ATOMIC MACHINE CODE

ECCE
PRODUCTIONS

ATOMIC MACHINE CODE

ATOMIC MACHINE CODE

ATOMIC MACHINE CODE

ECCE
PRODUCTIONS

Contents

AUTONUMBER	5
ON ERROR	9
SEARCH LINES	12
BRK & PAGE O ROUTINE	15
DELETE LINES	19
RENUMBER	23
FIND END OF DATA	28
REALTIME CLOCK	29
READ/DATA/RESTORE	32
*CAT STORE	36
DIRECT HEX	39
KEY SCAN	43
TRANSFER TEXT	44
FLASHING LINES	45
LONG STRINGS	47
ALPHABETA SORT	49
DESIGN LETTERS	52
MENU	59
FRUIT MACHINE	65
STRING WRITE MODE4	69
PONTOON	72
POOLS PREDICATION	86
SCREENCHASE	99
SUBROUTINES USED	105
M/C INSTRUCTIONS	106

© 1982 Ecce Productions

Introduction

This book is not intended as a text book nor as a simple collection of programmes, it is intended to take the reader one step further down the machine code road.

What are the advantages of machine code over BASIC, only one really and that is speed. You will see as you read this book that many of the programmes have delays written in to make them perform at a more human rate. On the other hand a machine code programm can take an enormous time to de-bug but the BRK & PAGE 0 programme does help!

Many of the programmes in this book are tool kit type but the manipulation of BASIC lines with the comparisons, the movement of data and the large areas of text are an ideal starting point for demonstrating machine code programmes. The book assumes the reader has a good working knowledge of BASIC and therefore there are no notes to the BASIC sections of the following programmes.

The notes to the machine code are all interelated and therefore the last programme in the book has less notes than the first.

All the programmes work and at the time of writing would not seem to have any bugs in them, but they are not guaranteed bug free!. Somebody will find a way to make them crash.

In a few cases unnecessary lines have been left in to make the actions clearer, if you find them on first reading you are an advanced programmer and these programmes will be of little use to you!

A full list of the ATOM subroutines used appears at the end of the book plus a list of all the 6502 instructions with their addressing modes.

When you have assembled these programmes remember to *SAVE the machine code to tape as well as the text.

Autonumber

This programme will automatically provide line numbers, any start and any step, when called by LINK#2850 in direct mode. It provides a space between the line number and first character except when a label is used or an assembler programme is entered.
A combination of BASIC and machine code is used and the assembled programme just fits into one page (256 bytes). Take care not to enter any extra spaces in the BASIC part and relocate from page #28 if F.P. ROM is used.

PAGE 0 LOCATIONS USED:-

#5,#6	Address of current BASIC statement.
#D,#E	Address of TOP.
#16,#25	Used by subroutine #C589.
#80,#81	Contains Start No in # , H.B in #81.
#82	Contains Step.
#83	Bit 7 set if M/C, otherwise 0.
#85,#86	Temp storage for BASIC address.
#100--	Input Line Buffer.
#321	Field Width Pointer.

SUBROUTINES USED:-

#C2F2 #C589 #FE94 #FE52 #FFED

PART NO: 1

ENTER IN TEXT AREA: #29
TEXT: 1K
VDU: -
M/C: .16K
EXECUTION: -

```
5REM AUTO NUMBER                    :
10DIMKK12                           :
20F.N=0 TO12;DIMP-1;KKN=P;N.        :
30F.N=1TO2;P=#2850;GOS.a;N.         :
50E.                                :
500a[                               :
510:KK0LDA#5                        :Save address of
520STA#85                           :BASIC statement
530LDA#6                            :
540STA#86                           :
550LDA@#28                          :Store new BASIC
560STA#6                            :address in #5,#6
570LDA@3                            :Change line 550 if
580STA#5                            :different page used
590JMP#C2F2                         :GOTO BASIC
600LDA#85                           :Restore original address
610STA#5                            :
620LDA#86                           :
```

```
630STA#6                    :
640DEC#D                    :Move TOP back by 1
650LDA@0                    :Set space pointer
660STA#83                   :to zero
670:KK2LDX@0                :Set X and
680STX#321                  :Field Width to zero
710LDA#81                   :Get H.B of Start Number
720STA#25                   :store in #25 and in
730STA#100,X                :Input Line Buffer
740INX                      :Add 1 to Buffer pointer
750LDA#80                   :Get L.B of Start No
760STA#16                   :store in #16 and in
770STA#100,X                :Input Line Buffer then
780JSR#C589                 :GOSUB print No to VDU
790LDX@1                    :X is destroyed by #C589
800:KK3INX                  :so reset X
810:KK12JSR#FE94            :Get ASCII code from K/B
820CMP@#1B                  :Has ESC key been pressed
830BNEKK4                   :if not GOTO KK4
840INC#D                    :otherwise reset TOP
850RTS                      :and exit
860:KK4CMP@#5B              :Has "[" been pressed
870BNEKK5                   :if not GOTO KK5
880PHA                      :Save ASCII code on stack
890LDA@#80                  :Set bit 7 of #83 to
900STA#83                   :indicate M/C entry
910PLA                      :Get code from stack
920:KK5CPX@2                :Have 3 codes been entered
930BNEKK6                   :if not GOTO KK6
940BIT#83                   :Check bit 7 of #83
950BMIKK6                   :If set GOTO KK6
960CMP@#60                  :Is code a lower case label
970BCSKK6                   :If >=#60 it is so GOTO KK6
980PHA                      :Otherwise save code on stack
990LDA@#20                  :and store a space
1000STA#100,X               :in Input Line Buffer
1010INX                     :Add 1 to Buffer pointer
1020PLA                     :Get code from stack and
1030:KK6JSR#FE52            :Output to VDU
1060CMP@#7F                 :Was it DELETE
1070BNEKK7                  :If not GOTO KK7
1080DEX                     :Subtract 1 from Buffer pointer
1090CPX@1                   :and loop for next code making
1095BNEKK12                 :certain X never points to the
1100JMPKK3                  :Line No
1110:KK7STA#100,X           :Store code in Buffer
1120CMP@#5D                 :Has "]" been pressed If so
1130BNEKK8                  :clear #83 to indicate end of
1140LDY@0                   :M/C
1150STY#83                  :
1160:KK8CMP@#D              :Has RETURN been pressed
1170BNEKK3                  :If not loop for next key
```



```
1180JSR#FFED                :Output CR/LF to VDU
1190LDY@0                   :
1200:KK9LDA#100,Y           :Store contents of Buffer
1210STA(#D),Y               :at Text space address
1220INY                     :held in #D,#E
1230CPY@2                   :Loop until #D is found
1240BCCKK9                  :but do not look for #D
1250CMP@#D                  :in Line Number
1260BNEKK9                  :Y<2 if Line No
1270LDA@#FF                 :Store End of Text Marker
1280STA(#D),Y               :after #D
1290TYA                     :Add value of Y to contents
1300CLC                     :of #D. If result >#FF Carry
1310ADC#D                   :Flag is set and contents of #E
1320STA#D                   :are incremented by 1. On exit
1330BCCKK10                 :#D,#E contain address of 1st
1340INC#E                   :free location
1350:KK10LDA#80             :Add step to Line No using
1360CLC                     :Carry Flag as above
1370ADC#82                  :
1380STA#80                  :On exit #80,#81 contain next
1390BCCKK11                 :Line No in #
1400INC#81                  :
1410:KK11JMPKK2             :Loop for next line
5000];R.                    :
```

PART NO: 2

```
ENTER IN TEXT AREA: #28
TEXT: .06K
VDU: .5K
M/C: -
EXECUTION: LINK#2850

10IN."START NUMBER"R        :
20!#80=R                    :
30IN."STEP"R                :
40?#82=R                    :
50LI.#2863                  :
```

LINES:-

510-660 Set up address of new BASIC statement and jump to interpreter. Restore original address on return.

670-780 Store Line Number in Buffer and display in decimal on VDU.

790-850 Get character from keyboard, look for ESC and exit if found.

860-910 Look for start of machine code character {[}. Check every input to allow for multi-statement lines.

920-930 Jump over next if not 1st keyboard character.

940-1020 Check if M/C entry being made, if not check for label, if no label then output space to buffer.
1030-1100 Output character to VDU, check for DELETE. If found adjust X. Suroutine #FE52 will have deleted character on VDU. Do not allow X to point to Line Number.
1110-1170 Check for end of M/C entry and for end of line.
1190-1340 Transfer contents of Buffer to current Text area and add End of Text Marker.
1350-1410 Adjust Line Number by adding Step and then loop for next line or exit.

Lines are stored by the ATOM in the following manner:

A 2 byte line number.
An optional lower case label.
A string containing the BASIC statement.
A string terminating character (#D).

The line number is stored as a hexadecimal number with the High Byte FIRST. The High Byte will only be in the range 0 to #7F as any High Byte with Bit 7 set will be interpreted as an End of Text Marker.

When looking for line terminating characters (#D), great care must be taken to ensure that line numbers are not investigated as they can contain a #D code. e.g. Line 525 is stored as #20D, #D being the Low Byte.

Locations #D,#E contain the address of TOP and are set by the instructions OLD or NEW. Location #18 contains the Text space pointer and should be set to correct page if you wish to enter Text at any page other than #29.

The use of OLD before calling AUTONUMBER will add lines to the current text in area specified by contents of #18.

On Error

A replacement On Error routine which prints out in full the offending line with an arrow to indicate the undesirable character. Some totally unnecessary graphics are included.

PAGE 0 LOCATIONS USED:-
#1,#2 Line Number where error occurred.
#3 Pointer to where error occurred.
#5 L.B of line address.
#12 Text Space H.B.
#80,#81 Address of Line where error occurred.
#82 Pointer to unrecognized character.
#DE,#DF Address of VDU line containing cursor.
#E0 Column position of cursor.
SUBROUTINES USED:-
#C589 #C9D8 #FE52 #FFED

ENTER IN TEXT AREA: #29
TEXT: .9K
VDU: .5K
M/C: .15K
EXECUTION: ON ERROR IF !#202=#2800

```
5REM ERROR PRINT                    :
10DIM CC11                          :
20F.N=0 TO 11;DIMP-1;CCN=P;N.       :
30F.N=1 TO 2;P=#2800;GOS.a;N.       :
40E.                                :
500a[                               :
510:CCOLDA@0                        :
520STA#80                           :
530STA#34                           :
540STA#43                           :
550LDA#12                           :Store Text space
560STA#81                           :H.B in #81
570LDY@0                            :
580:CC1LDA(#80),Y                   :Find line terminating
590CMP@#D                           :character routine
600BEQCC3                           :
610:CC2INY                          :
620JMPCC1                           :
630:CC3INY                          :As #D is stored in 1st text
640TYA                              :space location, this routine
650CLC                              :will return with #2901 in #80,
660ADC#80                           :#81 on first pass (if text
670STA#80                           :space in use is #29)
680BCCCC4                           :
690INC#80                           :
700:CC4LDY@0                        :
```

```
710LDA(#80),Y          :Compare 1st character of line
720CMP#2               :with #2, if = GOTO CC5 to
730BEQCC5              :check next character, if <>
740INY                 :move Y past line number and
750JMPCC2              :loop for next line
760:CC5INY             :Compare 2nd character with #1,
770LDA(#80),Y          :if <> loop for next line, if =
780CMP#1               :line where error occured has
790BNECC2              :been found
800LDA#3               :Calculate position in line of
810CLC                 :offending character , allowing
820ADC#5               :for multi-statement lines
830SEC                 :
840SBC#80              :
850STA#82              :On exit #82 will contain
855INC#82              :pointer to character
860LDX@#20             :Print graphic display at
870:CC6LDA@#2B         :specific address in VDU RAM
880STA#813F,X          :
890DEX                 :Loop until X=0
900BNECC6              :
910LDA@0               :
920STA#321             :Set field width to 0
930STA#E0              :
940LDA@#81             :Set cursor to start of
950STA#DF              :12th VDU line
960LDA@#60             :
970STA#DE              :
980LDA#1               :Store L.B of line number
990STA#16              :in #16
1000LDA#2              :Store H.B of line number
1010STA#25             :in #25 and print to VDU as a
1020JSR#C589           :decimal number on line 12
1030LDY@2              :Adjust Y to point to 1st
1040:CC7LDA(#80),Y     :character in line statement
1050CMP@#D             :and print to VDU until #D is
1060BEQCC8             :found
1070JSR#FE52           :
1080INY                :On exit Y will equal number of
1090JMPCC7             :characters printed
1100:CC8 CPY@#1F       :If length of line is < #1F
1110BCCCC9             :GOTO CC9, otherwise adjust X
1120LDX@#40            :to point to VDU line+1
1130JMPCC10            :
1140:CC9LDX@#1F        :
1150:CC10LDA@#2B       :Print display at specified VDU
1160STA#817F,X         :address
1170DEX                :
1175CPX@#20            :Loop until
1178BEQCC11            :X=#20 or
1180BNECC10            :X=0
1250:CC11LDX#82        :Print arrow at position
```



```
1260LDA@#9E            :pointed to by contents of #82
1270STA#817F,X         :
1280JSR#FFED           :Output CR/LF
1290JSR#C9D8           :GOTO standard ERROR handler
20001;R.               :
```

LINES:-

510-560 Set #80,#81 to address of text space where error occured.
580-690 Find and store address of Nth line.
700-750 Check H.B of each line with #2 which contains H.B of line where error occured.
760-790 Do same for L.B until line number is found.
800-855 Calculate position of offending character and store.
860-900 Print top line of display.
910-970 Reset cursor position.
980-1020 Print line number.
1030-1090 Print line.
1100-1180 Print 2nd line of display.
1250-1290 Print arrow then exit to standard error routine.

#1,#2 always contain number of line being interpreted, L.B first.
#5,#6 Contain address of statement being interpreted. On single statement lines this is the address of the 1st character after the line number, on multi-statement lines #5, #6 contain address of statement terminator {;} immediately before error. To print out full line ON ERROR it is necessary to find start address of line.

#3 contains pointer to the offending character as an offset from address in #5,#6, therefore on multi-statement lines a new offset must be calculated.

This routine will be automatically be called ON ERROR by BASIC if !#202 is set to the start address of the machine code. H.B in #203, L.B in #202.

!#202 is set by BASIC in direct mode to #C9D8, so !#202=#xxxx must be included as a line in the BASIC programme.

This programme may be assembled in any page without change, preferably well away from the text area in use.

Search Lines

A programme which will search any text area for a character or series of characters. If found the full line containing the character(s) will be printed out. It will return with a message if no occurrence or no text in specified area.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of line being investigated.
#82 Length of line.
#83 Used as data found flag. 0 if none found.
SUBROUTINES USED:-
#C589 #F802 #FB83 #FE52 #FFED

ENTER IN TEXT AREA: Any
TEXT: 1.2K
VDU: .5K
M/C: .16K
EXECUTION: RUN

```
5REM SEARCHLINES                :
10DIM WW18,B32,C32,D5,E12       :
20F.N=0TO18;DIMP-1;WWN=P;N.     :
30F.N=1TO2;P=#2800;GOS.a;N.     :
35P.$6$12                       :
40$C="NONE FOUND"               :
45$E="NO TEXT IN#"              :
50IN."TEXT AREA"A               :
60?#81=A;?#80=0                 :
70IN."DATA TO BE FOUND"$B       :
80LI.#2800                      :
90IN."MORE DATA IN SAME
AREA"$D                         :
100IFCH$D=CH"Y" G.60            :
110IN."NEW TEXT AREA"$D         :
120IFCH$D=CH"Y"G.50             :
130E.                           :
500a[                           :
510:WW0LDX@0                    :Set page 0 locations to 0
520STX#82                       :
530STX#83                       :
540STX#43                       :
550STX#34                       :
560LDY@0                        :
570LDA(#80),Y                   :See if there is text in area
580CMP@#D                       :
590BNEWW13                      :If not GOTO WW13
600INY                          :Move Y past line number
605INY                          :
610STY#82                       :Y now points to 1st character
```



```
620:WW1LDA#82
630CLC
640ADC#80
650STA#80
660BCCWW2
670INC#81
680:WW2LDY@0
690LDA(#80),Y
700CMP@#FF
710BEQWW10
720INY
730:WW3INY
740LDA(#80),Y
750CMP@#D
760BNEWW3
770INY
780STY#82
790LDY@1
800:WW4LDX@0
810:WW5INY
820CPY#82
830BEQWW1
840LDAB,X
850CMP@#D
860BEQWW6
870CMP(#80),Y
880BNEWW4
890INX
900JMPWW5
910:WW6LDY@0
920LDA(#80),Y
930STA#25
940INY
950LDA(#80),Y
960STA#16
970JSR#C589
980INC#83
985INY
990:WW7INY
1000LDA(#80),Y
1010CMP@#D
1020BEQWW8
1030JSR#FE52
1040JMPWW7
1050:WW8JSRWW9
1060JSR#FFED
1070JMPWW1
1080:WW9LDX@#60
1090JSR#FB83
1100RTS
1110:WW10LDA#83
1120BNEWW12
```

```
                                :Add Y to #80,#81. On 1st pass
                                :#80,#81 will contain address
                                :of 1st character, on
                                :subsequent loops #80,#81 will
                                :contain address of first
                                :location after #D character
                                :
                                :Look for End of Text Marker
                                :
                                :If found GOTO WW10
                                :
                                :Move Y past line number
                                :
                                :Look for #D
                                :Loop until found
                                :Point Y at following location
                                :and store Y in #82
                                :Point Y to start of line
                                :statement
                                :Compare each character of line
                                :with 1st character of $B, If =
                                :try next characters until #D
                                :is found in $B then GOTO WW6.
                                :If not found or characters <>
                                :reset $B pointer (X) to 0 and
                                :try all characters in line
                                :until Y=?#82 then loop for
                                :next line
                                :
                                :Set Y to point to H.B of line
                                :number, get H.B then L.B and
                                :store in page 0 locations used
                                :by #C589 then GOSUB #C589 to
                                :print line number in decimal
                                :
                                :
                                :Set #83 > 0 to indicate data
                                :found
                                :
                                :Print line to VDU
                                :
                                :
                                :
                                :
                                :GOSUB WW9 (Delay)
                                :
                                :Loop for next line
                                :Delay
                                :
                                :
                                :Has data been found
                                :If <>0 data found
```

```
1130LDX@0                       :
1140:WW11LDAC,X                 :Print $C routine
1150CMP@#D                      :
1160BEQWW12                     :
1170JSR#FE52                    :
1180INX                         :
1190JMPWW11                     :
1200:WW12JSR#FFED               :Output CR/LF
1210RTS                         :and exit
1220:WW13LDAE,X                 :Print $E routine
1230CMP@#D                      :
1240BEQWW14                     :
1250JSR#FE52                    :
1260INX                         :
1270JMPWW13                     :
1280:WW14LDA#81                 :Print contents of #81
1290JSR#F802                    :as 2 ASCII numerals
1295JSR#FFED                    :
1300JMPWW12                     :
1500];R.                        :
```

LINES:-

510-610 Checks for text in area specified.
620-670 Adjusts contents of #80,#81 so they contain address of line being investigated.
680-780 Finds length of current line, points at next location and stores result in #82.
790-900 Comparison routine.
910-1040 Prints line number and line.
1050-1100 Subroutine which delays and the prints CR/LF.
1110-1300 Prints message strings.

COMPARISON ROUTINE:

Y is pointer to character in line under investigation.
X is pointer to character in $B.
The 1st character of $B is compared with the 1st character of line, if they are not equal Y is incremented to point to next character in line and programme loops until either equality is found or Y increments past end of line. When Y moves past end of line the programme loops for next line or end of text marker.
If equality is found X and Y are incremented and subsequent characters are compared until either unequality is found or #D in $B is found.
On unequality X is reset to 0 and programme loops.
If #D is found all characters in $B must have been compared and equality found for all, so $B has been found and the programme branches to print line number and line routine.

