C78          LD A,0
4C7A          LD (£FD67),A
4C7D          LD (£FD68),A
4C80          CALL HEADER
4C83          LD BC,MES18
4C86          CALL SCR2
4C89          LD BC,MES7
4C8C          CALL SCR3
4C8F          LD BC,BUFFER
4C92          CALL SCR4
4C95          CALL DIMENS
4C98          CALL HEAD1
4C9B          CALL DELAY
4C9E SAVE1:   CALL TAPE
4CA1          LD DE,(TEMP3)
4CA5          LD HL,(TEMP2)
4CA8          CALL LRAM
4CAB          JP UPDATE
4CAE DIMENS:  LD BC,MARK
4CB1          DEC BC
4CB2          DEC BC
4CB3          LD A,(ROW)
4CB6          LD (BC),A
4CB7          INC BC
4CB8          LD A,(COL)
4CBB          LD (BC),A
4CBC          RET
4CBD LOAD:    LD (TEMP2),HL
4CC0          LD (TEMP3),DE
4CC4          LD A,0
4CC6          LD (£FD67),A
4CC9          INC A
4CCA          LD (£FD68),A
4CCD          CALL HEADER
4CD0          LD BC,MES19
4CD3          CALL SCR2
4CD6          LD BC,MES7
4CD9          CALL SCR3
4CDC          CALL LOAD2
4CDF          LD BC,MES20
4CE2          CALL SCR2
4CE5          LD BC,BUFFER
4CE8          CALL SCR4
4CEB          CALL TAPE
4CEE          JP RESET
4CF1 LOAD2:   LD HL,(XTRAROW)
4CF4          CALL HEAD2
4CF7          LD BC,(XTRAROW)
4CFB          CALL SCR4
4CFE          LD HL,(XTRAROW)
4D01          LD DE,BUFFER
4D04          LD B,19
4D06 LOAD3:   LD A,(DE)
4D07          CP (HL)
4D08          JR NZ,LOAD2
4D0A          INC HL
4D0B          INC DE
4D0C          DJNZ LOAD3
4D0E          LD A,(HL)
4D0F          LD (ROW),A
4D12          INC HL
4D13          LD A,(HL)
4D14          LD (COL),A
4D17          RET
4D18 VERIFY:  LD (TEMP2),HL
4D1B          LD (TEMP3),DE
4D1F          LD A,1
4D21          LD (£FD67),A
4D24          LD (£FD68),A
4D27          CALL HEADER
4D2A          LD BC,MES21
4D2D          CALL SCR2
4D30          LD BC,MES7
4D33          CALL SCR3
4D36          CALL DIMENS
4D39          CALL HEAD1
4D3C          LD BC,MARK
4D3F          DEC BC
4D40          DEC BC
4D41          LD A,32
4D43          LD (BC),A
4D44          LD BC,BUFFER
4D47          CALL SCR4
4D4A          JP SAVE1
4D4D HEADER:  LD BC,MES15
4D50          CALL SCR2
4D53          CALL GETWORD
4D56          LD BC,MES16
4D59          CALL SCR2
4D5C          LD BC,MES17
4D5F          CALL SCR3
4D62 HEAD:    CALL £79
4D65          JR Z,HEAD
4D67          CP 3
4D69          JP Z,BRK4
4D6C          RET
4D6D HEAD1:   LD HL,BUFFER
4D70 HEAD2:   LD DE,22
4D73          CALL £AAE
4D76          RET
4D77 TAPE:    LD HL,STORE
4D7A          LD DE,£4CE0
4D7D          CALL £AAE
4D80 ENDTAPE:LD BC,MES7
4D83          CALL SCR3
4D86          LD HL,(TEMP2)
4D89          LD DE,(TEMP3)
4D8D          CALL LRAM
4D90          RET
4D91 GETWORD:PUSH AF
4D92          PUSH BC
4D93          PUSH DE
```

```
4D94          PUSH HL
4D95 G1:      LD DE,BUFFER
4D98          LD BC,MES7
4D9B          CALL SCR3
4D9E          LD B,21
4DA0          LD A,32
4DA2 G2:      LD (DE),A
4DA3          INC DE
4DA4          DJNZ G2
4DA6 G3:      LD DE,BUFFER
4DA9          LD A,£FF
4DAB          LD (MARK),A
4DAE          LD B,19
4DB0          LD HL,£1F85
4DB3 G4:      CALL LRAM
4DB6          CALL CSR
4DB9          CALL LRAM
4DBC G5:      CALL £79
4DBF          JR Z,G5