Brk & Page O Routine

An essential machine code de-bugging tool. On BRK the processor jumps to address in #202,#203 so store address of this programme in !#202. All registers are preserved but the VDU is cleared so beware of using during graphic programmes. The display consists of the programme counter, the status register displayed as flags, the X register, the Y register and the accumulator. This programme will return automatically after a short wait unless CTRL is pressed, if so it will display contents of page 0 locations #80 - #9F inclusive and will then wait for any key to be pressed before returning.

PAGE 0 LOCATIONS USED:-
#E1 Bit 7 is cursor on/off flag.
#100-- Stack.
SUBROUTINES USED:-
#F7FD #F802 #FB83 #FE52 #FFE3 #FFED

ENTER IN TEXT AREA: Any
TEXT: 1.3K
VDU: .5K
M/C: .23K
EXECUTION: ON 'BRK' if !#202=#xx00

```
5REM BRK & PAGE0 ROUTINE        :
10DIM QQ12,KK3,R24              :
20F.N=0 TO 12;DIMP-1;QQN=P;     :
KKN=P;N.                        :
30F.N=1 TO 2;P=#3B00;GOS.a;N.   :
40F.N=1 TO 2;P=#2800;GOS.b;N.   :
50$R=" PC  NV:BDIZC  X  Y  A"   :
60LI.KK0                        :
70E.                            :
500a[                           :
510:KK0LDY@0                    :
520:KK2LDA R,Y                  :Store $R at #28E0
530STA#28E0,Y                   :
540CMP@#D                       :
550BEQKK3                       :
560INY                          :
570JMPKK2                       :
580:KK3RTS                      :
590]                            :
600R.                           :
610b[                           :
620:QQ0STA#28FD                 :Save A,Y,X
630STY#28FE                     :
640STX#28FF                     :
```

```
650TSX                  :Get stack pointer
660INX                  :Point X at return address L.B
670INX                  :
680LDA#100,X            :Get L.B
690SEC                  :Subtract
700SBC@1                :1 from L.B and
710STA#100,X            :restore to stack
720BCSQQ1               :If Carry used
730INX                  :point X at H.B
740DEC#100,X            :and adjust H.B
750:QQ1LDA@12           :Clear VDU
760JSR#FE52             :
770LDA@#80              :
780STA#DF               :
790LDA@0                :
800STA#DE               :
810LDA@6                :Set VDU column position
820STA#E0               :
830LDY@0                :Turn off cursor
840STY#E1               :
850:QQ2LDA#28E0,Y       :Print $#28E0 to VDU
860CMP@#D               :
870BEQQQ3               :
880JSR#FE52             :
890INY                  :
900JMPQQ2               :
910:QQ3JSR#FFED         :Output CR/LF and
930LDA@6                :reset VDU column
940STA#E0               :
950TSX                  :Get H.B of programme counter
960INX                  :
970INX                  :
980INX                  :
990LDA#100,X            :Print to VDU as
1000JSR#F802            :2 ASCII numerals
1010DEX                 :Point X at L.B
1020LDA#100,X           :Get L.B and print to VDU
1030JSR#F802            :as 2 ASCII numerals
1040DEX                 :Point X at status register
1050JSR#F7FD            :Output a space
1060LDA#100,X           :Get status register from stack
1070LDY@8               :Load Y with number of bits
1080:QQ4ASLA            :Shift bits left so that bit 7
1090PHA                 :falls into carry. Save
1100LDA@0               :remainder on stack and clear
1110ROLA                :Accum. Shift carry bit into
1120ADC@#30             :bit 0 of Accum, add #30 to
1130JSR#FE52            :convert to ASCII code and
1140PLA                 :output to VDU. Loop for all 8
1150DEY                 :bits
1160BNEQQ4              :
1170LDY@3               :
```



```
1180:QQ5JSR#F7FD        :Output a space, get X,Y and A
1190LDA#28FC,Y          :from store, output to VDU as
1200JSR#F802            :2 ASCII numerals
1210DEY                 :
1220BNEQQ5              :
1230JSR#FFED            :
1240LDX@#20             :Wait
1250JSR#FB83            :
1260BIT#B001            :See if Bit 6 of #B001 is clear
1270BVCQQ8              :If so CTRL is pressed GOTO QQ8
1280:QQ12LDX#28FF       :Restore X
1290LDY#28FE            :Restore Y
1300LDA@#80             :Switch on cursor
1310STA#E1              :
1320LDA#28FD            :Restore Accum
1330RTI                 :Return from interrupt
1340:QQ8 LDA@#80        :Store display data
1350STA#28FA            :
1360JSR#FFED            :
1370LDX@0               :
1380STX#28FB            :Clear pointer to page 0 store
1390LDX@4               :Load X with No of lines to
1400:QQ9STX#28FC        :print
1410LDY@8               :Load Y with No of columns to
1420STY#E0              :print and set column start
1430:QQ10LDA#28FA       :position
1440JSR#F802            :Output page 0 location names
1450JSR#F7FD            :with space between each
1460INC#28FA            :
1470DEY                 :
1480BNEQQ10             :
1490LDY@8               :Reset column position
1500STY#E0              :
1510LDX#28FB            :Load X with pointer to page 0,
1520:QQ11LDA#80,X       :get contents of page 0 and
1530JSR#F802            :output as 2 ASCII numerals to
1540JSR#F7FD            :VDU. Do for 8 locations with
1550INX                 :space between each
1560DEY                 :
1570BNEQQ11             :
1580STX#28FB            :Save pointer
1590JSR#FFED            :
1600LDX#28FC            :Get number of lines
1610DEX                 :and loop for
1620BNEQQ9              :4 lines
1630JSR#FFE3            :Wait for key to be pressed
1640JMPQQ12             :before returning
16501;R.                :
```

LINES:-

510-580 A method of storing strings.
620-740 Save registers and adjust return address.

750-910 Prints string.
930-1050 Prints programme counter.
1060-1160 Prints status register.
1170-1230 Prints X,Y and Accum.
1240-1270 Delay and see if CTRL pressed.
1280-1330 Restores registers and returns from interrupt.
1340-1640 Prints contents of #80 to #9F.

On BRK the processor pushes the programme counter +2 onto the stack followed by the status register.
On RTI the status register is restored, the programme counter is fetched from the stack and reloaded into the counter and the processor interprets the instruction at that address.
However BRK is a one byte instruction and it is necessary to adjust the saved programme counter to allow for this.
Lines 650-740 calculate the new return address and restore it to the stack.

The method used to convert the flags of the status register to 0 or 1 symbols is relatively easy for it is stored as an eight bit number.

The method is as follows:

	Accum bits 76543210	Carry bit
Line 1060	11000011	-
Line 1080	10000110	1
Line 1100	00000000	1
Line 1110	00000001	0
Line 1120	00110001	-

Accum now holds #31 which is the ASCII code for "1", this code is output to the VDU and programme loops until all eights bits are done.

The BIT instruction transfers Bits 6 and 7 of it operand into Bits 6 and 7 of the status register. These bits can be tested by using the following instructions:
BMI Branch if Bit 7 set.
BPL Branch if Bit 7 clear.
BVS Branch if Bit 6 set.
BVC Branch if Bit 6 clear.

Delete Lines

This programme will delete any line or number of lines in any text area. Once assembled in text area, say #39, it can be called by LINK#3900 or via the Menu programme later in this book.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of start line when found.
#82,#83 Address of finish line when found.
#85,#86 Address of string to be printed.
#87 Used as flag, if 1 print # number.
SUBROUTINES USED:-
#F802 #FE52 #FFED

ENTER IN TEXT AREA: Any
TEXT: 1.8K
VDU: .5K
M/C: .27K
EXECUTION: RUN or LINK#xx00

```
5REM DELETE LINES                :
10DIMKK26,WW7,C12,E23,S22        :
20F.N=0TO26;DIMP-1;KKN=P;N.      :
30F.N=0TO6;DIMP-1;WWN=P;N.       :
40$S="START NUMBER NOT FOUND"    :
50$C="NO TEXT IN #"              :
60$E="FINISH NUMBER NOT          :
FOUND"                           :
70F.N=1TO2;P=#3700;GOS.b;N.      :
80LI.WW0                         :
90F.N=1TO2;P=#3900;GOS.a;N.      :
100T=0;J=0;F=0                   :
110IN."TEXT AREA"T               :
120?#81=T;?#80=0                 :
130IN."DELETE FROM LINE NO."J    :
140!#238=J                       :
150IN."FINISH AT LINE NO."F      :
160!#23C=F                       :
170LI.#3900                      :
180E.                            :
500a[                            :
510:KK0LDX@0                     :
520STX#82                        :
530STX#83                        :
540LDY@0                         :
550LDA(#80),Y                    :See if there is text in
560CMP@#D                        :specified area
570BNEKK17                       :If not GOTO KK17
580INY                           :
```

```
590STY#80                :
600:KK1LDY@0             :
610LDA(#80),Y            :Look for End of Text marker,
620CMP@#FF               :if found GOTO KK18
630BEQKK18               :
640CMP#239               :Otherwise compare H.B of line
650BNEKK2                :number with H.B of start
660INY                   :number, if <> GOTO KK2. If =
670LDA(#80),Y            :move Y to point to next
680CMP#238               :location, get L.B of line
690BEQKK5                :number and compare to L.B of
700:KK2INY               :start number, if = start line
710:KK3INY               :is found GOTO KK5
720LDA(#80),Y            :Loop until #D is found
730CMP@#D                :
740BNEKK3                :
750INY                   :
760TYA                   :Adjust #80,#81 so that they
770CLC                   :contain address of next line
780ADC#80                :
790STA#80                :
800BCCKK4                :
810INC#81                :
820:KK4JMPKK1            :Loop for next line
830:KK5DEY               :Adjust #80,#81 so that they
840TYA                   :contain address of start line.
850CLC                   :DEY is necessary as on entry Y
860ADC#80                :points to L.B of line number.
870STA#80                :
880BCCKK6                :
890INC#81                :
900:KK6LDA#81            :Transfer address in #80,#81 to
910STA#83                :#82,#83
920LDA#80                :
930STA#82                :
940:KK7LDY@0             :Look for finish line using #82,
950LDA(#82),Y            :#83 to contain address of
960CMP@#FF               :current line and #23C,#23D
970BEQKK19               :containing finish line number.
980CMP #23D              :Use same comparison and find
990BNEKK8                :end of line routine as before
1000INY                  :
1010LDA(#82),Y           :
1020CMP#23C              :
1030BEQKK11              :
1040:KK8INY              :
1050:KK9INY              :
1060LDA(#82),Y           :
1070CMP@#D               :
1080BNEKK9               :
1090INY                  :
1100TYA                  :
```



```
1110CLC                  :
1120ADC#82               :
1130STA#82               :
1140BCCKK10              :Loop until finish line number
1150INC#83               :or end of text is found
1160:KK10JMPKK7          :
1164:KK17JMPKK25         :Stepping stones to KK25
1168:KK18JMPKK26         :and KK26
1170:KK11INY             :
1180LDA(#82),Y           :Adjust #82,#83 so that they
1190CMP@#D               :contain address of line
1200BNEKK11              :following finish line
1210INY                  :
1220TYA                  :
1230CLC                  :
1240ADC#82               :
1250STA#82               :
1260BCCKK12              :
1270INC#83               :
1280:KK12LDY@0           :Move memory from address in
1290:KK13LDA(#82),Y      :#82,#83 to address in #80,#81.
1300STA(#80),Y           :Do for each line until #D is
1310CMP@#D               :found then check for end of
1320BEQKK15              :text, if found exit otherwise
1330:KK14INY             :continue until Y increments to
1340BNEKK13              :0 then increment H.Bs in #81
1350INC#83               :and #83, then loop until end
1360INC#81               :of text
1370JMPKK13              :
1380:KK15INY             :
1390BNEKK16              :
1400INC#83               :
1410INC#81               :
1420:KK16LDA(#82),Y      :
1430STA(#80),Y           :
1440CMP@#FF              :
1450BNEKK14              :
1460RTS                  :
1470:KK25LDA@#80         :Store $C address L.B in #85
1480STA#85               :
1490INC#87               :Set #87 > 0
1500JMPKK21              :
1510:KK26LDA@#A0         :Store $S address L.B in #85
1520STA#85               :
1530JMPKK20              :
1540:KK19LDA@#C0         :Store $E address L.B in #85
1550STA#85               :
1560:KK20LDY@0           :
1570STY#87               :Clear #87
1580:KK21LDA@#3A         :Store strings H.B address in
1590STA#86               :#86
1600:KK22LDA(#85),Y      :Print message string routine
```

```
1610CMP@#D                    :
1620BEQKK23                   :
1625JSR#FE52                  :
1630INY                       :
1640JMPKK22                   :
1650:KK23LDA#87               :If ?#87=0
1660BEQKK24                   :GOTO KK24
1670LDA#81                    :Print contents of
1680JSR#F802                  :#81 as 2 numerals
1690:KK24JSR#FFED             :
1700RTS                       :
1710];R.                      :
1790b[                        :
1800:WW0 LDY@0                :String store routine
1810:WW1 LDAC,Y               :
1820STA#3A80,Y                :
1830CMP@#D                    :
1840BEQWW2                    :
1850INY                       :
1860JMPWW1                    :
1870:WW2LDY@0                 :
1880:WW3LDAS,Y                :
1890STA#3AA0,Y                :
1900CMP@#D                    :
1910BEQWW4                    :
1920INY                       :
1930JMPWW3                    :
1940:WW4LDY@0                 :
1950:WW5LDAE,Y                :
1960STA#3AC0,Y                :
1970CMP@#D                    :
1980BEQWW6                    :
1990INY                       :
2000JMPWW5                    :
2010:WW6RTS                   :
3000];R.                      :
```

LINES:-

510-590	Check for text in area specified.
600-1270	Look for start and finish lines, if found stores addresses.
1280-1460	Deletes text by moving remaining text up.
1470-1640	Prints message strings.
1650-1700	Prints # number after $C.
1800-2010	Stores message strings.

Lines 1164 and 1168 are stepping stones as the maximum distance available with Branch instructions is +129 to -127 bytes. In this case direct Branches to KK25 and KK26 would be out of range.

Renumber

A programme that will renumber any line or number of lines and sort them into the correct numerical order.

PAGE 0 LOCATIONS USED:-

#80,#81	Address of Nth line.
#82,#83	Address of Nth+1 line.
#84	Used as flag to indicate changes. 0 if no change has occured.
#85,#86	Start line number.
#87,#88	Finish line number.
#89,#8A	New start number.
#8B	Step.
#8C	Text area.

SUBROUTINES USED:-
#FE52

ENTER IN TEXT AREA: Any
TEXT: 1.9K
VDU: .5K
M/C: .29K
EXECUTION: RUN

```
5REM RENUMBER                      :
10DIMZZ29                          :
20F.N=0TO29;DIMP-1;ZZN=P;N.        :
30F.N=1TO2;P=#3200;GOS.a;N.        :
40R=#9200                          :
50$R="START NUMBER NOT FOUND"      :
60!#85=0                           :
70!#89=0                           :
80IN."START LINE"R                 :
90!#85=R                           :
100IN."END LINE"R                  :
110!#87=R                          :
120IN."NEW START NO"R              :
130!#89=R                          :
140IN."STEP"R                      :
150?#8B=R                          :
160IN."TEXT AREA"R                 :
170?#8C=R                          :
180LI.#3200                        :
200E.                              :
500a[                              :
520:ZZ0LDA#8C                      :Load #80,#81 with address of
540STA#81                          :1st line
560LDA@1                           :
570STA#80                          :
580:ZZ1LDY@0                       :Look for start line number
```

```
600:ZZ2LDA(#80),Y       :routine. Note that this
620CPY@2                :routine also looks for end of
640BCCZZ3               :text marker and end of line
660CMP@#D               :marker. A saving in size
680BEQZZ5               :compared to previous
700:ZZ3CMP@#FF          :programmes
720BEQZZ6               :
740CMP#86               :
760BEQZZ7               :
820:ZZ4INY              :
840JMPZZ2               :
860:ZZ5JSRZZ15          :Also note this subroutine
880JMPZZ1               :
900:ZZ6JMPZZ13          :Stepping stone
920:ZZ7INY              :
940LDA(#80),Y           :Compare L.B of line number
960CMP#85               :with L.B of start number
980BNEZZ4               :
1000:ZZ8LDA#89          :Get L.B of new number and
1020STA(#80),Y          :store
1040DEY                 :Point Y at H.B
1060LDA#8A              :Get new H.B and store
1080STA(#80),Y          :
1100INY                 :Move Y past
1120:ZZ9INY             :line number and
1140LDA(#80),Y          :loop until #D is found
1160CMP@#D              :
1180BNEZZ9              :
1200JSRZZ15             :Adjust address in #80,#81
1220LDA#89              :Add step to new line number
1240CLC                 :
1260ADC#8B              :
1280STA#89              :
1300BCCZZ10             :
1320INC#8A              :
1340:ZZ10LDY@0          :Look for end of text
1360LDA(#80),Y          :Look for finish line
1370INY                 :
1380CMP@#FF             :
1400BEQZZ12             :
1420CMP#88              :
1430BNEZZ8              :If not found loop for new
1440LDA(#80),Y          :number and continue until
1460CMP#87              :finish line or end of text is
1480BNEZZ8              :found
1540:ZZ11LDA#89         :Store final new number
1560STA(#80),Y          :
1580DEY                 :
1600LDA#8A              :
1620STA(#80),Y          :
1640:ZZ12JSRZZ17        :GOSUB Sort routine
1660RTS                 :
```



```
1680:ZZ13LDY@0          :Print $#9200 routine
1700:ZZ14LDA#9200,Y     :
1720CMP@#D              :
1740BEQZZ12             :
1760JSR#FE52            :
1780INY                 :
1800JMPZZ14             :
1820:ZZ15INY            :Subroutine to add Y register
1840TYA                 :to contents of #80,#81
1860CLC                 :
1880ADC#80              :
1900STA#80              :
1920BCCZZ16             :
1940INC#81              :
1960:ZZ16RTS            :
1980:ZZ17LDA#8C         :Sort subroutine
2000STA#81              :Reset #80,#81 to contain start
2020LDA@1               :address of text
2040STA#80              :
2060LDX@0               :
2070LDY@#FF             :
2080STX#84              :Clear change flag
2090:ZZ18INY            :
2100:ZZ19LDA(#80),Y     :
2110CPY@0               :Look for end of text marker
2120BNEZZ20             :only when Y=0
2130CMP@#FF             :
2140BNEZZ18             :
2160LDA#84              :If end of text check #84 to
2180BNEZZ17             :see if changes have been made,
2200RTS                 :loop until no changes made
2220:ZZ20CPY@2          :Do not check for #D in line
2230BCCZZ18             :number
2240CMP@#D              :Find end of line, point Y at
2250BNEZZ18             :next location
2260LDA#81              :Transfer H.B to #83
2280STA#83              :
2290INY                 :Add Y to #80 and store in #82
2300TYA                 :On exit #80,#81 will contain
2320CLC                 :address of line N and #82,#83
2340ADC#80              :will contain address of line
2360STA#82              :N+1
2380BCCZZ21             :
2400INC#83              :
2420:ZZ21LDY@0          :Compare H.B of line N with H.B
2440LDA(#80),Y          :of line N+1. If >= GOTO ZZ23
2460CMP(#82),Y          :otherwise transfer address in
2480BCSZZ23             :#82,#83 to #80,#81 and loop to
2500:ZZ22LDA#83         :check next line
2520STA#81              :
2540LDA#82              :
2560STA#80              :
```

```
2580LDY@0                    :
2600JMPZZ19                  :
2620:ZZ23BNEZZ24             :If H.Bs <> GOTO ZZ24
2640INY                      :Point Y at L.Bs
2660LDA(#82),Y               :Compare L.Bs, if >= loop to
2680CMP(#80),Y               :check next line
2700BCSZZ22                  :
2720:ZZ24JSRZZ25             :GOSUB Change routine
2760LDY@#FF                  :
2770JMPZZ18                  :
2780:ZZ25INC#84              :Set #84<>0 to show change
2800LDY@#FF                  :
2820:ZZ26INY                 :
2840LDA(#80),Y               :Store line N in temporary
2860STA#140,Y                :store
2880CPY@2                    :
2900BCCZZ26                  :
2920CMP@#D                   :
2940BNEZZ26                  :
2960LDY@#FF                  :
2980:ZZ27INY                 :
3000LDA(#82),Y               :Store line N+1 at address of
3020STA(#80),Y               :line N
3040CPY@2                    :
3060BCCZZ27                  :
3080CMP@#D                   :
3100BNEZZ27                  :
3120INY                      :Adjust #80,#81 to contain
3140TYA                      :address of 1st location after
3160CLC                      :line N+1's #D character
3180ADC#80                   :
3200STA#80                   :
3220BCCZZ28                  :
3240INC#81                   :
3260:ZZ28LDY@#FF             :
3280:ZZ29INY                 :
3300LDA#140,Y                :Get line N from temporary
3320STA(#80),Y               :store and store at new address
3340CPY@2                    :contained in #80,#81
3360BCCZZ29                  :
3380CMP@#D                   :
3400BNEZZ29                  :
3420LDY@0                    :
3440RTS                      :
50001;R.                     :
```

LINES:-

520-980 Find start line.
1000-1080 Store new line number.
1100-1620 Look for finish line while continuing to renumber. If finish line not found exit on end of text marker.