4DC1          CP 3
4DC3          JP Z,BRK4
4DC6          CP 8
4DC8          JR Z,GL
4DCA          CP 25
4DCC          JR Z,GR
4DCE          CP 13
4DD0          JR Z,GETOUT
4DD2          CP 32
4DD4          JR C,G5
4DD6          CP 123
4DD8          JR NC,G5
4DDA          LD (DE),A
4DDB GR:      LD A,(DE)
4DDC          OUT (1),A
4DDE          INC DE
4DDF          CALL CSR
4DE2          INC HL
4DE3          CALL LRAM
4DE6          DJNZ G5
4DE8          LD HL,£1F98
4DEB          CALL LRAM
4DEE          LD A,32
4DF0          OUT (1),A
4DF2          JP G3
4DF5 GL:      LD A,19
4DF7          CP B
4DF8          JR Z,GL1
4DFA          INC B
4DFB          LD A,(DE)
4DFC          OUT (1),A
4DFE          DEC DE
4DFF          DEC HL
4E00          JR G4
4E02 GL1:     LD A,(DE)
4E03          OUT (1),A
4E05          LD BC,18
4E08          LD HL,BUFFER
4E0B          ADD HL,BC
4E0C          EX DE,HL
4E0D          LD HL,£1F85
```

```
4E10          ADD HL,BC
4E11          LD B,1
4E13          JP G4
4E16 GETOUT:  POP HL
4E17          POP DE
4E18          POP BC
4E19          POP AF
4E1A          RET
4E1B STORE:   DS 240
4F0B          DS 240
4FFB          DS 240
50EB          DS 240
51DB          DS 240
52CB          DS 240
53BB          DS 240
54AB          DS 240
559B          DS 240
568B          DS 240
577B          DS 240
586B          DS 240
595B          DS 240
5A4B          DS 240
5B3B          DS 240
5C2B          DS 240
5D1B          DS 240
5E0B          DS 240
5EFB          DS 240
5FEB          DS 240
60DB          DS 240
61CB          DS 240
62BB          DS 240
63AB          DS 240
649B          DS 240
658B          DS 240
667B          DS 240
676B          DS 240
685B          DS 240
694B          DS 240
6A3B          DS 240
6B2B          DS 240
6C1B          DS 240
6D0B          DS 240
6DFB          DS 240
6EEB          DS 240
6FDB          DS 240
70CB          DS 240
71BB          DS 240
72AB          DS 240
739B          DS 240
748B          DS 240
757B          DS 240
766B          DS 240
775B          DS 240
784B          DS 240
793B          DS 240
7A2B          DS 240
7B1B          DS 240
7C0B          DS 240
7CFB          DS 240
7DEB          DS 240
```

```
7EDB            DS 240
7FCB            DS 240
80BB            DS 240
81AB            DS 240
829B            DS 240
838B            DS 240
847B            DS 240
856B            DS 240
865B            DS 240
874B            DS 240
883B            DS 240
892B            DS 240
8A1B            DS 240
8B0B            DS 240
8BFB            DS 240
8CEB            DS 240
8DDB            DS 240
8ECB            DS 240
8FBB            DS 240
90AB            DS 240
919B            DS 240
928B            DS 240
937B            DS 240
946B            DS 240
955B            DS 240
964B            DS 240
973B            DS 240
982B            DS 240
991B            DS 240
9A0B            DS 240
9AFB BUFFER:    DS 21
9B10 MARK:      DB £FF
9B11            RET
9B12            RET
9B13            RET
9B14            RET
9B15            RET
9B16            RET
9B17            RET
9B18            RET
9B19            RET
```

Symbols:

CLEAR	4038	X	4022
Y	4023	XX	4024
YY	4025	CHR	4026
INS	4027	TEMP1	4028
TEMP2	402A	TEMP3	402C
TEMP4	402E	STORE	4E1B
CLEAR1	404A	CLS	4303
LRAM	42F8	RESET	40DE
GETCHR	4119	LEFT	41B2
DOWN	41DC	UP	4221
OUTCHR	4173	INS2	46A5
RIGHT	4183	SCROLL	4267
CSR	4318	SCROLL1	4284
CLS1	430C	LEFT1	41C0
LEFT2	41D5	DOWN1	41F4
TAB	444E	CTRLW	451D
INSERT	4678	DELETE	46C4

EOL	46E9	RETURN	457F
ESCAPE	4AAA	UP1	423B
MULT	431E	MPR	433F
MPD	4340	MULT1	4331
NOADD	4335	RES	4341
LINE	4428	TAB6	44CD
TAB5	44C4	CW1	4526
CW2	452D	CW3	453C
CW4	4550	CW7	4579
CW5	455F	CW6	4567
TAB1	4457	TAB2	4473
TAB3	4495	TAB4	44A5
RET2	4590	RET4	45C5
RET3	459B	RET5	45EE
RET7	45F3	RET8	4607
RET10	464F	RET9	4612
PRINT1	4C28	PRINT2	4C56
INS1	4692	LASTROW	4032
XTRAROW	4034	EOL1	46F5
EOL2	4706	INS3	46B7
DEL1	46D7	ESCI	4B47
ESCJ	4B68	ESCK	4B96
ESC3	4BB1	ESC2	4B84
JOYS	4723	FAST	47A1
EXIT	4B1E	KEY	4758
STROBE	4759	LL	4732
RR	473B	UU	4744
DD	474D	DONE	4760
SLOW	4782	COL	4037
ROW	4036	RET11	4673
CHECK	4800	SIZE	47C2
DIV	4356	DIV1	4365
DIV2	436C	SIZE2	47C8
SIZE3	47E1	BUFF	434C
DEC1	4385	DEC2	438D
DEC3	43A5	DEC4	43B4
DEC5	43C3	VAL1	4351
ACC1	4344	RES1	4343
ACC2	4345	DEC6	43CB
DEC7	43D5	OKN	43F3
HIGH	43D9	DEC8	43FE
MULT2	4406	MULT3	440B
EMPT	4347	DEC	437E
HIGH2	43E3	NEWSIZE	47B7
ESCAPE1	4AB7	CLEAR2	4059
XPOS	470D	YPOS	4718
SCREEN	4877	SCR1	4882
MES1	48C9	SCR2	488A
MES2	48DD	CLEAR5	406F
CLEAR3	405C	CLEAR4	4064
MES3	48F1	MES4	4905
SCR3	488F	MES5	4919
MES6	492D	MES7	4941
MES8	4955	DLY	479D
MES9	4969	FIRE	4729
COLOURS	4BBC	MES10	497D
MES11	4991	COL1	4BC1
COL2	4BE1	CHECK2	4846
CHECK1	483C	OKTAB	44EE
NOTAB	450B	MES12	49A5

RIGHT1	4191	RIGHT2	41AB	LOAD2	4CF1	LOAD3	4D06
DOWN2	4217	UP2	425B	ENDTAPE	4D80	OLDNEW	4868
STOREND	4030	UPD	42CD	SCR7	48AF	SCR6	489F
UPD1	42DE	UPD2	42E0	SCR5	4897	SCR4	4894
UPD3	42E5	UPD4	42ED	HEAD1	4D6D	HEAD	4D62
UPD5	42EE	TAB7	44D7	MES21	4A59	MARK	9B10
TAB8	44E4	TAB9	44E5	BUFFER	9AFB	G1	4D95
TAB10	44E9	SCROLL2	428E	G2	4DA2	G3	4DA6
UPDATE	42A0	OUTCHR1	417B	G4	4DB3	G5	4DBC
DEL2	46D9	MES13	49B9	GETOUT	4E16	GL	4DF5
PRINT3	4C67	MES14	49CD	GL1	4E02	GR	4DDB
SAVE	4C71	TAPE	4D77	BREAK	408F	SAVE1	4C9E
MES15	49E1	MES16	49F5	MES22	4A6D	DIMENS	4CAE
MES17	4A09	MES19	4A31	MES23	4A81	MES24	4A95
MES18	4A1D	GETWORD	4D91	MES0	48B5	BRK1	40A1
LOAD	4CBD	MES20	4A45	BRK2	40A8	BRK3	40C1
VERIFY	4D18	HEADER	4D4D ↑	BRK4	40C8	STACK	4517
DELAY	479B	HEAD2	4D70				

The Complete Price List

Hardware

DESCRIPTION	MEMBERS PRICE	NON MEMBERS PRICE	CARRIAGE
COMPLETE CP/M PACKAGE			
1 X 1 MBYTE 3.5" INDUSTRY STANDARD DISC DRIVE, 500K FAST ACCESS RAM DISC CP/M 2.2 OPERATING SYSTEM. 256K RAM. 12" GREEN SCREEN MONITOR CENTRONICS STANDARD PRINTER I/F POSITIVE ACTION KEYBOARD. COLOUR MONITOR OUTPUT. TWO JOYSTICK I/F.	359.95	399.95	12.96
PURCHASES INDIVIDUALLY (basic system)			
256K COMPUTER PLUS TAPE OPERATING SYSTEM	89.95	99.95	6.48
CP/M SYSTEM			
1 X 1 MBYTE 3.5" DRIVE + 512 SILICON DISC + 80 COL + CP/M + N.W.	237.59	264.00	6.48
HX 12" GREEN SCREEN MONITOR	85.49	95.00	6.48
TWIN RS232 INTERFACE	26.96	29.95	2.00
FDX 2 X 1 MBYTE CP/M + 2 MBYTE SILICON DISC.	877.50	975.00	6.48
32K MEMORY EXPANSION	37.95	39.95	2.00
64K MEMORY EXPANSION	47.45	49.95	2.00
128K MEMORY EXPANSION	75.95	79.95	2.00
NEWWORD ON ROM	37.95	39.95	2.00
PASCAL ON ROM	37.95	39.95	2.00
RS232 INTERFACE (FULL BOARD)	37.95	39.95	2.00