1680-1800 Print $#9200.
1820-1960 Subroutine for adding Y to #80,#81.
1980-2770 Comparison routine.
2780-3440 Change routine.

The comparison routine compares every line number with following number, lines are changed over if necessary and the change flag is set. When end of text is found the change flag is looked at and if changes have been made the programme loops back to check lines again. The programme will only exit if no changes have been made when end of text is found.

COMPARISON ROUTINE:
The H.B of line N is compared with the H.B of line N+1. Lines are in correct order if it is less than. Lines are in incorrect order if it is greater than, and need changing. If there is equality the L.B of N+1 is compared to the L.B of N. They are in incorrect order if it is less than and a change is necessary. If result is greater than or equal no change is necessary. This allows for two lines with the same number, a not uncommon occurrence during renumbering of sections! Note that the accumulator is loaded with the H.B of line N then with the L.B of line N+1. This allows a single branch instruction as BCS after CMP recognizes greater than and equal to.

CHANGE ROUTINE:
Line N is moved to a temporary store.
Line N+1 is moved to original address of line N.
The address of 1st location after line N+1 is found.
Line N is moved from temporary store to this address.

INDIRECT Y ADDRESSING:
There are 8 Indirect Y instructions which may be used with any page 0 reference and these are:

ADC AND CMP EOR LDA ORA SBC STA

If ?#80=0, ?#81=#39 and Y=4 then LDA(#80),Y will load the accumulator with the contents of #3904.

As Y is an 8 bit register LDA(#80),Y can only access locations #3900 to #39FF unless #81 is incremented when Y is incremented from #FF to 0. Any range from #0000 to #FFFF can then be accessed.

Find End of Data

This programme will find an End of Data Marker and return with its address. Also included is a small subroutine which clears page 0 locations. Normally these routines would be used in larger programmes.

PAGE 0 LOCATIONS USED:-
#80,#81 Start address.
#82,#83 Address of marker on exit.

```
ENTER IN TEXT AREA: Any
TEXT: .4K
VDU: -
M/C: .03K
EXECUTION: RUN

5REM FIND END OF DATA          :
10DIM DD5                      :
20F.N=0TO5;P=#3800;DDN=P;N.    :
30F.N=1TO2;P=#3800;GOS.a;N.    :
40LI.DD4                       :
50IN."START POINT"S            :
60!#80=S                       :
70LI.DD0                       :
80E=!#82                       :
90P."END POINT"&E              :
100E.                          :
500a[:DD0LDY@0                 :
510:DD1LDA(#80),Y              :Look at every location until
520CMP@#FF                     :#FF is found then branch to
530BEQDD2                      :DD2
540INY                         :
550BNEDD1                      :
560INC#81                      :
570JMPDD2                      :
580:DD2LDA#81                  :Store address of #FF in #82,
590STA#83                      :#83 and exit
600STY#82                      :
610RTS                         :
620:DD4LDY@8                   :Set 8 page 0 locations to 0
630LDA@0                       :
640:DD5STA#7F,Y                :
650DEY                         :
660BNEDD5                      :
670RTS                         :
680];R.                        :
```


Realtime Clock

This programme uses Timer 1 of the 6522 VIA chip to generate interrupts. On interrupt the processor will update a digital clock which is displayed at the top right hand corner of the VDU. This clock will continue to run until BREAK is pressed and there is no noticeable increase in execution times. No page 0 locations are used, nor any subroutines.

```
ENTER IN TEXT AREA: Any
TEXT: 1.2K
VDU: .5K
M/C: .18K
EXECUTION: RUN

5REM CLOCK                          :
10DIMPP8                            :
20R=#28F7                           :
30F.N=0TO8;P=#2800;PPN=P;N.         :
40F.N=1TO2;P=#2800;GOS.a;N.         :
50P.$12"ENTER 0 BEFORE SINGLE       :
NOS"'                               :
60IN."HOUR"$R                       :Store start time as a string
70R=R+LENR                          :of ASCII codes at address R
80IN."MINS"$R                       :
90R=R+LENR                          :
100IN."SECS"$R                      :
120?#28FD=20                        :Set 1/20th sec counter
130?#B80B=#40                       :Set Timer 1 to free run
140?#B80E=#C0                       :Enable Timer 1 interrupts
150?#B806=#50                       :Load Timer 1 low latch
160?#B805=#C3                       :Load Timer 1 high counter
170?#204=0                          :Set IRQ address
180?#205=#28                        :
190P.$12                            :
200E.                               :
500a[                               :
510:PP0PHP                          :Push status register on stack
520STY#28FF                         :Save Y and X
530STX#28FE                         :
540LDA@#40                          :Clear interrupt flag
550STA#B80D                         :
590DEC#28FD                         :Decrement 1/20th counter
600BNEPP1                           :If <>0 GOTO PP1
610LDA@20                           :Reset counter to 20
620STA#28FD                         :
630LDX@6                            :Load X with pointer to Secs*1
640JSRPP5                           :Add 1 to Secs*1
650CMP@#3A                          :If <ASCII code for 9+1
660BNEPP1                           :GOTO PP1
```

```
670JSRPP6               :Set Secs*1 to 0
680DEX                  :Point to Secs*10
690JSRPP5               :Add 1 to Secs*10
700CMP@#36              :If <> ASCII code for 6
710BNEPP1               :GOTO PP1
720JSRPP6               :Set Secs*10 to 0
730DEX                  :Point to Mins*1
740JSRPP5               :Add 1 to Mins*1
750CMP@#3A              :If <> #3A
760BNEPP1               :GOTO PP1
770JSRPP6               :Set Mins*1 to 0
780DEX                  :Point to Mins*10
790JSRPP5               :Add 1 to Mins*10
800CMP@#36              :If <> #36
810BNEPP1               :GOTO PP1
820JSRPP6               :Set Mins*10 to 0
830DEX                  :Point to Hours*1
840JSRPP5               :Add 1 to Hours*1
850JSRPP7               :GOSUB Check Hours
860BNEPP1               :If <> GOTO PP1
950JSRPP6               :Set Hours*1 to 0
960DEX                  :Point to Hours*10
970JSRPP5               :Add 1 to Hours*10
980CMP@#32              :If <> #32
990BNEPP1               :GOTO PP1
1000JSRPP6              :Set Hours*10 to 0
1010INX                 :Point to Hours*1
1020JSRPP5              :Add 1 to Hours*1
1030:PP1LDX@6           :Print display routine, Y
1040LDY@8               :points to VDU RAM locations, X
1050:PP2LDA#28F6,X      :points to clock data locations
1060STA#8017,Y          :
1070DEX                 :
1080DEY                 :
1090CPY@6               :
1100BNEPP2              :
1110LDA@CH":"           :Separate secs and mins with {:
1120STA#8017,Y          :} symbol
1130DEY                 :
1140:PP3LDA#28F6,X      :
1150STA#8017,Y          :
1160DEX                 :
1170DEY                 :
1180CPY@3               :
1190BNEPP3              :
1200LDA@CH":"           :Separate hours and mins
1210STA#8017,Y          :
1220:PP4LDA#28F6,X      :As all calculations use ASCII
1230STA#8017,X          :codes it is possible to store
1240DEX                 :results directly in VDU RAM
1250BNEPP4              :
1280LDY#28FF            :Restore Y register
```



```
1290LDX#28FE            :Restore X register
1300PLP                 :Restore Status register
1310PLA                 :Restore Accum
1320CLI                 :Clear Interrupt flag
1330RTI                 :Return from interrupt
1340:PP5LDA#28F6,X      :Add 1 to contents of location
1350CLC                 :pointed to by X
1360ADC@1               :
1370STA#28F6,X          :
1380RTS                 :
1390:PP6LDA@#30         :Store ASCII code for 0 in
1400STA#28F6,X          :location pointed to by X
1410RTS                 :
1420:PP7PHA             :Push Hours*1 data on stack
1430LDA#28F7            :Look at Hours*10 data
1440CMP@#30             :If = ASCII 0
1450BEQPP8              :GOTO PP8
1460PLA                 :Restore Hours*1 to Accum and
1470CMP@#33             :compare to #33, return with Z
1480RTS                 :flag set or clear
1490:PP8PLA             :Restore Hours*1 to Accum and
1500CMP@#3A             :compare to #3A, return with Z
1510RTS                 :flag set or clear
2000J;R.                :
```

LINES:-

510-530 Save X,Y and Status registers.

540-620 Clear 6522 interrupt flags and decrement 1/20th sec counter. Reset counter on reaching 0.

630-1020 Increment Secs, Mins and Hours as required.

1030-1250 Display time.

1280-1330 Restore all registers and return.

1340-1380 Add 1 to contents of location pointed to by X.

1390-1410 Store ASCII code for 0 in location pointed to by X.

1420-1510 Allow Hours*1 to increment to 10 if Hours*10 = 0 or increment to 3 if Hours*10 = 1. This provides for a 12 hour clock.

Note that on an interrupt the ATOM interpreter pushes the accumulator onto the stack before jumping to the address held in #204,#205.
#204,#205 should not be set by the ! instruction as !#204=#2800 would set #206,#207 to 0 when they should contain the command line address.

Note also the use of a string to hold the clock start time and the necessity to enter 0's before single numbers.

Read/Data/Restore

This programme adds the useful instructions READ, DATA and RESTORE to ATOM BASIC. To execute !#202 must be set to contain address of this programme.

PAGE 0 LOCATIONS USED:-
#80,#81 Current address of DATA.
#82,#83 Current address of STRING.
#84 Used as flag to indicate DATA statement found.
SUBROUTINES USED:-
#C9D8

ENTER IN TEXT AREA: Any
TEXT: 1.4K
VDU: -
M/C: .23K
EXECUTION: In BASIC if !#202=#xx00

```
5REM READ/DATA/RESTORE          :
10DIMCC27,S15                   :
20F.N=0TO27;P=#3900;CCN=P;N.    :
30F.N=1TO2;P=#3900;GOS.a;       :
S=P+1;N.                        :
50$S="RESTOREDATAREAD"          :Store $S immediately after M/C
60E.                            :
500a[                           :
520:CC0                         :
540LDX@#FF                      :
560JSRCC2                       :See if word being interpreted
580CPX@7                        :is RESTORE, if X=7 it is so
600BEQCC5                       :GOTO CC5
620LDX@6                        :Point X to next word
640JSRCC2                       :Is word DATA
660CPX@#B                       :If X=#B it is so
680BEQCC6                       :GOTO CC6
700LDX@#A                       :Point X at next word
720JSRCC2                       :Is word READ
740CPX@#F                       :If X=#F it is so
760BEQCC10                      :GOTO CC10
780:CC1JMP#C9D8                 :Jump to standard ERROR routine
800:CC2LDY@0                    :Comparison subroutine
820:CC3INX                      :
840:CC4INY                      :
850LDA(#5),Y                    :
860CMP@#20                      :
870BEQCC4                       :
880CMP S,X                      :
900BEQCC3                       :Returns with X pointing at the
920RTS                          :1st unequal character
```



```
940:CC5JSRCC23                  :
960:CC6JSRCC8                   :
980:CC7TSX                      :Transfer Stack pointer to X
1000INX                         :Point X at
1020INX                         :L.B return address
1040LDA@#F6                     :Store new L.B
1060STA#100,X                   :
1080INX                         :Point X at H.B
1100LDA@#C4                     :Store new H.B
1140STA#100,X                   :
1160LDA(#5),Y                   :
1180RTI                         :
1200:CC8LDY@1                   :Subroutine which returns with
1220:CC9INY                     :Y pointing at #D
1240LDA(#5),Y                   :
1260CMP@#D                      :
1280BNECC9                      :
1300RTS                         :
1320:CC10CMP@CH"$"              :Look for "$" after READ
1340BNECC1                      :
1360INY                         :Point Y at next character
1380LDA(#5),Y                   :Get character
1400SEC                         :Subtract #40 to give
1420SBC@#40                     :1 to 26 offset
1440TAX                         :Transfer result to X
1460LDA#321,X                   :Get L.B of string variable
1480STA#82                      :Store in #82
1500LDA#33C,X                   :Get H.B of string variable
1520STA#83                      :Store in #83
1540LDA#84                      :See if DATA has been found
1560BNECC11                     :If so GOTO CC11
1580JSRCC24                     :GOSUB Find DATA
1600:CC11LDY@#FF                :Store DATA statement at string
1620:CC12INY                    :variable address
1640LDA(#80),Y                  :Loop until DATA delimiter {,}
1660CMP@CH","                   :or end of line found
1680BEQCC13                     :
1700CMP@#D                      :
1720BEQCC14                     :
1740STA(#82),Y                  :
1760JMPCC12                     :
1780:CC13LDA@#D                 :Store #D at end of DATA, set
1800STA(#82),Y                  :#80,#81 as address of next
1820JSRCC17                     :DATA
1840JMPCC6                      :
1860:CC14LDA@#D                 :Store #D at end of DATA, see
1880STA(#82),Y                  :if next line is DATA and reset
1900INY                         :address in #80,#81, if not
1920INY                         :find 1st DATA line and reset
1940JSRCC21                     :#80,#81 to that address
1960CPX@#B                      :
1980BNECC15                     :
```

```
2000JSRCC18                 :
2020JMPCC6                  :
2040:CC15JSRCC24            :
2060JMPCC6                  :
2080:CC16INY                :Add Y to #80,#81 routine
2100:CC17INY                :Note various entry points
2120:CC18TYA                :
2140CLC                     :
2160ADC#80                  :
2180STA#80                  :
2200BCCCC19                 :
2220INC#81                  :
2240:CC19RTS                :
2260:CC20LDY@#FF            :
2280:CC21LDX@6              :Check if statement is DATA
2300:CC22INX                :
2320:CC23INY                :
2340LDA(#80),Y              :
2360CMP@#20                 :
2380BEQCC23                 :
2400CMP S,X                 :
2420BEQCC22                 :
2440RTS                     :
2460:CC24INC#84             :Set #84 >0 to indicate DATA
2480LDA#12                  :found
2500STA#81                  :Load #80,#81 with address of
2520LDA@0                   :Text area
2540STA#80                  :
2580:CC25LDY@#FF            :
2600:CC26INY                :Find start of Nth line
2620LDA(#80),Y              :
2640CMP@#D                  :
2660BNECC26                 :
2680INY                     :
2700LDA(#80),Y              :Check for end of text
2720BPLCC27                 :
2740JMPCC1                  :If found exit to ERROR
2760:CC27JSRCC16            :Move Y past line number,
2780JSRCC20                 :adjust #80,#81, see if 1st
2800CPX@#B                  :statement in line is DATA
2840BNECC26                 :If not loop for next line and
2860JSRCC18                 :do until DATA or end of text
2900RTS                     :is found
50001;R.                    :
```

LINES:-

520-780 Check statement to see if RESTORE,DATA or READ. If not jump to standard error routine.

800-920 Comparison subroutine. Returns with X containing number of valid comparisons plus 1.

980-1180 Store new return address in stack and return to that address.

1200-1300 Find end of line subroutine.

1320-1340 Check for "$" character after READ.

1360-1520 Find which string variable, get string address and transfer address to #82,#83.

1540-1580 See if line containing DATA statement has been found. Find if not.

1600-2020 Read data into address in #82,#83. When delimiter character or end of line found, store #D after read string and adjust data address.

2080-2240 Adds Y to #80,#81.

2260-2440 Returns with X=#B if DATA statement found.

2460-2860 Looks through programme lines for the first occurrence of DATA statement. Returns with #80,#81 containing address of first character after DATA statement. If DATA statement not found in programme standard error rountine is used.

The new return address #C4F6 ensures that BASIC FOR...NEXT and DO...UNTIL loops are kept intact. On entry to CC7 Y always points to the last character in the line, so Line 1160 always loads the accumulator with #D before returning.

The action taken by DIM in BASIC is to allow space for the specified string and to store the address of that space in the low two bytes of the specified integer variable. #322 to #33B hold the low bytes and #33D to #356 hold the second bytes of the integer variables A to Z.
As A to Z are coded #41 to #5A, it is a simple matter to convert the codes to an offset from #321 and #33C.
e.g. If $B address is #341C, this would be stored low byte (#1C) in #323 and high byte (#34) in #33E. The code for B is #42, subtract #40 and transfer result to X. X=2 so LDA#321,X would load the accumulator with the contents of #323 which is #1C and LDA#33C,X would fetch #34.

Note the subroutine CC16 which adds Y to the contents of #80, #81. By using three entry points (CC16,17 and 18) the versatility of this routine is considerably enhanced.

Cat Store

If you use long audio cassettes it can be a chore to catalogue the full tape, especially as the *CAT routine prints out all the blocks. This routine will catalogue a full tape and only display one name for each programme. It also stores those names at #9000 onwards so that they can be accessed after the tape has finished. The bell is rung whenever a new name is encountered.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of current name.
#82,#83 Address of next free space.
#84,#85 Start address of storage area.
#86 Number of names in store.
SUBROUTINES USED:-
#FB83 #FBEE #FC38 #FD1C #FD69 #FE52
#FE71 #FFED

ENTER IN TEXT AREA: #29
TEXT: 1.2K
VDU: .5K
M/C: .2K
EXECUTION: RUN

```
5REM *CAT STORE / DISPLAY     :
10DIMPP18                     :
20F.N=0TO18;P=#3200;PPN=P;N.  :
30F.N=1TO2;P=#3200;GOS.a;N.   :
40LI.#3200                    :
50E.                          :
500a[                         :
520:PP0LDY@8                  :
540:PP1LDA@0                  :
550STA#7F,Y                   :
560STA#9000,Y                 :
570DEY                        :
580BNEPP1                     :
590STA#321                    :Set field width to 0
620LDA@#90                    :Set up H.Bs
630STA#81                     :
640STA#83                     :
650STA#85                     :
660JSR#FD69                   :Clear VDU
700SEC                        :Print PLAY TAPE
710JSR#FC38                   :
720LDX@#78                    :Delay
740JSR#FB83                   :
760:PP2JSR#FE71               :Scan K/B
770CPY@#FF                    :If key pressed
```



```
780BNEPP6                     :GOTO PP6
840LDX@2                      :
850:PP3JSR#FBEE               :Get byte from tape
860CMP@#2A                    :Is it #2A
880BNEPP2                     :If not GOTO PP2
900DEX                        :Loop until 2 #2A codes are
920BNEPP3                     :obtained from tape
950LDY@0                      :
960:PP4JSR#FBEE               :Get next byte from tape
980CMP@#2A                    :Loop until not #2A
1000BEQPP4                    :
1020STA#21C,Y                 :Store in buffer
1040INY                       :and loop
1060CMP@#D                    :until #D
1080BNEPP4                    :is found
1100LDY@#FF                   :
1120:PP5INY                   :
1140LDA#21C,Y                 :Compare name in buffer with
1160CMP(#80),Y                :current name
1180BNEPP7                    :If new name GOTO PP7
1200CMP@#D                    :If same GOSUB PP17
1220BNEPP5                    :
1230JSRPP17                   :
1240JMPPP2                    :and then loop
1280:PP6RTS                   :
1290:PP7LDA#86                :If ?#86=0 then 1st name
1300BEQPP8                    :GOTO PP8
1320LDA#82                    :Load #80,#81 with
1340STA#80                    :next free space address
1360LDA#83                    :
1380STA#81                    :
1400:PP8LDY@#FF               :
1410INC#86                    :Increment No of names counter
1420:PP9INY                   :Store
1440LDA#21C,Y                 :name in buffer
1460STA(#80),Y                :in main store
1480CMP@#D                    :
1500BNEPP9                    :
1520INY                       :Adjust #82,#83 to contain
1660TYA                       :address of 1st free space
1680CLC                       :
1700ADC#80                    :
1720STA#82                    :
1740BCCPP10                   :
1760INC#83                    :
1770:PP10LDY@0                :Store end of text marker
1780LDA@#FF                   :
1790STA(#82),Y                :
1800LDA@5                     :Sound
1810JSR#FD1C                  :bell
1820LDY@0                     :
1840:PP11LDA@12               :Clear screen
```

```
1860JSR#FE52                :
1880:PP12LDY@0              :Print data in name store to
1900LDA(#84),Y              :VDU
1920CMP@#FF                 :
1940BEQPP16                 :
1960:PP13LDA(#84),Y         :
1980CMP@#D                  :
2000BEQPP14                 :
2020JSR#FE52                :
2040INY                     :
2060JMPPP13                 :
2070:PP14INY                :
2380TYA                     :
2400CLC                     :
2420ADC#84                  :
2440STA#84                  :
2460BCCPP15                 :
2480INC#85                  :
2500:PP15JSR#FFED           :
2520JMPPP12                 :
2530:PP16LDA@0              :Reset #84,#85 to start address
2540STA#84                  :of main name store
2560LDA@#90                 :
2580STA#85                  :
2590JSRPP17                 :
2600JMPPP2                  :
2800:PP17JSR#FBEE           :Wait for end of data marker
2820CMP@#D                  :from tape
2840BNEPP17                 :
2860JSR#FBEE                :
2880CMP@#FF                 :
2900BNEPP17                 :
2920RTS                     :
5000];R.                    :
```

LINES:-

500-580 Clear page 0 locations and start of main store.
620-650 Set up high bytes.
660-740 Clear VDU, print PLAY TAPE and delay.
760-780 Scan keyboard.
840-920 Get bytes from tape until 2 #2A codes have been obtained.
950-1080 Store subsequent bytes in buffer.
1100-1280 Compare current name with name in buffer.
1290-1380 Checks for 1st name,transfers free space address.

1770-1810 Stores end of text marker and sounds bell.
1820-2520 Prints names to VDU.
2530-2600 Resets #84,#85 to start of data address.
2800-2920 Waits for end of text marker from tape.

Direct Hex

This programme allows the entry of hexadecimal numbers without the prefix "#". 8 numbers are entered and the bits set or reset are displayed in an 8 x 8 matrix. This programme is used to enter known character codes for eventual use with MODE4. It can be used for the design of new characters but it is tedious!

PAGE 0 LOCATIONS USED:-
#80,#81 Address of current code.
#82 Temporary store for Y.
SUBROUTINES USED:-
#F802 #F87E #FD69 #FE52 #FE94

ENTER IN TEXT AREA: #29
TEXT: 1.4K
VDU: .5K
M/C: .25K
EXECUTION: RUN

```
5REM HEX DIRECT                 :
10DIMQQ16,R12,S12               :
20F.N=0TO16;DIMP-1;QQN=P;N.     :
30F.N=1TO2;P=#3200;GOS.a;N.     :
40!#80=#38FF                    :
50$R="O.K?"                     :
60$S="FINISHED?"                :
70LI.#3200                      :
80E.                            :
500a[                           :
510:QQ0JSR#FD69                 :Clear VDU
515JSRQQ14                      :GOSUB VDU display
520:QQ1LDY@8                    :
530LDX@0                        :
540:QQ2LDA@#3F                  :Output "?"
550JSR#FE52                     :to VDU
560JSR#FE94                     :Get character from K/B
570JSR#F87E                     :Convert to binary low 4 bits
580ASLA                         :Move low 4 bits to high 4 bits
590ASLA                         :
600ASLA                         :
610ASLA                         :
620STA(#80),Y                   :Store result
630JSR#FE94                     :Get next character
640JSR#F87E                     :Convert to binary low 4 bits
650CLC                          :
660ADC(#80),Y                   :Add to previous result and
670STA(#80),Y                   :store
680STX#E0                       :Set cursor position
```

```
690INX                  :Point X at next column
700INX                  :position
710INX                  :
720LDA@#80              :Set VDU line
730STA#DE               :
740LDA(#80),Y           :Output # number as 2 ASCII
750JSR#F802             :numerals
760LDA@0                :Reset VDU line
770STA#DE               :
780STA#E0               :Reset cursor position
790DEY                  :Loop for 8 times
800BNEQQ2               :
808STY#E0               :Reset cursor position
810LDA@#E8              :Set new VDU line position
820STA#DE               :
830LDY@8                :Point Y at start of codes
840:QQ3 LDA(#80),Y      :Get 1st code
850STY#82               :Save Y
860LDY@8                :Print codes as graphic blocks
870:QQ4CLC              :in 8 x 8 matrix.
880RORA                 :Rotate low bit into Carry
890PHA                  :Save remainder
900BCSQQ5               :If Carry set print white block
910LDA@#20              :{#7F} to VDU, if Carry clear
920JMPQQ6               :print black block {#20} to VDU
930:QQ5LDA@#7F          :
940:QQ6STA(#DE),Y       :
950PLA                  :Reload remainder and loop for
960DEY                  :all 8 bits
970BNEQQ4               :
990STY#E0               :Adjust VDU line and column
1000LDA#DE              :
1010CLC                 :
1020ADC@#20             :
1030STA#DE              :
1040BCCQQ7              :
1050INC#DF              :
1060:QQ7LDY#82          :Get Y pointer to data and loop
1070DEY                 :for 8 codes
1080BNEQQ3              :
1090LDA@0               :Adjust VDU line and column
1100STA#DE              :
1110STA#E0              :
1120LDA@#80             :
1130STA#DF              :
1140:QQ8LDAR,Y          :Print $R
1150CMP@#D              :
1160BEQQQ9              :
1170JSR#FE52            :
1180INY                 :
1190BNEQQ8              :
1200:QQ9JSR#FE94        :Get character from K/B
```



```
1210CMP@#59             :Is it "Y"
1220BNEQQ11             :If not GOTO QQ11
1230LDY@0               :
1240JSR#FD69            :Clear VDU
1250:QQ10LDAS,Y         :Print $S
1260CMP@#D              :
1265BEQQQ13             :
1270JSR#FE52            :
1280INY                 :
1290BNEQQ10             :
1300:QQ13JSR#FE94       :Get character from K/B
1310CMP@#59             :Is it "Y"
1320BEQQQ12             :If so GOTO QQ12
1330LDA#80              :Adjust #80,#81
1340CLC                 :to contain
1350ADC@8               :address of
1360STA#80              :next free location
1370BCCQQ11             :
1380INC#81              :
1390:QQ11JMPQQ0         :Loop for next input
1400:QQ12RTS            :
1500:QQ14LDA@#60        :Print VDU display
1510STA#DE              :Set cursor to 4th VDU line
1520LDA@1               :Reset VDU column
1530STA#E0              :
1540LDA@#30             :Load Accum with code for 0
1550LDY@8               :
1560:QQ15PHA            :
1570JSR#FE52            :Output ASCII code to VDU
1580INC#E0              :Move cursor
1590INC#E0              :
1600PLA                 :
1610CLC                 :Add 1 to ASCII code
1622ADC@1               :
1630DEY                 :Loop for 8 times
1640BNEQQ15             :
1645TYA                 :
1646TAX                 :
1647STY#E0              :Reset cursor
1650STY#E1              :Turn cursor off
1650LDA@#E5             :Set cursor position Line 8
1660STA#DE              :Column 6 on VDU
1670LDY@8               :
1680LDA@#30             :Load Accum with ASCII code for
1690:QQ16PHA            :0
1700JSR#FE52            :Print ASCII code
1705STX#E0              :Reset cursor
1710LDA#DE              :Move cursor down 1 line
1720CLC                 :
1730ADC@#20             :
1740STA#DE              :
1750BCCQQ17             :
```

```
1760INC#DF                      :
1770:QQ17PLA                    :
1780CLC                         :
1790ADC@1                       :Add 1 to ASCII code
1800DEY                         :Loop for
1810BNEQQ16                     :8 times
1820LDA@30                      :Home cursor
1830JSR#FE52                    :
1840RTS                         :
3000];R.                        :
```

```
LINES:-
510-550     Clear VDU and print display.
560-670     Get two keyboard entries, convert to hexadecimal
and store.
680-780     Output # number to specific VDU location.
790-800     Loop for 8 entries.
808-830     Adjust cursor position.
840-1080    Print 8 bit x 8 byte display.
1090-1130   Adjust cursor position.
1140-1190   Print $R.
1200-1240   Wait for key to be pressed and see if "Y".
1250-1290   Print $S.
1300-1320   Wait for key to be pressed and see if "Y".
1330-1390   Adjust #80,#81 to contain address of next free
location.
1500-1840   Print VDU display.

Lines 560 to 670 convert two ASCII codes received from the
keyboard into one hexadecimal number.
If keys F and 4 are pressed then:
```

	Accum	76543210	Carry
Line 560	46	01000110	-
Line 570	0F	00001111	-
Line 580	1E	00011110	0
Line 590	3C	00111100	0
Line 600	78	01111000	0
Line 610	F0	11110000	0
Line 630	34	00110100	-
Line 640	04	00000100	-
Line 670	F4	11110100	-

KeyScan

This programme allows specific keys to be scanned. It must have some uses!

```
PAGE 0 LOCATIONS USED:-
#80         Temporary output store.
SUBROUTINES USED:-
#FB8A


ENTER IN TEXT AREA: ANY
TEXT: .4K
VDU: -
M/C: .05K
EXECUTION: RUN
```

```
5REM SINGLE KEY SCAN            :
10DIMJJ3                        :
20F.N=0TO3;P=#2800;JJN=P;N.     :
30F.N=1TO2;P=#2800;GOS.a;N.     :
40IN."KEYBOARD ROW 1-6"Z        :
50IN."KEYBOARD COLUMN 0-9"Y     :
60LI.#2800                      :
70E.                            :
500aI                           :
510:JJ0LDX#33B                  :Load X with L.B of Z
520LDA@i                        :
530:JJ1DEX                      :Move bit to position
540BEQJJ2                       :pointed to by X
550ASLA                         :
560JMPJJ1                       :
570:JJ2EOR@#FF                  :Set bit to 0
580STA#33B                      :Store back in L.B of Z
590:JJ3LDA#B000                 :Save graphics mode
595PHA                          :on stack
600CLC                          :Add keyboard column
610ADC#33A                      :
620STA#B000                     :Drive keyboard
630LDA#B001                     :Get Keyboard output
640STA#80                       :and store in #80
650PLA                          :Reset graphics mode
660STA#B000                     :
670LDA#80                       :Compare output with keyboard
680CMP#33B                      :row
690BNEJJ3                       :If not equal loop
700JSR#FB8A                     :Wait
710RTS                          :Exit
2000];R.                        :
```

Note the use of EOR instruction to invert bits.

TransferText

A simple BASIC programme which transfers specific lines from one text area to another.

```
5REM PROGRAM TRANSFER          :
10Z=0;Y=0                      :
20IN."FROM TEXT AREA"A         :
30IN."TO TEXT AREA"B           :
40IN."START LINE NUMBER"C      :
50IN."END LINE NUMBER"D        :
60A=A+1                        :
70E=C&#FF                      :
80F=C/256                      :
90G=D&#FF                      :
100H=D/256                     :
110DO                          :
120IF Z<>0 G.140               :
130IF ?A=F IF ?(A+1)=E Z=A     :
140IF Y<>0 G.190               :
150IF ?A=H IF ?(A+1)=G Y=A     :
160A=A+1                       :
170U.?A=#FF                    :
190IF Z=0 G.a                  :
200IF Y=0 G.a                  :
210?B=13                       :
220B=B+1                       :
230DO                          :
240?B=?Z                       :
250B=B+1                       :
260Z=Z+1                       :
270U.Z=Y                       :
280DO                          :
290?B=?Y                       :
300B=B+1                       :
310Y=Y+1                       :
320U.?Y=13                     :
330B=B+1                       :
340?B=13                       :
350B=B+1                       :
360?B=#FF                      :
370E.                          :
380a P."ERROR --"'             :
390P."LINE NUMBER  "$7         :
400@=0                         :
410IF Z=0 P.C                  :
420IF Y=0 P.D                  :
430P." NOT FOUND"              :
440E.                          :
```


Flashing Lines

This programme will flash a VDU line or part of that line.

```
PAGE O LOCATIONS:-
#80,#81     Address of 1st position to be flashed.
#82         VDU line position number.
#83         VDU column position number.
#84         Pointer to number of columns.
#85         Duration of flash.
SUBROUTINES USED:-
#FB83


ENTER IN TEXT AREA: Any
TEXT: .8K
VDU: .5K
M/C: .12K
EXECUTION: RUN

5REM FLASHING LINES                :
10DIM FF9                          :
20F.N=0 TO 9;DIMP-1;FFN=P;N.       :
30F.N=1 TO 2;P=#2800;GOS.a;N.      :
40@=0                              :
50IN."LINE NO 1-16"L               :
60IN."START NO 1-32"S              :
70P."FINISH NO "S                  :
80IN."-32"F                        :
90IN."DURATION IN 30TH SECS"D      :
100E.                              :
500a[                              :
510:FFOLDA@#80                     :Set up #80,#81 to contain VDU
520STA#81                          :address
530LDA@0                           :
540STA#80                          :
550LDA#32D                         :Get L.B of L
560STA#82                          :Store in #82
570LDA#334                         :Get L.B of S
580STA#83                          :Store in #83
590LDA#325                         :Get L.B of D
600STA#85                          :Store in #85
610LDA#327                         :Get L.B of F
620SBC#83                          :Calculate number of columns to
630STA#84                          :be flashed
635INC#84                          :Store in #84
636INC#84                          :
640:FF1LDX#82                      :Calculate line address by
650:FF2DEX                         :adding #20 for each line
660BEQFF4                          :
670LDA#80                          :
```

```
680CLC                          :
690ADC@#20                      :
700STA#80                       :
710BCCFF3                       :
720INC#81                       :
730:FF3JMPFF2                   :
740:FF4LDY#83                   :Calculate column position of
750DEY                          :1st character
760TYA                          :
770CLC                          :
780ADC#80                       :
790STA#80                       :
800LDY@0                        :
810:FF5LDA(#80),Y               :Store characters in temporary
820STA#2880,Y                   :store
830INY                          :
840CPY#84                       :Check for no of columns to be
850BNEFF5                       :stored
860:FF6LDX#85                   :Delay by ?#85/60 secs
870JSR#FB83                     :
880LDY@0                        :
890:FF7LDA@#20                  :Store spaces in place of
900STA(#80),Y                   :characters
910INY                          :
920CPY#84                       :
930BNEFF7                       :
940BIT#B001                     :See if CTRL pressed
950BVCFF9                       :
960LDX#85                       :Delay
970JSR#FB83                     :
980LDY@0                        :
990:FF8LDA#2880,Y               :Restore characters to VDU
1000STA(#80),Y                  :
1010INY                         :
1020CPY#84                      :
1030BNEFF8                      :
1040BIT#B001                    :See if CTRL pressed
1050BVSFF6                      :If not loop
1060:FF9RTS                     :
2000];R.                        :
```

LINES:-

510-636	Set up page 0 locations.
640-720	Calculate VDU line and store.
740-790	Calculate VDU column and store.
800-870	Store characters temporarily and delay.
880-930	Store spaces at character positions.
940-970	See if CTRL pressed and delay.
980-1050	Restore characters to VDU and see if CTRL pressed.

Long Strings

A programme whichs allows strings of up to 256 characters to be entered directly into memory.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of 1st free location.
SUBROUTINES USED:-
#FE52 #FE94

ENTER IN TEXT AREA: Any
TEXT: .5K
VDU: .5K
M/C: .05K
EXECUTION: RUN

```
5REM STORE STRINGS              :
10DIM BB5                       :
20F.N=0TO5;P=#3B00;BBN=P;N.     :
30F.N=1TO2;P=#3B00;GOS.a;N.     :
40P.$12'"         STRING        :
STORE"'                         :
50IN."STORAGE SPACE START"Z     :
60IN."NUMBER OF STRINGS"W       :
70!#80=Z                        :
80F.N=1 TO W                    :
90LI.BB0                        :
100N.                           :
110E.                           :
500a[                           :
510:BB0LDA@#3F                  :Output "?" to VDU
520JSR#FE52                     :
530LDY@0                        :
540:BB1JSR#FE94                 :Get character from keyboard
550:BB2JSR#FE52                 :Output to VDU
560CMP@#7F                      :See if DELETE pressed
570BNEBB3                       :If not GOTO BB3
580DEY                          :
585CPY@#FF                      :
588BNEBB1                       :
590JMPBB0                       :
600:BB3STA(#80),Y               :Store character in memory
610CMP@#D                       :Check for #D
620BEQBB4                       :
640INY                          :Point Y at next location
650BNEBB1                       :and loop
660INC#81                       :If Y > #FF increment H.B
670LDA@#D                       :Load Accum with #D and loop so
680JMPBB2                       :exit takes place
690:BB4INY                      :Adjust #80,#81
```

```
700CLC                          :
710TYA                          :
720ADC#80                       :
730STA#80                       :
740BCCBB5                       :
750INC#81                       :
760:BB5LDY@0                    :
770LDA@#FF                      :Store end of text marker
780STA(#80),Y                   :
790RTS                          :
800];R.                         :
```

LINES:-
510-520 Output "?" to VDU.
540-590 Get character from keyboard, output to VDU and recognize DELETE. Do not allow Y to decrement past 0.
600-680 Store characters at address in #80,#81 until #D is input or 256 characters have been input. When 256 characters have been input cause exit by sending #D to output.
690-750 Adjust #80,#81.
760-790 Store end of text marker.

Note that the end of text marker is stored at the address in #80,#81. This is the address of the 1st free space, so the end of text marker is over-written on re-entry.

Line 70 transfers the 4 bytes of integer variable Z into #80, #81,#82,#83. Storage space should, therefore, be entered as a four digit hexadecimal number or in decimal if you can remember that number of digits!

Alfabeta Sort

This programme alphabetically sorts strings to the Nth character, where N is the length of the shortest of the two strings being compared. The temporary storage during the change routine is VDU memory, it can of course be any area.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of Nth string.
#82,#83 Address of Nth+1 string.
#84 Used as flag to indicate changes.
#85 Length of shortest string.
#90,#91 Address of string storage.

ENTER IN TEXT AREA: Any
TEXT: .8K
VDU: -
M/C: .14K
EXECUTION: LINK#xx00

```
5REM ALPHABETASORT              :
10DIMDD14                       :
20F.N=0TO14;P=#2800;DDN=P;N.    :
30F.N=1TO2;P=#2800;GOS.a;N.     :
40E.                            :
500a[                           :
510:DD0LDA#90                   :Transfer address
520STA#80                       :of strings
530LDA#91                       :to #80,#81
540STA#81                       :
550LDA@0                        :Clear change flag
560STA#84                       :
570:DD1LDY@0                    :
580:DD2LDA(#80),Y               :Find length of Nth string
590CMP@#D                       :
600BEQDD3                       :
610INY                          :
650JMPDD2                       :
660:DD3STY#85                   :Store length of string
670LDA#81                       :Set #82,#83 to contain address
680STA#83                       :of Nth+1 string
690INY                          :
700CLC                          :
710TYA                          :
720ADC#80                       :
730STA#82                       :
740BCCDD4                       :
750INC#83                       :
760:DD4LDY@0                    :
770LDA(#82),Y                   :Check for end of text
```

```
780CMP@#FF                  :
790BNEDD6                   :
800LDA#84                   :See if any changes have been
810BNEDD0                   :made
820:DD5RTS                  :
830:DD6INY                  :Find length of Nth+1 string
840LDA(#82),Y               :
850CMP@#D                   :
860BNEDD6                   :
870CPY#85                   :Compare to length of Nth
880BCSDD7                   :If less
890STY#85                   :store in #85
900:DD7LDY@0                :
910:DD8LDA(#80),Y           :Comparison routine
920CMP(#82),Y               :
930BNEDD10                  :If not equal GOTO DD10
940INY                      :If equal compare each
950CPY#85                   :character until all characters
960BCCDD8                   :in shortest string compared
965BEQDD8                   :
970:DD9LDA#82               :Transfer address in #82,#83 to
980STA#80                   :#80,#81
990LDA#83                   :
1000STA#81                  :
1010JMPDD1                  :
1020:DD10BCCDD9             :If less than GOTO DD9
1030INC#84                  :Set change flag
1040LDY@#FF                 :
1050:DD11INY                :Store Nth
1060LDA(#80),Y              :string in temporary
1070STA#8100,Y              :store
1080CMP@#D                  :
1090BNEDD11                 :
1100LDY@#FF                 :
1110:DD12INY                :Move Nth+1 string to address
1120LDA(#82),Y              :of Nth string
1130STA(#80),Y              :
1140CMP@#D                  :
1150BNEDD12                 :
1160LDX@#FF                 :
1170:DD13INX                :
1180INY                     :Get Nth string from temporary
1190LDA#8100,X              :store and store at address Y
1200STA(#80),Y              :points to
1210CMP@#D                  :
1220BNEDD13                 :
1240JMPDD1                  :Loop for next
1500];R.                    :
```

LINES:-

510-540 Set up #80,#81.

550-560 Clear change flag.

570-660 Find length of Nth string and store.

670-750 Adjust #82,#83 to contain address of Nth+1 string.

760-820 Look for end of text marker and when found check change flag.

830-890 Check for shortest string.

900-965 Compare each character of strings until all characters in shortest string have been compared.

970-1010 Transfer address.

1020-1030 Check for less than.

1040-1240 Change routine.

The change routine used in this programme is identical to the one used in RENUMBER. However the temporary store must be at least .5K to allow for strings of up to 256 characters.

The comparison routine first checks for equality . If equality is not found a branch to DD10 occurs and if more than is found the programme changes the order of the strings.

As many words can start with the same letters the equality routine ensures that the shortest string is sorted to the top. e.g. concussion and concuss would return as concuss followed by concussion.

Note the loading of Y and X with #FF during the change routine. As INY and INX are the first instructions in the loop, Y and X are set to 0 on entering the loop.

Design Letters

This programme allows the user to design characters for use with Graphics Mode 4. A 8 x 8 box is displayed with a flashing cursor, this cursor can be moved around the box by using the keyboard. F moves the cursor forward one space, B moves it back one space, U moves it up one line, D moves it down one line. COPY sets the cursor position and stops it flashing, DELETE clears a set position and T transfers the set positions into an 8 bit x 8 byte set of codes, stores those codes and displays the result at the top of the screen. E escapes from the programme.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of cursor.
#82 Cursor on/off data.
#83 Current line position of cursor.
#84 Current column position of cursor.
#86,#87 Address of box.
#88,#89 Address of character display.
#8A,#8B Address of code storage space.
#8C Temporary for Y.
#90-#97 Data for setting cursor.
#98-#9F Cursor set/reset store.
SUBROUTINES USED:-
#FB8A #FE71

ENTER IN TEXT AREA: #29
TEXT: 2.5K
VDU: 6K
M/C: .47K
EXECUTION: RUN

```
5REM DESIGN LETTERS            :
10DIMMM48                      :
20F.N=0TO48;P=#3400;MMN=P;N.   :
30F.N=1TO2;P=#3400;GOS.a;N.    :
40!#90=#10204080               :
50!#88=#36008102               :
60!#94=#01020408               :
70CLEAR4                       :
80LI.#3400                     :
90E.                           :
500aE                          :
510:MM0JSRMM37                 :Clear box and draw outline
600LDA@1                       :
620STA#83                      :
640STA#84                      :
660LDX@8                       :
680LDA@0                       :Clear cursor set/reset store
```



```
700:MM1STA#97,X                :
720DEX                         :
740BNEMM1                      :
760:MM2LDY@0                   :Flash cursor routine
770JSR#FB8A                    :
800JSRMM17                     :
840JSR#FB8A                    :
860JSRMM18                     :
940JSR#FE71                    :Scan keyboard
960CPY@#FF                     :if no key pressed
980BEQMM2                      :GOTO MM2
1000:MM3CPY@#35                :Has U been pressed
1020BEQMM4                     :
1040CPY@#34                    :Has T been pressed
1060BEQMM6                     :
1080CPY@#26                    :Has F been pressed
1100BEQMM7                     :
1120CPY@#22                    :Has B been pressed
1140BEQMM9                     :
1160CPY@#24                    :Has D been pressed
1180BEQMM11                    :
1200CPY@#F                     :Has DELETE been pressed
1220BEQMM13                    :
1240CPY@#E                     :Has COPY been pressed
1260BEQMM15                    :
1280CPY@#25                    :Has E been pressed
1300BNEMM2                     :
1320RTS                        :
1340:MM4JSRMM16                :Move cursor up routine, checks
1360BNEMM5                     :if cursor set, if not clears
1380JSRMM17                    :cursor, gets new cursor
1400:MM5JSRMM21                :position and prints cursor
1420JSRMM18                    :
1440JMPMM2                     :
1460:MM6JSRMM23                :Transfer routine
1480JMPMM0                     :
1500:MM7JSRMM16                :Move cursor forward routine,
1520BNEMM8                     :checks if cursor set, if not
1540JSRMM17                    :clears cursor, gets new
1560:MM8JSRMM27                :address and prints cursor
1580JSRMM18                    :
1600JMPMM2                     :
1620:MM9JSRMM16                :Move cursor backwards routine,
1640BNEMM10                    :actions as forwards routine
1644JSRMM17                    :
1680:MM10JSRMM30               :
1700JSRMM18                    :
1720JMPMM2                     :
1740:MM11JSRMM16               :Move cursor down routine,
1760BNEMM12                    :actions as up routine
1780JSRMM17                    :
1800:MM12JSRMM33               :
```

```
1820JSRMM18                :
1840JMPMM2                 :
1850:MM13JSRMM35           :Clear set cursor
1890JMPMM2                 :
1900:MM15JSRMM36           :Set cursor and
1910:MM14JSR#FE71          :stop flashing cursor
1920CPY@#FF                :
1930BEQMM14                :
1940JMPMM3                 :
1980:MM16LDY#83            :Check if cursor set routine
2000LDX#84                 :See notes
2020LDA#97,Y               :
2040AND#8F,X               :
2060RTS                    :
2080:MM17LDA@0             :Clear cursor pixel data
2100STA#82                 :
2120JSRMM19                :
2140RTS                    :
2160:MM18LDA@#FF           :Set cursor pixel data
2180STA#82                 :
2200JSRMM19                :
2220RTS                    :
2240:MM19LDA#83            :Add line number to #8E to give
2260CLC                    :cursor address H.B
2280ADC@#8E                :
2300STA#81                 :Store in #81
2320LDA#84                 :Add column number to #15 to
2340CLC                    :give cursor address L.B
2360ADC@#15                :
2380STA#80                 :Store in #80
2400LDX@8                  :
2420LDY@0                  :
2440:MM20LDA#82            :Load cursor pixel data and
2460STA(#80),Y             :print cursor. Cursor consists
2480LDA#80                 :of block 8 bits x 8 bytes.
2500CLC                    :Each byte has a #20 separation
2520ADC@#20                :which is 1 Mode 4 VDU line.
2540STA#80                 :
2560DEX                    :
2580BNEMM20                :
2600RTS                    :
2620:MM21LDA#83            :Subtract 1 from current line
2640SEC                    :number, if this leaves 0
2660SBC@1                  :return, #83 is never equal to
2680BEQMM22                :0, otherwise store new line
2700STA#83                 :number and return
2720:MM22RTS               :
2740:MM23LDY@0             :Display character routine, Y
2760LDX@0                  :always equals 0 as character
2780:MM24LDA#98,X          :is made up the same way as the
2790STA(#88),Y             :cursor
2800JSRMM47                :GOSUB Store codes routine
```



```
2810LDY@0                  :
2820LDA#88                 :Adjust #88,#89 to contain
2840CLC                    :address of next VDU line
2860ADC@#20                :
2880STA#88                 :
2890INX                    :
2900CPX@8                  :Loop for 8 codes
2920BNEMM24                :
2940LDA#88                 :Adjust #88,#89 to contain
2960CLC                    :address of next VDU display
2980ADC@1                  :position
3000CMP@#1F                :Check for end of line
3020BEQMM26                :
3040:MM25STA#88            :
3060RTS                    :
3080:MM26INC#89            :Set new line address
3100LDA@2                  :
3120JMPMM25                :
3140:MM27LDA#84            :Add 1 to cursor column number,
3160CLC                    :if result >8 GOTO MM 28 to
3180ADC@1                  :move cursor down by 1 line,
3190CMP@9                  :otherwise store new position
3200BCSMM28                :and return
3220STA#84                 :
3240RTS                    :
3260:MM28JSRMM33           :GOSUB Move down 1 line
3280CMP@9                  :If Accum returns with 9,
3300BEQMM29                :cursor is extreme bottom left
3320LDA@1                  :of box so no movement possible,
3340STA#84                 :otherwise set column to 1 and
3360:MM29RTS               :return
3380:MM30LDA#84            :Move cursor backwards, same
3400SEC                    :actions as MM27 except cursor
3420SBC@1                  :will be moved up 1 line when
3440BEQMM31                :column number =0
3460STA#84                 :
3480RTS                    :
3500:MM31JSRMM21           :Move cursor up 1 line unless
3520BEQMM32                :in extreme top left position
3540LDA@8                  :
3560STA#84                 :
3580:MM32RTS               :
3600:MM33LDA#83            :Move cursor down 1 line, if =
3620CLC                    :9 cursor on bottom line so
3640ADC@1                  :return without further action
3660CMP@9                  :
3680BCSMM34                :
3700STA#83                 :
3720:MM34RTS               :
3740:MM35LDY#83            :Clear set cursor
3760LDX#84                 :
3780LDA#97,Y               :See notes
```

```
3800EOR#8F,X            :
3820STA#97,Y            :
3840RTS                 :
3860:MM36LDY#83         :Set cursor - see notes
3880LDX#94              :
3900LDA#97,Y            :
3920ORA#8F,X            :
3940STA#97,Y            :
3960RTS                 :
4000:MM37JSRMM42        :Draw box routine
4010JSRMM43             :Clear box
4020LDY@#8              :Print top line of box
4040:MM38LDA@#FF        :Set pixel data
4060STA(#86),Y          :
4080DEY                 :Print for 8 successive bytes
4100BNEMM38             :
4110LDA#87              :Calculate address of bottom
4120CLC                 :line and store in #86,#87
4130ADC@#9              :
4140STA#87              :
4150LDA#86              :
4160CLC                 :The L.B calculation will set
4170ADC@#20             :the carry bit but it is
4180STA#86              :ignored
4200LDY@#8              :
4220LDA@#FF             :
4240:MM39STA(#86),Y     :Print bottom line
4260DEY                 :
4280BNEMM39             :
4300JSRMM42             :Reset #86,#87 to box start
4320LDX@#42             :address
4340:MM40LDY@0          :Print left side of box by
4360LDA@1               :setting extreme right pixel of
4380STA(#86),Y          :VDU column 22
4400LDY@#9              :
4420LDA@#80             :Print right side of box by
4440STA(#86),Y          :setting extreme left pixel of
4460LDA#86              :VDU column 31
4480CLC                 :Calculate address of next VDU
4500ADC@#20             :line
4520STA#86              :
4540BCCMM41             :
4560INC#87              :
4580:MM41DEX            :Loop for 66 lines
4600BNEMM40             :
4620RTS                 :
4640:MM42LDA@#8E        :Set #86,#87 to contain start
4660STA#87              :address of box
4680LDA@#F5             :
4700STA#86              :
4720RTS                 :
4740:MM43LDX@#42        :Clear box routine
```



```
4760:MM44LDY@8          :Set every pixel within box to
4780:MM45LDA@0          :0
4800STA(#86),Y          :
4820DEY                 :
4840BNEMM45             :
4860LDA#86              :
4880CLC                 :
4900ADC@#20             :
4920STA#86              :
4940BCCMM46             :
4960INC#87              :
4980:MM46DEX            :
5000BNEMM44             :
5020JSRMM42             :
5040RTS                 :
5060:MM47LDY#8C         :Store codes in memory routine
5070STA(#8A),Y          :
5080INY                 :
5090BNEMM48             :
5100INC#8B              :
5120:MM48STY#8C         :
5140RTS                 :
6000];R.                :
```

LINES:-

510-740 Clear box and print outline, clear page 0 locations.
760-860 Flash cursor.
940-1320 Scan keyboard and interpret any key pressed.
1340-2060 Execute key press.
2080-2220 Cursor on/off execution.
2240-2380 Calculate address of current cursor position.
2400-2600 Print cursor.
2620-2720 Move cursor position up 1 line, if possible.
2740-2920 Display 8 codes as one character.
2940-3120 Adjust character display position, move address to next line if necessary.
3140-3360 Move cursor position 1 column forwards, move to next line if necessary or take no action if bottom line extreme right.
3380-3580 Move cursor position 1 column backwards, move to next line if possible or take no action if top line extreme left.
3600-3720 Move cursor 1 line down, if possible.
3740-3840 Delete cursor setting.
3860-3960 Set cursor and stop flashing.
4000-5040 Draw box.
5060-5140 Store codes at address in #8A,#8B.

Most characters are displayed as a 8 x 8 pixel matrix. When using graphics Mode4 this matrix is 1 byte x 8 bytes. Each bit set in this matrix will be displayed as a white dot.

Mode 4 graphics produce a VDU display of 32 bytes x 192 lines, the separation between lines being #20. While characters can be displayed at any VDU address, it is advisable to use an address which has a low byte equal to zero as 7 additions of #20 will not set the Carry flag and therefore no instructions are needed to increment the high byte.

The actions taken to set the cursor, to reset the cursor and to check whether the cursor is set are as follows. The cursor position is line 1, column 8 and columns 2 and 3 are already set.

```
Line:
3860      Load Y with contents of #83, Y=1.
3880      Load X with contents of #84, X=8.
3900      Load Accum with contents of #98 (#97,Y), (#98
contains line 1 data) Accum = #60.
3920      OR Accum with contents of #97 (#8F,X), #01.
3940      Store result (#61) in #98 (#97,Y).
```

The OR sets the bit, EOR clears the bit and AND checks the bit. e.g.

```
OR Action.          Bits 76543210
Accum                    01100000
Memory                   00000001
Result                   01100001

EOR Action.
Accum                    01100001
Memory                   00000001
Result                   01100000

AND action.
Accum if bit set         01100001
Memory                   00000001
Result   <>0             00000001

Accum if bit clear       01100000
Memory                   00000001
Result   =0              00000000
```

Note that #98 - #9F contain the data for each line and that each bit is equivalent to the column, bit 7 being column 1.

Menu Program

This programme which is assembled into the last 1K of graphics mode 4 RAM stores up to 12 programme titles and their execution addresses. New entries can be made when this programme is called or the stored list of titles can be printed out. Any programme can then be executed by direct entry of its title, assuming it is in memory. Machine code and BASIC programmes are recognized and appropriate action taken.

```
PAGE 0 LOCATIONS USED:-
#80,#81     Temporary storage of programme address.
#82         Temporary storage for registers.
#83,#84     Used during calculation of address.
#85         Number of titles.
SUBROUTINES USED:-
#C2F2     #C589     #CD0F     #F7FD     #F87E     #FB8A
#FD69     #FE52     #FE94     #FFED


ENTER IN TEXT AREA: Any
TEXT: 2.6K
VDU: .5K
M/C: .44K
EXECUTION: LINK#9400
```

```
5REM MENU  PROGRAM                :
10DIM LL22,KK7,TT8                :
12F.N=0 TO 22;DIMP-1;LLN=P;       :
KKN=P;TTN=P;N.                    :
20S=#9670                         :
30$S="ENTER OR LIST?"             :
40TT0=S                           :
50S=S+LENS+1                      :
60$S="ADD TO MENU?"               :
70TT1=S                           :
80S=S+LENS+1                      :
90$S="PROGRAMME NAME?"            :
100TT2=S                          :
110S=S+LENS+1                     :
120$S="ADDRESS? #"                :
130TT3=S                          :
140S=S+LENS+1                     :
150$S="MORE?"                     :
160TT4=S                          :
170S=S+LENS+1                     :
180$S="menu----"                  :
190TT5=S                          :
200S=S+LENS+1                     :
210$S="ERROR ON NAME"             :
```

```
220TT6=S                        :
230S=S+LENS+1                   :
240$S="LIST FULL"               :
250TT7=S                        :
260S=S+LENS+1                   :
270$S="NAME TOO LONG"           :
280TT8=S                        :
320F.N=1TO2;P=#9400;GOS.a;N.    :
330E.                           :
500a[                           :
520:LL0JSR#FD69                 :Clear VDU
530LDY@0                        :
540LDX@#6F                      :Point X at title store
550STY#85                       :Clear counter
560LDA@#96                      :Store H.B of store address
580STA#81                       :in #81
600LDA@TT0&#FF                  :Store L.B of message string
610STA#80                       :1 in #80
620JSRKK0                       :GOSUB Print
630JSR#FE94                     :Wait for key to be pressed
640CMP@CH"E"                    :Was it E
650BNELL5                       :If not GOTO LL5
660LDA@TT1&#FF                  :Print message string 2
670STA#80                       :
680JSRKK0                       :
690JSR#FE94                     :Wait for key to be pressed
700CMP@CH"Y"                    :Was it Y
710BNELL22                      :If not GOTO LL22
720JSRKK4                       :Gosub Find end of data
730:LL22LDA@TT2&#FF             :Print message string 3
740STA#80                       :
760JSRKK0                       :
780:LL2LDY@0                    :
785JSR#CD14                     :Input Title
790CPY@#0C                      :Check to see if title exceeds
800BCCLL20                      :12 characters
810LDA@TT8&#FF                  :Print message string 9
820STA#80                       :
830JSRKK1                       :
840JMPLL22                      :Loop for next try
850:LL20LDY@0                   :
860:LL21LDA#100,Y               :Store new title in title store
870STA#9600,X                   :pointed to by X
880CMP@#D                       :
890BEQLL6                       :
900DEX                          :
905INY                          :
910JMPLL21                      :
940:LL3LDA@TT7&#FF              :Print message string 8
960STA#80                       :
965JSR#FFED                     :
980JSRKK1                       :
```



```
984LDX@#A0                      :Delay for 2.5 secs
985JSR#FB83                     :
990JMPLL8                       :
1000:LL4RTS                     :
1020:LL5JMPLL8                  :Stepping stone
1040:LL6LDA@TT3&#FF             :Print message string 4
1060STA#80                      :
1080JSRKK1                      :
1100DEX                         :Point X at next free location
1120STX#82                      :and store
1140LDY@2                       :
1160:LL7JSR#FE94                :Get 4 ASCII numerals from
1170JSR#FE52                    :keyboard and convert to
1180JSR#F87E                    :hexadecimal as per HEX DIRECT
1200ASLA                        :programme
1220ASLA                        :
1240ASLA                        :
1260ASLA                        :
1280STA#82,Y                    :
1300JSR#FB8A                    :
1320JSR#FE94                    :
1330JSR#FE52                    :
1340JSR#F87E                    :
1360CLC                         :
1380ADC#82,Y                    :
1400STA#82,Y                    :
1420JSR#FB8A                    :
1440DEY                         :
1460BNELL7                      :
1470INC#85                      :Add 1 to number of titles
1480LDX#82                      :
1500LDA#84                      :Store address of programme
1520STA#9600,X                  :immediately after title
1540DEX                         :
1560LDA#83                      :
1580STA#9600,X                  :
1600DEX                         :
1620LDA@#7F                     :Store an end of address marker
1640STA#9600,X                  :
1650DEX                         :
1660LDA@#FF                     :Store end of data marker
1670STA#9600,X                  :
1680LDA#85                      :Check to see if title store
1690CMP@#C                      :full
1700BCSLL3                      :
1705LDA@TT4&#FF                 :Print message string 5
1710STA#80                      :
1715JSRKK0                      :
1720JSR#FE94                    :Wait for key
1725CMP@CH"Y"                   :Is it Y
1730BNELL8                      :If not GOTO LL8
1735JMPLL22                     :
```

```
1740:LL8JSR#FD69          :Clear VDU
1760LDY@0                 :Clear page 0 locations used by
1780STY#25                :#C589
1790STY#34                :
1800STY#43                :
1820LDA@TT5&#FF           :Print message string 6
1840STA#80                :
1860JSRKK0                :
1870JSR#FFED              :
1880LDY@#6F               :Point Y at title store
1900LDA@1                 :
1920STA#85                :
1930:LL9LDA#85            :Output title number
1960STA#16                :
1970STY#82                :
1980JSR#C589              :
1990JSR#F7FD              :
2000INC#85                :
2030LDY#82                :Y is saved as #C589 uses Y
2040:LL10LDA#9600,Y       :Output title
2060CMP@#D                :
2080BEQLL11               :
2100JSR#FE52              :
2120DEY                   :
2140JMPLL10               :
2160:LL11JSR#FFED         :
2180DEY                   :Move Y past #D, address and
2200DEY                   :end of address marker
2220DEY                   :
2230DEY                   :
2240LDA#9600,Y            :Check for end of data
2260CMP@#FF               :
2280BEQLL12               :
2300LDA#85                :
2320JMPLL9                :
2420:LL12LDA@TT2&#FF      :Print message string 3
2440STA#80                :
2460JSRKK0                :
2470LDY@0                 :
2480JSR#CD14              :Get title from keyboard
2500JMPLL13               :
5000:KK0JSR#FFED          :Print message string
5010:KK1LDY@0             :subroutine
5020:KK2LDA(#80),Y        :
5030CMP@#D                :
5040BEQKK3                :
5050JSR#FE52              :
5060INY                   :
5070JMPKK2                :
5080:KK3RTS               :
6020:LL13LDY@#6F          :See if entered title is in
6040:LL14LDX@0            :store
```



```
6060:LL15LDA#9600,Y       :
6070CMP@#D                :Note that if you have two
6080BEQLL18               :titles with similar characters
6090CMP#100,X             :this routine will always find
6100BNELL16               :the shorter one e.g MOON and
6120DEY                   :MOONS, MOON will be returned
6140INX                   :
6160JMPLL15               :
6170CMP@#D                :
6180:LL16                 :
6220:LL17DEY              :Move Y past end of address
6240LDA#9600,Y            :marker
6260CMP@#7F               :
6280BNELL17               :
6300DEY                   :
6320LDA#9600,Y            :Check for end of data marker
6340CMP@#FF               :
6360BNELL14               :If not found loop
6380LDA@TT6&#FF           :Print message string 7
6400STA#80                :
6420JSRKK1                :
6430LDX@#40               :Delay
6440JSR#FB83              :
6450JMPLL8                :
6460:LL18STY#82           :
6470JSR#FD69              :Clear VDU
6480LDY#82                :
6490DEY                   :
6500LDA#9600,Y            :Store H.B of address
6510STA#81                :in #81
6520DEY                   :
6540LDA#9600,Y            :Store L.B of address
6560STA#80                :in #80
6580DEY                   :
6600LDY@0                 :
6620LDA(#80),Y            :Check for BASIC programme
6640CMP@#D                :
6660BEQLL19               :If it is GOTO LL19
6740JMP(#80)              :Jump to machine code
6760:LL19LDA#80           :Transfer address to #5,#6
6780STA#5                 :
6790LDA#81                :
6800STA#6                 :
6810LDY@1                 :Store 0 at 2nd text area
6820LDA@0                 :location
6830STA(#5),Y             :Equivalent to BASIC's OLD
6840JMP#C2F2              :Jump to BASIC
6860:KK4LDA#9600,X        :Routine for finding end of
6870CMP@#D                :data
6880BNEKK5                :Also sets #85 to number of
6890INC#85                :current titles
6900:KK5CMP@#FF           :Returns with Y pointing to end
```

```
6910BEQKK6
6920DEX                          :of data marker
6930JMPKK4                       :
6940:KK6LDA#85                   :
6950CMP@#C                       :Checks for number of titles
6960BCCKK7                       :If less than 12 return
6970PLA                          :
6980PLA                          :Discard return address and
6990JMPLL3                       :jump to LL3
7000:KK7RTS                      :
8000J;R.                         :
```

LINES:-

520-580 Set up registers and page 0 locations.
600-760 Print message and interpret answers.
780-840 Get new programme title.
850-910 Store title.
940-990 Print message.
1040-1700 Get address of programme from keyboard as 4 ASCII codes, convert to hexadecimal and store after title. Store end of address marker and end of data marker. Check number of titles in store.
1705-1735 Print message and interpret answer.
1740-2460 Print message, print list of titles.
2480-2500 Get title from keyboard.
5000-5070 Print message to VDU subroutine.
6020-6440 See if title is in store.
6460-6600 If found load address into #80,#81.
6620-6660 See if BASIC programme.
6740 Jump to machine code programme.
6760-6840 Set #5,#6 to address of BASIC programme, set 2nd text space location to 0 and jump to interpreter.
6860-7000 Find end of data, check for number of titles in store.

Note that #CD14 is a later entry to subroutine #CD0F. The only difference is that the cursor remains in the same position.

Note also the use of arrays for holding the various message string addresses. Only the low bytes are accessed as all strings are stored in page #96.

Fruit Machine

This is a simple Fruit Machine type game with a good spinning effect. There is no scoring system but as the machine code jumps back to Line 40 after each spin a suitable routine could easily be inserted. Note that no extra spaces should be entered before Line 40.

PAGE 0 LOCATIONS USED:-
#80,#81 Contains address of line currently being printed.
#82 - #84 Contain L.Bs of current line addresses.
#85 - #87 Contain H.Bs of current line addresses.
#88 - #8A Pointers to reel start symbols.
#8B Register for number of lines.
#8C Temporary for X.
#8E H.B reel start address.
#8F - #91 L.Bs reel start addresses.
#92 - #94 Number of spins left counters.
#95 - #97 Flags indicating spins left.
#98 - #9F Display data.
SUBROUTINES USED:-
#C2F2 #FE71

ENTER IN TEXT AREA: #29
TEXT: 1.4K
VDU: 6K
M/C: .21K
EXECUTION: RUN

```
5REM FRUIT MACHINE               :
10DIMBB30                        :
20F.N=0TO30;P=#3900;BBN=P;N.     :
30F.N=1TO2;P=#3900;GOS.a;N.      :
40!#8E=#150C0388                 :
50!#98=#FFBD66E7                 :
60!#9C=#DBA5C399                 :
70!#88=#020406                   :
80F.N=1TO3                       :
90A=(ABSRND%(35-15))+15          :
100N?#91=A                       :
110N.N                           :
120CLEAR4                        :
140LI.#3900                      :
150E.                            :
500a[                            :
520:BB0LDX@3                     :Set up page 0 data for each
580:BB1JSRBB24                   :reel
640DEX                           :
660BNEBB1                        :
680:BB2LDX@3                     :
```

```
700:BB3JSRBB28          :Print reels routine
720DEX                  :
740BNEBB3               :
760DEC#8B               :
780BNEBB2               :
800:BB4LDX@3            :Set spins left flags
820:BB5LDA@1            :
880STA#94,X             :
900JSRBB24              :Reset reel start addresses
920DEX                  :
930BNEBB5               :
950:BB7BIT#B002         :Wait for flyback
960BMIBB7               :
970LDA#95               :See if any spins left reel 1
980BEQBB8               :If not GOTO BB8
1000LDX@1               :Otherwise print 1 new line
1020JSRBB28             :
1030:BB8BIT#B002        :Wait for flyback
1040BMIBB8              :
1050LDA#96              :See if any spins left reel 2
1060BEQBB9              :If not GOTO BB9
1080LDX@2               :Print 1 new line reel 2
1100JSRBB28             :
1110:BB9BIT#B002        :Wait for flyback
1120BMIBB9              :
1140LDA#97              :See if any spins left reel 3
1150BEQBB10             :If not GOTO BB10
1160LDX@3               :Print 1 new line reel 3
1180JSRBB28             :
1190:BB10DEC#8B         :Loop until all lines done
1220BNEBB7              :
1240LDX@3               :
1260:BB11JSRBB24        :Reset start address
1280LDA@1               :
1300DEC#91,X            :Decrement number of spins left
1320BNEBB12             :If result = 0 reset number of
1340STA#91,X            :spins to 1 and clear spins
1360LDA@0               :left flag
1380:BB12STA#94,X       :
1400DEX                 :Loop for all three reels
1420BNEBB11             :
1440LDA#95              :Check spins left flags, if any
1460BNEBB7              :set loop to BB7
1480LDA#96              :
1500BNEBB7              :
1520LDA#97              :
1540BNEBB7              :
1560:BB13BIT#B001       :See if SHIFT pressed
1600BPLBB14             :If so GOTO BB14 and exit
1700JSR#FE71            :Wait for key to be pressed
1720CPY@#FF             :
1740BEQBB13             :
```



```
1760LDA@#29             :Load return BASIC address into
1780STA#6               :#5,#6
1800LDA@#5A             :
1820STA#5               :
1840LDY@1               :
1860STA#3               :
1880JMP#C2F2            :Jump to BASIC
2700:BB14RTS            :
2720:BB24LDA#8E         :Store H.B and L.B of start
2740STA#84,X            :address for each reel, X
2760LDA#8E,X            :points to reel
2780STA#81,X            :
2800LDA@#41             :Set line counter
2820STA#8B              :
2840LDA#87,X            :Increment symbol pointer for
2860CLC                 :each reel
2880ADC@1               :
2900CMP@9               :If result >8 reset to 1
2920BCCBB27             :
2960LDA@1               :
2980:BB27STA#87,X       :
3000RTS                 :
3020:BB28LDA#81,X       :Transfer current reel line
3040STA#80              :address to #80,#81
3060LDA#84,X            :
3080STA#81              :
3100STX#8C              :Save X
3120TXA                 :Load X with pointer to symbol
3140TAY                 :
3160LDX#87,Y            :
3180LDY@6               :
3200:BB29LDA#97,X       :Print 1 line of reel display
3220STA(#80),Y          :
3240DEY                 :
3260BNEBB29             :
3280LDA#80              :Adjust #80,#81 to contain
3300CLC                 :address of next line
3320ADC@#20             :
3340STA#80              :
3360BCCBB30             :
3380INC#81              :
3400:BB30LDX#8C         :Restore X
3420LDA#80              :Store new line address
3440STA#81,X            :X points to reel
3460LDA#81              :
3480STA#84,X            :
3500RTS                 :
5000]:R.                :
```

LINES:-

520-780 Set up page 0 locations and display reels.

800-920 Set spins left flags.

950-1220 Check for spins left, print 1 line if found and loop for all lines.
1240-1540 Decrement spins left counter, set/reset flags and check if all reset.
1560-1880 Set up BASIC return address and jump to interpreter.
2720-3000 Subroutine to set reel start addresses, set line counter and adjust symbol pointers.
3020-3200 Transfers current reel addresses to #80,#81 and gets reel symbol data.
3220-3380 Prints 1 line of specified reel.
3400-3500 Stores next line address before returning.

Note the use of X register indexing where X is the reel number. This simplifies the accessing of addresses and data but note that addresses are not stored L.B then H.B. The L.B is stored 3 locations away from its H.B.

The action taken to give the impression of spinning is to print the top line of reel 1 with new data, wait then print top line of reel 2, wait then print top line of reel 3, wait then print 2nd line of reel 1, wait ..... continue until all lines have been printed.

String Write Mode 4

This programme replaces the normal write character routine address and allows strings to be written to the screen when using graphics mode 4. It is necessary to have the Word Pack ROM installed as this programme uses the 8 byte character codes stored in the ROM. If you do not have this ROM you will have to design your own characters using DESIGNLETTERS or enter known codes using HEX DIRECT. Codes should be entered in standard ASCII order, #20-#3F in the 1st page, #40-#5F in next and #60-#7F in 3rd page. Lines 1000,1070 and 1130 should be changed to reflect the new storage areas. On entry #82,#83 should contain address of VDU line.

PAGE 0 LOCATIONS USED:-
#80,#81 Contains address of codes.
#82,#83 Contains address of VDU line.
#90 - #92 Temporary store for Y,X and A.

ENTER IN TEXT AREA: Any
TEXT: .8K
VDU: 6K
M/C: .14K
EXECUTION: On output in Mode 4 if #208,#209 contain address of machine code.

```
5REM STRINGS IN MODE 4          :
10DIM KK11                      :
20F.N=0TO11;P=#3700;KKN=P;N.    :
30F.N=1TO2;P=#3700;GOS.a;N.     :
40E.                            :
500a[                           :
520:KKOSTY#90                   :Save registers
530STX#91                       :
540STA#92                       :
560LDY#3                        :Check to see if it is end of
570LDA(#5),Y                    :text which has been
580BPLKK1                       :interpreted, if so restore
590LDA@#52                      :standard write character
600STA#208                      :routine to #208,#209 , load
610LDA@#FE                      :accumulator with ">" and jump
620STA#209                      :to standard write character
630LDA@CH">"                    :address
640JMP(#208)                    :
900:KK1LDA#92                   :Is character lower case
910CMP@#60                      :
920BCSKK2                       :If so GOTO KK2
940CMP@#40                      :Is it upper case
950BCSKK3                       :If so GOTO KK3
960SEC                          :Calculate offset
```

```
970SBC@#20                  :
980JSRKK9                   :
990STA#80                   :
1000LDA@#AD                 :Load H.B address of codes
1010STA#81                  :
1020JMPKK4                  :
1030:KK2SBC@#60             :calculate offset
1040JSRKK9                  :
1050STA#80                  :
1070LDA@#AF                 :Load H.B
1080STA#81                  :
1090JMPKK4                  :
1100:KK3SBC@#40             :Calculate offset
1110JSRKK9                  :
1120STA#80                  :
1130LDA@#AE                 :Load H.B
1140STA#81                  :
1150:KK4LDX@0               :Print character routine
1160LDY@0                   :
1170:KK5LDA(#80),Y          :#80,#81 contain address of
1180STA(#82,X)              :codes
1190LDA#82                  :
1200CLC                     :
1210ADC@#20                 :Move VDU address down 1 line
1220STA#82                  :
1230INY                     :
1240CPY@8                   :Loop until all 8 codes have
1250BNEKK5                  :been printed
1260LDA#82                  :Adjust VDU address to next
1270CLC                     :free location
1280ADC@1                   :
1290CMP@#20                 :Check for end of line
1300BNEKK6                  :
1310JSRKK7                  :
1320:KK6STA#82              :Restore registers
1330LDY#90                  :
1340LDX#91                  :
1350RTS                     :
1360:KK7LDA#83              :Adjust #82,#83 to next line of
1370CLC                     :VDU
1380ADC@1                   :
1390CMP@#98                 :Check if bottom line, if so
1400BNEKK8                  :reset to start of bottom line,
1410LDA@#97                 :VDU does not scroll
1420:KK8STA#83              :
1430LDA@0                   :
1440RTS                     :
1640:KK9TAX                 :Subroutine to find L.B of
1650LDA@0                   :codes for specified character
1660:KK10CLC                :
1670ADC@8                   :
1680DEX                     :
```



```
1690BNEKK10                 :
1700RTS                     :
3000J;R.                    :
```

LINES:-

520-640 Save registers, check for end of text output and if found restore standard write character address and jump to that address.

900-1140 Check character to see which page its codes lie in, calculates position of codes and stores L.B in #80, H.B in #81.

1150-1250 Displays code, moves display address down by 1 line, gets next code and loops until all 8 codes displayed.

1260-1310 Moves display address 1 column forward and checks for end of line.

1320-1350 Restores registers and returns.

1360-1440 Moves display address down by 8 lines, checks for bottom of display and if found resets address to start of bottom line.

1640-1700 Calculates L.B address of codes.

Note that CTRL characters are not recognized and should not be input to this routine.

See PONTOON for another method of accessing character codes held in #A000.

Pontoon

The computer plays you at pontoon. Cards are displayed in Mode 4 and all standard pontoon rules are observed. It uses character codes stored in the Word Pack ROM so cannot be used unless this ROM is fitted or you have your own codes. The programme is in 4 parts as it is too long for the standard ATOM.

PAGE 0 LOCATIONS USED:-

#80,#81	Address of cards.
#82,#83	Address of data.
#84	Flag - 0 if computer playing.
#85	Pointer to computer's card.
#86	Pointer to player's card.
#87	Computer's score.
#88	Player's score.
#89	Pointer to Ace.
#90 - #99	L.Bs card addresses.
#9A,#9B	H.Bs card addresses.
#9C,#9D	H.Bs card addresses bottom line.

These are the main page 0 locations used.

SUBROUTINES USED:-

#C2F2 #C589 #FB83 #FD1A #FE94

PART NO: 1

```
ENTER IN TEXT AREA: #29
TEXT: .5K
VDU: -
M/C: -
EXECUTION: RUN

5REM PONTOON INSTRUCTIONS      :
10S=#2E20                      :
20?S=13                        :
30S=S+1                        :
40$S="YOUR SCORE IS:"          :
50S=S+LENS+1                   :
60$S="DO YOU WANT ANOTHER      :
CARD?"                         :
70S=S+LENS+1                   :
80$S="MY TURN"                 :
90S=S+LENS+1                   :
100$S="MY SCORE IS:"           :
110S=S+LENS+1                  :
120$S="I WIN"                  :
130S=S+LENS+1                  :
140$S="YOU WIN"                :
150S=S+LENS+1                  :
160$S="PONTOON"                :
```



```
170S=S+LENS+1                  :
180$S="BUST!!"                 :
190S=S+LENS+1                  :
200$S="FIVE CARD TRICK"        :
210S=S+LENS+1                  :
220$S="ANOTHER GO?"            :
230S=S+LENS+1                  :
240$S="CARDS BEING SHUFFLED"   :
250S=S+LENS+1                  :
260$S="CURRENT SCORE POSITION  :
IS:"                           :
270S=S+LENS+1                  :
280$S="YOUR WINS:"             :
290S=S+LENS+1                  :
300$S="MY WINS:"               :
310E.                          :
```

PART NO: 2

```
ENTER IN TEXT AREA: #2F
TEXT: .1K
VDU: -
M/C: -
EXECUTION: -

10X=0                          :
20Y=#3000                      :
30K=#3040                      :
40DO                           :
50b W=ABSRND%52                :
60IFW?Y=0 G.b                  :
70X?K=W?Y                      :
80W?Y=0                        :
90X=X+1                        :
100U.X=52                      :
110LI.(Z+1)                    :
```

PART NO: 3

```
ENTER IN TEXT AREA: #82
TEXT: 5.2K
VDU: -
M/C: 1.1K
EXECUTION: -

5REM PONTOON                   :
10DIM NN63,MM19                :
20F.N=0T063;P=#29B0;MMN=P;     :
```

```
NNN=P;N.                        :
30F.N=1TO2;P=#29B0;GOS.a;N.     :
40E.                            :
500a[                           :
520:NNOLDA#5                    :Save BASIC address
540STA#30F0                     :
560LDA#6                        :
580STA#30F1                     :
600JMPMMO                       :
620:NN1LDA#80                   :Increment VDU address by 1
640CLC                          :line routine
660ADC@#20                      :
680STA#80                       :
700RTS                          :
720:NN2LDX@7                    :Print card subroutine
740:NN3LDY@4                    :4 columns x 56 line matrix
760:NN4LDA#8D                   :
780STA(#80),Y                   :
800DEY                          :
820BPLNN4                       :
840JSRNN1                       :Note loop using Carry flag and
860BCCNN3                       :JSR NN1
880INC#81                       :
900DEX                          :
920BPLNN3                       :
940RTS                          :
960:NN5PLA                      :Store return address in
980STA#33B                      :integer variable Z
1000PLA                         :
1020STA#356                     :
1040JSRNN39                     :Clear text lines subroutine
1090LDX@11                      :Print string 11
1100JSRNN53                     :
1110LDY@#FF                     :Set up pack
1120LDA@0                       :Set card pointer to 0
1140STA#8A                      :
1150LDA@#80                     :Set bit 7 of Accum
1160STA#21C                     :Store in #21C, top 4 bits
1170:NN6LDX@13                  :indicate suit, set up number
1180:NN7INY                     :of cards per suit
1190TXA                         :Add card number to suit
1200CLC                         :
1210ADC#21C                     :
1220STA#3000,Y                  :Store at #3000,Y, Y points to
1240DEX                         :free location, do for 13 cards
1250BNENN7                      :
1260LSR#21C                     :Shift set bit to indicate next
1280LDA#21C                     :suit
1300EOR@#08                     :Check if all suits done
1320BNENN6                      :
1420LDA@#2F                     :Load address of BASIC card
1440STA#6                       :shuffle
```



```
1460LDA@03                      :
1480STA#5                       :
1500JMP#C2F2                    :Jump to BASIC
1520:NN8LDX@4                   :Find suit routine
1540AND@#F0                     :Clear low 4 bits
1560CLC                         :Shift Accum left until set bit
1580:NN9DEX                     :falls into carry, X points to
1600ASLA                        :suit, get ASCII code for suit
1620BCCNN9                      :
1640LDA#21D,X                   :
1660:NN10STA#225                :Save in temporary
1680LDA@#AD                     :Store H.B of codes address in
1700STA#83                      :#83
1720ASL#225                     :Move Accum bits left, check
1740LDA#225                     :for bit 7 set, if set code is
1760BPLNN11                     :upper case character so
1780INC#83                      :increment H.B
1800:NN11ASL#225                :Shift Accum left another 2
1880:NN12LDA#225                :times, this gives L.B address
1900ASLA                        :of character codes, #82,#83
1920STA#82                      :now contain address of codes
1940LDY@0                       :Print character routine
1960LDX@0                       :
1980:NN13LDA(#82),Y             :
2000EOR@#FF                     :Invert character
2020STA(#80,X)                  :
2040INY                         :
2060JSRNN1                      :
2080BCCNN13                     :
2100RTS                         :
2120:NN14LDY@5                  :Clear page 0 locations used by
2140LDA@0                       :score calculation routine
2160:NN15STA#226,Y              :
2180DEY                         :
2200BNENN15                     :
2220RTS                         :
2240:NN16CLC                    :Adjust Accum to contain ASCII
2260ADC@#30                     :code for numeral
2280JMPNN10                     :
2300:NN17JSRNN26                :Score calculation routine
2320AND@#0F                     :Get current card value and
2340CMP@#A                      :clear suit bits
2360BCCNN18                     :Compare to 10, if >10 card =10
2380LDA@#A                      :
2400JMPNN19                     :
2420:NN18CMP@1                  :Check for ACE, if ACE change
2440BNENN19                     :value to 11 and store in
2460LDA@11                      :temporary score table, X
2480:NN19LDX#84                 :points to which player, Y
2500LDY#85,X                    :points to card
2520STA#226,Y                   :
2540:NN20LDY@5                  :Calculate score by adding
```

```
2560LDA@0
2580STA#87,X
2600STA#89
2620:NN21LDA#225,Y
2640CMP@11
2660BNENN22
2680STY#89
2700:NN22CLC
2720ADC#87,X
2740STA#87,X
2760DEY
2780BNENN21
2800CMP@22
2820BCSNN24
2840:NN23JMPNN27
2860:NN24LDA#89
2880BEQNN23
2900TAY
2920LDA@1
2940STA#225,Y
2960JMPNN20
2980:NN25LDX#84
3000LDA#9A,X
3020STA#81
3040LDY#85,X
3060LDA#90,Y
3080STA#80
3100JMPNN2
3120:NN26LDY#8A
3140LDA#3040,Y
3160RTS
3180:NN27LDY#8A
3200INY
3220CPY@52
3240BCCNN28
3300JMPNN5
3320:NN28STY#8A
3340RTS
3360:NN30LDY@0
3380LDA(#8C),Y
3400CMP@#D
3420BEQNN32
3440JSRNN10
3460LDA#8C
3480CLC
3500ADC@1
3520STA#8C
3540BCCNN31
3560INC#8D
3580:NN31INC#80
3600JMPNN30
3620:NN32RTS
```

```
:contents of temporary table
:Clear last score
:Clear ACE pointer
:
:Check for ACE
:
:If found store pointer
:
:
:
:Loop until all 5 locations
:have been added
:Compare to 22
:If >= GOTO NN24
:
:See if ACE found
:
:If found store 1 in place of
:11 and do calculation again
:
:
:Get pointer to computer's or
:player's card, get H.B of
:address and store in #81, get
:pointer to card, get L.B
:address and store in #80
:
:
:Get value of next card from
:pack, return with value in
:Accum
:Update pointer to pack
:
:If all 52 cards used GOTO
:shuffle routine
:
:
:
:Print string routine
:
:Do not output #D
:
:
:
:
:
:
:
:
:
:
:
:
```



```
3640:NN33LDA@0
3660STA#25
3680STA#34
3720STA#43
3740LDX#84
3760LDA#87,X
3780STA#16
3800:NN34INC#80
3820JSRNN37
3840JSR#C589
3860RTS
3880:NN35STX#219
3900STY#21A
3920CMP@#20
3940BEQNN36
3960JSRNN10
3980INC#80
4000LDY#21A
4020LDX#219
4040BNENN36
4060PLA
4080PLA
4100LDA@#FE
4120STA#209
4140LDA@#52
4160STA#208
4180:NN36RTS
4200:NN37TSX
4230INX
4240INX
4260LDA#100,X
4280STA#209
4300DEX
4320LDA#100,X
4340CLC
4350ADC@5
4360STA#208
4370BCCNN29
4380INC#209
4390:NN29RTS
4400:NN38STX#219
4440LDX@#40
4460JSR#FB83
4500LDX#219
4520RTS
4540:NN39LDX@3
4560LDA@#8A
4580STA#81
4600LDA@#C0
4620STA#80
4640:NN40LDY@#1F
4660:NN41LDA@#FF
```

```
:Print score routine
:Subroutine #C589 is used to
:convert score from hexadecimal
:to decimal. However this
:routine outputs via address
:held in #208,#209. This
:address is #FE52 which will
:not print to VDU in mode4, it
:is necessary to store a new
:routine address, routine NN37
:does this by calculating
:address of NN35 (NN35 could be
:obtained direct)
:NN35 is subroutine to print
:score after conversion. #C589
:will not return to correct
:address when X=0 so when X
:reaches 0 the 1st return
:address on stack is discarded,
:it will now return correctly
:
:
:
:
:
:
:
:Works out address of NN35 by
:taking its own return address
:and adding 5, it then stores
:this address at #208,#209
:
:
:
:
:
:
:
:
:
:Wait subroutine, delay of
:approx 1 sec
:
:
:
:Routine to clear text lines,
:As each character needs 8
:lines, two text lines would be
:16 VDU lines however as
:characters are inverted it is
:necessary to add 1 extra VDU
:line to top to avoid visual
```

```
4680STA(#80),Y          :problems
4700DEY                 :
4720BPLNN41             :
4740JSRNN1              :
4760BCCNN40             :
4780INC#81              :
4800DEX                 :
4820BNENN40             :
4840RTS                 :
4860:NN42LDX#84         :Load #80,#81 with address of
4880LDA#9A,X            :card just displayed
4900STA#81              :
4920LDY#85,X            :
4940LDA#90,Y            :
4960STA#80              :
4970STA#8E              :
4980JSRNN44             :Print card number top lt
5000LDX#84              :Get L.B of card address top rt
5020LDY#85,X            :
5040LDA#95,Y            :
5060STA#80              :
5080JSRNN43             :Print suit top rt
5100LDX#84              :Get H.B bot lf
5120LDA#9C,X            :
5140STA#81              :
5160LDY#85,X            :Get L.B bot lf
5180LDA#90,Y            :Print suit bot lf
5200STA#80              :
5220JSRNN43             :Print suit bot lf
5240LDX#84              :Get L.B bot rt
5260LDY#85,X            :
5280LDA#95,Y            :
5300STA#80              :
5320JSRNN44             :Print number bot rt
5340RTS                 :
5360:NN43JSRNN26        :Get current card and print
5380JMPNN8              :
5400:NN44JSRNN26        :Get card value
5420AND@#0F             :Clear suit
5440CMP@#A              :See if 10 or less
5460BCCNN46             :If less GOTO NN46
5480BEQNN48             :If equal GOTO NN48
5500SEC                 :Otherwise calculate ASCII code
5520SBC@#A              :for J,Q,K
5540TAX                 :#221,X contain codes, return
5640LDA#221,X           :with code in Accum
5660JMPNN10             :
5680:NN46CMP@1          :See if ACE
5700BEQNN47             :If so GOTO NN47
5720JMPNN16             :Must be 2 to 9 so GOTO NN47
5740:NN47LDA#221        :Get code for ACE
5760JMPNN10             :
```



```
5780:NN48LDA#8E         :If 10, it is necessary to
5800BPLNN49             :print 2 numerals on top lf and
5820DEC#80              :bot rt, bit 7 of #8E is set
5880:NN49LDA@CH"1"      :for bot rt, clear for top lf
5900JSRNN10             :
5920INC#80              :
5940LDA@CH"0"           :
5960JSRNN10             :
5980LDA@#FF             :Set bit 7 of #8E
6000STA#8E              :
6020RTS                 :
6040:NN50LDA@#2E        :Load address of string table
6060STA#8D              :into #8C,#8D
6080LDA@#20             :
6100STA#8C              :
6120LDY@#FF             :
6140:NN51INY            :Loop until #D is found
6160LDA(#8C),Y          :
6180CMP@#D              :
6200BNENN51             :
6220INY                 :Adjust #8C so that #8C,#8D
6240TYA                 :contain address of next string
6260CLC                 :
6280ADC#8C              :
6300STA#8C              :
6320LDY@0               :Loop until
6340DEX                 :X=0
6360BNENN51             :
6380RTS                 :
6400:NN52JSRNN50        :Print string on text line 1
6420LDA@#8B             :routine
6440STA#81              :
6460LDA@1               :
6480STA#80              :
6500JMPNN30             :
6520:NN53JSRNN50        :Print string on text line 2
6540LDA@#8C             :routine
6560STA#81              :
6580LDA@1               :
6600STA#80              :
6620JMPNN30             :
6640:NN54LDX#84         :Check score for PONTOON,BUST
6660LDA#87,X            :or FIVE CARD TRICK
6680CMP@22              :If over 21
6700BCCNN56             :
6720INC#80              :Print BUST message
6730INC#80              :
6740LDX@8               :
6750JSRNN50             :
6760JSRNN30             :
6780LDA#84              :Check if player or computer
6800BNENN55             :BUST
```

```
6810JMPNN61
6820:NN55JMPNN63
6850:NN56CMP@21
6860BNENN58
6880LDA#85,X
6900CMP@1
6920BNENN59
6930INC#90
6940INC#80
6950LDX@7
6960JSRNN50
6970JSRNN30
6980:NN45LDA#84
6990BNENN57
7000JMPNN63
7010:NN57PLA
7020PLA
7030JMPMM4
7070:NN58LDA#85,X
7080:NN59CMP@4
7090BEQNN60
7100RTS
7110:NN60LDX@1
7120LDA@#8C
7130STA#81
7140LDA@0
7150STA#80
7160JSRNN40
7180LDX@9
7190JSRNN53
7200JMPNN45
7260:NN61JSRNN38
7270JSRNN39
7290JSRMM17
7310LDX@6
7320:NN62JSRNN52
7340PLA
7360PLA
7380JMPMM11
7400:NN63JSRNN38
7410JSR#FD1A
7420JSRNN39
7430JSRMM12
7440LDX@5
7460JMPNN62
8020:MM0JSRNN39
8030JSRNN5
8040:MM1JSRNN14
8050LDA@0
8060STA#84
8080STA#85
8100STA#86
```

```
:
:
:See if 21
:
:See if 21 in two cards
:
:
:If it is then print PONTOON
:message
:
:
:
:See if computer or player
:
:
:
:
:See if FIVE CARD TRICK
:If it is GOTO NN60
:
:
:Clear 2nd text line
:
:
:
:
:
:Print FIVE CARD TRICK message
:
:
:YOU WIN routine
:Clear text lines
:Update current wins store
:Print string 6
:
:Discard return address
:
:
:I WIN routine
:Sound bell
:
:
:Print string 5
:
:Clear text lines
:Set up pack
:Clear score stores
:
:
:
:
```



```
8110STA#87
8120STA#88
8140LDA@#AA
8160STA#8D
8180JSRNN25
8200LDA@#FF
8220STA#8D
8240INC#84
8260JSRNN25
8280JSRNN42
8300JSRNN17
8320INC#85
8340DEC#84
8360LDA@#AA
8380STA#8D
8400JSRNN25
8420INC#84
8440:MM3INC#86
8460LDA@#FF
8480STA#8D
8500JSRNN25
8520JSRNN42
8540JSRNN17
8560JSRNN39
8580LDX@1
8600JSRNN52
8620JSRNN33
8630JSRNN54
8640LDX@2
8660JSRNN53
8680JSR#FE94
8700CMP@CH"Y"
8720BNEMM4
8740JMPMM3
8750:MM4JSRNN14
8780LDA@0
8800STA#84
8820STA#85
8840JSRNN38
8860JSRNN39
8880LDX@3
8900JSRNN52
8920:MM5JSRNN38
8940LDA@#FF
8960STA#8D
8980JSRNN25
9000JSRNN42
9020JSRNN17
9040JSRNN39
9060LDX@4
9080JSRNN52
9100JSRNN33
```

```
:
:
:Store card display data, #AA
:for face down #FF for face up
:Print 1st card
:
:
:Indicate players card
:Print card
:Print suit and number
:GOSUB score routine
:Increment computers card
:Indicate computers card
:Print computers 2nd card
:
:
:Indicate players card
:
:
:
:Print card
:Print suit and number
:GOSUB score routine
:Clear text lines
:
:Print string 1
:Output score
:GOSUB score check routine
:
:Print string 2
:Wait for key to be pressed
:Is it Y
:
:If so loop for next card
:Clear score table
:
:Set #84 to point to computers
:1st card, indicate computers
:card
:Clear text lines
:Print string 3
:
:Wait
:Print cards, suit and numbers,
:GOSUB score routine and print
:string until computer's score
:is greater than 16
:
:
:
:
:
```

```
9120JSRNN54
9140INC#85
9160LDA#87
9170CMP@16
9180BCSMM6
9190JMPMM5
9200:MM6CMP#88
9210BCSMM7
9220LDA#89
9230BNEMM19
9240JSRNN26
9250AND@#0F
9260CLC
9280ADC#87
9300CMP@22
9320BCSMM7
9340:MM19JMPMM5
9360:MM7LDA#86
9380CMP@1
9400BNEMM8
9420LDA#88
9440CMP@21
9460BNEMM9
9480JSRNN61
9500:MM8CMP@4
9520BNEMM9
9530LDA#89
9540BNEMM19
9550JSRNN61
9560:MM9LDA#87
9580CMP#88
9600BCSMM10
9620JSRNN61
9640:MM10JSRNN63
9660:MM11JSRNN38
9680JSRNN39
9690JSRMM16
9700LDX@10
9720JSRNN52
9740JSR#FE94
9760CMP@CH"Y"
9780BNEMM15
9800LDA@#82
9820STA#81
9840LDA@0
9860STA#80
9880:MM13LDY@#1F
9900:MM14LDA@0
9920STA(#80),Y
9940DEY
9960BPLMM14
9980JSRNN1
```

```
:
:
:
:
:
:If you don't understand what
:lines 9200 - 9340 do, play
:game first then look at back
:of book!
:
:
:
:
:
:
:
:Checks for player's pontoon,
:if found jump to YOU WIN
:routine
:
:
:
:
:Checks for player's five card
:trick, if found jumps to YOU
:WIN routine
:
:
:Checks player's score against
:computer's
:If less jump to I WIN routine
:otherwise jump to YOU WIN
:routine
:
:
:Print string 10
:
:
:Wait for key to be pressed
:
:
:Clear VDU and jump to MMi
:
:
:
:
:
:
:
:
:
```



```
10060BCCMM13
10080INC#81
10100LDA#81
10120CMP@#98
10140BCCMM13
10160JMPMM1
10500:MM15LDA#30F0
10520STA#5
10540LDA#30F1
10560STA#6
10580RTS
10600:MM16JSRNN39
10620LDX@12
10640JSRNN52
10680LDX@13
10700JSRNN53
10720LDA#230
10740STA#16
10760LDA#231
10780STA#24
10800JSRNN34
10820INC#80
10840INC#80
10860LDX@14
10880JSRNN50
10890JSRNN30
10900LDA#232
10910STA#16
10920LDA#233
10930STA#24
10940JSRNN34
10950JSRNN38
10960JSRNN38
10970JSRNN38
10980JSRNN39
10990RTS
11000:MM17LDA#230
11010CLC
11020ADC@1
11040STA#230
11060BCCMM18
11080INC#231
11100:MM18RTS
11120:MM12LDA#232
11140CLC
11160ADC@1
11180STA#232
11200BCCMM2
11220INC#233
11240:MM2RTS
11300]
11340R.
```

```
:
:
:
:
:
:
:Restore original BASIC address
:and exit
:
:
:
:Print current score  and wait
:approx 3 secs
:
:
:
:
:
:
:
:
:
```

PART NO: 4

ENTER IN TEXT AREA: #29
TEXT: .2K
VDU: 6K
M/C: -
EXECUTION: RUN

```
5REM PONTOON                    :
10!#90=#130D0701                :
20!#94=#110B0519                :
30!#98=#8E821D17                :
40!#9C=#8C8B9589                :
50!#21C=#53484443               :
60!#221=#4B514A41               :
70!#230=0                       :
80CLEAR 4                       :
90LI.#29B0                      :
100E.                           :
```

Notes for PART 3.
LINES:-

520-600 Save BASIC address.
620-700 VDU line increment subroutine.
720-940 Display card subroutine.
960-1500 Set up pack and jump to BASIC shuffle subroutine.

1540-2100 Calculate address of character codes and print to VDU.
2120-2280 Clear score stores subroutine.
2300-2960 Get current card value, store in score store, calculate current score. Allow for cards >10 and for aces. Store ace as 11 on first pass, if total score >21 with ace, set ace =1 and recalculate.
2980-3100 Get address of next card, X is pointer to computer's card or player's card, Y points to which card.
3120-3160 Get value of card from pack.
3180-3340 Increment pointer to pack, if all cards used jump to set up pack routine.
3360-3620 Print string subroutine.
3640-4390 Print score subroutine.
4400-4520 Delay subroutine.
4540-4840 Clear text lines subroutine.
4860-6020 Print suit and number on card, take account of ace, jack, queen and king. Print 2 numerals if 10.
6040-6380 Find address of string pointed to by X.
6400-6500 Print string on top text line.
6520-6620 Print string on bottom text line.
6640-7460 Check score routine, take account of BUST,

PONTOON and FIVE CARD TRICKS.
8020-8420 Set up pack and data locations, print computer's 1st card face down, adjust pointers, print player's 1st card face up, go score routine, adjust pointers, print computer's 2nd card face down, adjust pointers.
8440-8740 Print player's 2nd card, go score routine, print score, prompt for another card, adjust pointers, loop until no more cards required or five cards dealt or BUST.
8750-9340 Computer's routine to print cards, calculate score and adjust pointers.
9360-9620 Final score check routine. Looks for player's pontoon or five card trick, if not found compares player's score with computer's score and takes appropriate action. This routine will only be entered if computer does not bust or achieve a pontoon or a five card trick.
9640-9780 Prompts for another go and interprets answer.
9800-10160 Clears screen.
10500-10580 Restore BASIC address and exit.
10600-10990 Display current wins position.
11000-11100 Increment player's win.
11120-11300 Increment computer's wins.

Text of part 3 just fits into graphics space (#8200-#97FF). If you enter too many extra spaces you may find that ERROR 30 LINE 10 occurs. This will be due to insufficient room for the labels to be dimensioned. A considerable amount of memory space is used by labels as each label is reserved 4 bytes even though only 2 will be used. In this programme 84 labels are used so 336 bytes of memory space have to be available between TOP and #97FF.

Subroutine NN37 is a way of calculating the address of NN35, it is only incorporated in this programme to show the use of return addresses. NN35's address can be accessed using LDA@NN35/256 for the High byte and LDA@NN35&#FF for the low byte.

It is a good idea to save the text of part 3 on one tape and after assembly to *SAVE "PONTOON" 2900 2FFF to another tape.

If you do not have the Word Pack ROM fitted create your own codes using pages #31 and #32, numerals and upper case. Line 1680 would need changing to LDA@#31.

Pools Prediction

This programme will store a pools data base, add to that data base, update the data and predict draws. The prediction algorithm creates a probability factor of +127 to -128 by the addition and subtraction of previous results. This algorithm could easily be replaced by one of your own choosing. The data base is stored in Mode 4 graphics RAM.

PAGE 0 LOCATIONS USED:-

#80,#81	Address of storage space.
#82,#83	Address od data.
#84,#85	Return addresses.
#8F	Probability factor.
#90,#91	Address of VDU line.
#98	Error flag.
#99	Temporary for X.
#9A	Temporary store for Home wins.
#9B	Temporary store for Draws.
#9C	Probability factor for Home team.
#9D	Temporary store for Away wins.
#9E	Temporary store for Draws.
#9F	Pointer to year.

SUBROUTINES USED:-

#C589	#F7D1	#F7FD	#FB83	#FBEE	#FC38
#FC7C	#FE52	#FE94	#FFED		

PART NO: 1

ENTER IN TEXT AREA: #82
TEXT: 5.2K
VDU: -
M/C: 1.1K
EXECUTION: -

```
5REM POOLS M/C                          :
10DIM LL40,SS9,WW17,KK2,EE2,            :
BB4                                     :
30F.N=0TO40;P=#3000;LLN=P;              :
SSN=P;WWN=P;KKN=P;BBN=P;                :
EEN=P;N.                                :
50F.N=1TO2;P=#3000;GOS.a;N.             :
70$P="HOME TEAM"                        :
90Q=P+LENP                              :
110F.N=1TO2;P=Q;GOS.b;N.                :
130$P="AWAY TEAM"                       :
150Q=P+LENP                             :
170F.N=1TO2;P=Q;GOS.c;N.                :
190$P="TEAM NAME NOT FOUND"             :
210Q=P+LENP                             :
230F.N=1TO2;P=Q;GOS.d;N.                :
```



```
250E.                                   :
270a[                                   :
290:LL38LDA@#81                         :Set up VDU line and column
310STY#E0                               :
330STA#DF                               :
350LDA@#40                              :
370STA#DE                               :
390JSR#F7D1                             :Print string following until
410];R.                                 :character found with bit 7 set
430b[                                   :
470:LL40LDA@#80                         :Home cursor
490STA#DF                               :
510LDA@0                                :
530STA#DE                               :
550RTS                                  :
570:LL39LDY@0                           :Set up VDU line and column
590STY#E0                               :
610LDA@#81                              :
630STA#DF                               :
650LDA@#80                              :
670STA#DE                               :
690JSR#F7D1                             :Print string
710];R.                                 :
730c[                                   :
750NOP                                  :Note this instruction to exit
770JMPLL40                              :from #FD71
790:LL0LDA@#20                          :
810:LL1STA(#90),Y                       :
830:LL2JSR#FE94                         :Get character from keyboard
850CMP@#7F                              :and recognize DELETE
870BNELL5                               :
890DEY                                  :
910BPLLL0                               :
930:LL3LDY@0                            :
950:LL4LDA@#3F                          :Load Accum with ASCII "?"
970JMPLL1                               :
990:LL5RTS                              :
1010:LL6LDA@#83                         :Compare name whose address is
1030STA#83                              :held in #80,#81 with data bank
1050LDA@0                               :whose address is held in #82,
1070STA#82                              :#83, if not equal find start
1090:LL7LDY@#FF                         :of next string, store in #82,
1110:LL8INY                             :#83 and loop until valid
1130LDA(#82),Y                          :comparison or end of data
1150CMP@#FF                             :found, if end of data found
1170BEQLL10                             :discard return address and
1190CMP(#80),Y                          :output "TEAM NAME NOT FOUND"
1210BNELL9                              :
1230CMP@#D                              :
1250BNELL8                              :
1270RTS                                 :
1290:LL9INY                             :
```

```
1310LDA(#82),Y
1330CMP@#D                :
1350BNELL9                :
1370TYA                   :
1390CLC                   :
1410ADC@13                :
1430ADC#82                :
1450STA#82                :
1470BCCLL7                :
1490INC#83                :
1510JMPLL7                :
1530:LL10PLA              :
1550PLA                   :
1570JSR#FFED              :
1590JSR#F7D1              :
1610];R.                  :
1630d[                    :
1650NOP                   :
1670JSRLL11               :
1690LDA@#80               :Set #90,#91 to VDU start
1710STA#91                :address
1730LDA@#20               :
1750STA#90                :
1770JSRLL11               :
1790JMP(#84)              :Delay
1810:LL11LDX@#30          :Jump to address stored by LL36
1830JSR#FB83              :Delay
1850LDY@#17               :
1870LDA@#20               :Clear line whose address is in
1890:LL12STA(#90),Y       :#90,#91
1910DEY                   :
1930BPLLL12               :
1950RTS                   :
1970:LL13JSRBB1           :
1990LDA@0                 :Set up current match storage
2010STA#E1                :area
2030LDA@#FF               :
2050STA#8F                :
2070LDA#33B               :
2090STA#9F                :Store number of matches in #9F
2110:LL14LDY@0            :
2130STY#9A                :Clear draw prediction work
2150STY#9B                :space
2170STY#9D                :
2190STY#9E                :
2230JSRLL38               :
2250JSRLL36               :Print HOME TEAM
2270JSRBB2                :Save return address
2310LDA@#4B               :Store H.B VDU line address
2330STA#90                :Store L.B VDU line address in
2350JSRLL11               :#90
2370JSRLL3                :Clear line
                          :Get character
```



```
2390:LL15STA(#80),Y       :Store character
2410CMP@#D                :
2430BEQLL16               :
2450AND@#3F               :Convert to ATOM VDU code
2470STA(#90),Y            :Output to VDU
2490INY                   :
2510JSRLL2                :
2530JMPLL15               :Loop until #D is input
2550:LL16JSRLL6           :Find address of HOME TEAM in
2570JSRLL33               :data bank, on return Y points
2590DEY                   :to #D, adjust #80,#81 to
2610LDX@4                 :contain 1st free location
2630:LL17INY              :address in current match
2650LDA(#82),Y            :storage area
2670CLC                   :Get HOME team data routine
2690ADC#9A                :On exit #9A will hold number
2710STA#9A                :of HOME wins over past seasons
2730INY                   :and #9B will hold number of
2750INY                   :draws
2770LDA(#82),Y            :
2790CLC                   :
2810ADC#9B                :
2830STA#9B                :
2850DEX                   :Loop for all years stored, X =
2870BNELL17               :number of years
2890LDA#9A                :Calculate probability and
2910SEC                   :store in #9C
2930SBC#9B                :
2950STA#9C                :
2970LDA#90                :Move VDU address down 1 line
2990CLC                   :
3010ADC@#20               :
3030STA#90                :
3050LDY@0                 :
3070LDA@22                :Output  "V" to VDU
3090STA(#90),Y            :
3110JSRLL39               :Print AWAY TEAM
3130JSRLL36               :Save return address
3150JSRBB2                :Set #90,#91 to VDU line
3190LDA@#8B               :address
3210STA#90                :
3230JSRLL3                :
3250:LL18STA(#80),Y       :Get characters from keyboard,
3270CMP@#D                :store in current match storage
3290BEQLL19               :area and loop until #D input
3310AND@#3F               :
3330STA(#90),Y            :
3350INY                   :
3370JSRLL2                :
3390JMPLL18               :
3410:LL19JSRLL6           :Find address of AWAY team in
3430JSRLL33               :data bank
```

```
3450DEY
3470LDX@4
3490:LL20INY
3510INY
3530LDA(#82),Y
3550CLC
3570ADC#9D
3590STA#9D
3610INY
3630LDA(#82),Y
3650CLC
3670ADC#9E
3690STA#9E
3710DEX
3730BNELL20
3750LDA#9D
3770SEC
3790SBC#9E
3810CLC
3830ADC#9C
3850BPLLL21
3870CMP#8F
3890BCSLL21
3910STA#8F
3930:LL21LDY@0
3950STA(#80),Y
3970INY
3990LDA@#80
4010STA(#80),Y
4030JSRLL33
4050JSRLL11
4070DEC#9F
4090BEQLL22
4110JMPLL14
4130:LL22JMPWW4
4210:LL23LDX@0
4230STX#25
4250STX#34
4270STX#43
4290STX#321
4310LDA#8F
4330STA#21C
4350:LL24JSRBB1
4370:LL25LDY@#FF
4390INX
4410:LL26INY
4430LDA(#80),Y
4450EOR@#80
4470BNELL26
4490DEY
4510LDA(#80),Y
4530CMP#8F
```

```
:
:
:Get AWAY data routine, on exit
:#9D holds AWAY wins and
:#9E holds draws
:
:
:
:
:
:
:
:
:
:Calculate AWAY probability
:factor
:
:Add to HOME probability factor
:
:If +ve number GOTO LL21
:See if highest -ve number, if
:so store in #8F
:
:Store probability factor in
:current match store
:Point Y at next location
:Set bit 7 of location to
:indicate end of factor
:Adjust address in #80,#81
:Delay
:Loop until all matches entered
:
:
:Store end of data marker
:Clear page 0 locations used by
:#C589
:
:
:Set field width to 0
:Store highest probability
:factor in #21C
:Reset #80,#81
:
:
:
:Find location with bit 7 set
:
:
:Point Y at probability factor
:and check to see if factor is
:equal to contents of #8F
```



```
4550BNELL31
4560STX#99
4570STX#16
4590JSR#C589
4600LDX#99
4610JSR#F7FD
4630LDY@0
4650:LL27LDA(#80),Y
4670CMP@#D
4690BEQLL28
4710JSR#FE52
4730INY
4750JMPLL27
4770:LL28JSR#F7FD
4790LDA@CH"V"
4810JSR#FE52
4830JSR#F7FD
4850:LL29INY
4870LDA(#80),Y
4890JSR#FE52
4910CMP@#D
4930BNELL29
4950INY
4990JSR#FFED
5010DEC#9F
5030BNELL31
5070:LL30RTS
5090:LL31INY
5110INY
5130LDA(#80),Y
5150CMP@#FF
5170BEQLL32
5190JSRLL34
5210JMPLL25
5230:LL32INC#8F
5240LDA#8F
5250CMP#21C
5270BEQLL30
5290LDX@0
5310JMPLL24
5350:LL33INY
5370:LL34TYA
5390CLC
5410ADC#80
5430STA#80
5450BCCLL35
5470INC#81
5490;LL35RTS
5510:LL36PLA
5530STA#84
5550PLA
5570STA#85
```

```
:
:Save X, X points to match
:number
:Output match number
:
:
:
:Output HOME team name
:
:
:
:
:Output space
:Output "V"
:
:Output space
:Output AWAY team name
:
:
:
:
:
:Loop until number
:of matches = 0
:
:Check for end of data
:
:
:
:Adjust #80,#81 and loop
:
:Increment #8F, when #8F = #21C
:256 increments have been
:completed so exit
:
:Otherwise set match number to
:0 and loop
:Subroutine to add Y to
:contents of #80,#81
:
:
:
:
:
:Save return address +1 in #84,
:#85
:
:
```

```
5590PHA
5610LDA#84
5630PHA
5650CLC
5670ADC@1
5690STA#84
5710BCCLL37
5730INC#85
5750:LL37RTS
5770:SS0LDA@#65
5790STA#90
5810LDX@4
5830STX#9F
5850JSRBB2
5890:SS1LDA@8
5910STA#9E
5674:SS2LDY@0
5950LDA@#A0
5970STA(#90),Y
5990BIT#B001
6010BPLSS4
6030LDX@#5
6050JSR#FB83
6070LDA@#20
6090STA(#90),Y
6110BIT#B001
6130BPLSS4
6150LDX@#5
6170JSR#FB83
6190DEC#9E
6210BNESS2
6230LDX@#10
6250JSR#FB83
6270LDA#90
6290CLC
6310ADC@7
6330STA#90
6350DEC#9F
6370BNESS1
6390JMPSS0
6410:SS4LDA@#34
6430STA#81
6440STA#98
6450LDA@#20
6470STA#80
6490JSRBB0
6510JSRLL6
6530INY
6550LDX@4
6560STX#98
6570:SS5CPX#9F
6590BEQSS6
```

:
:
:
:
:
:
:
:
:Flashing and moving cursor
:routine
:#90,#91 contain address of
:cursor
:
:
:
:
:
:Checks at each cursor position
:for depression of SHIFT
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:On exit #9F will contain
:pointer to year
:Store address of string in #80,
:#81, #98 is used as error flag
:
:
:
:Find name in data bank
:
:
:Clear error flag
:Move Y to point at data for
:year indicated


```
6610INY
6630INY
6650INY
6670DEX
6690JMPSS5
6710:SS6TYA
6730CLC
6750ADC#82
6770STA#82
6790BCCSS7
6810INC#83
6830:SS7RTS
6850:SS8LDY@2
6870:SS9LDA#339,Y
6890STA(#82),Y
6910DEY
6930BPLSS9
6950RTS
6970:WW0JSRWW12
6990LDY@0
7000LDA@#20
7010STA#82
7030LDA@#34
7050STA#83
7070:WW1LDA(#82),Y
7090STA(#80),Y
7110INY
7130CMP@#D
7150BNEWW1
7170LDX@4
7190STX#9C
7210LDA@#81
7230STA#DF
7250LDA@#20
7270:WW2STA#DE
7290LDX@3
7310STX#9B
7330LDA@#D
7350:WW3STA#E0
7370STY#9F
7390LDA@0
7410STA#9E
7430JSR#FB8A
7450JSRWW5
7470LDY#9F
7490LDA#9D
7510STA(#80),Y
7530INY
7550LDA#E0
7570CLC
7590ADC@5
7610DEC#9B
```

:
:
:
:
:
:Adjust #82,#83 so that they
:contain address of data for
:year indicated
:
:
:
:
:Store new data routine
:#339,#33A,#33B are the L.Bs of
:integer variables X,Y,Z
:
:
:
:Load #82,#83 with address of
:string
:
:
:Store string in data bank
:
:
:
:Load X with number of seasons
:
:set VDU address
:
:
:Load X with pointer to Home
:wins, Away wins and draws
:Set column
:
:Routine which prompts for data
:input under each heading for
:each season
:
:WW5 returns with data in #9D
:
:Store data in data bank
:
:
:Move prompt
:
:
:

```
7630BNEWW3
7650LDA#DE
7670CLC
7690ADC@#40
7710DEC#9C
7730BNEWW2
7750JSRLL34
7770:WW4LDA@#FF
7790LDX@0
7810STA(#80,X)
7830RTS
7850:WW5LDY#E0
7870LDA@#3F
7890STA(#DE),Y
7910:WW6JSR#FE94
7930CMP@#7F
7950BNEWW7
7970DEC#E0
7990DEC#9E
8010JMPWW5
8030:WW7CMP@#D
8050BEQWW8
8070JSR#FE52
8090AND@#0F
8110LDX#9E
8130STA#90,X
8150INX
8170STX#9E
8190CPX@3
8210BCCWW6
8230DEC#9E
8290:WW8LDX#9E
8310DEX
8330LDA#90,X
8350STA#9D
8370DEX
8390BMIWW11
8410LDY#90,X
8430:WW9LDA#9D
8450CLC
8470ADC@#A
8490STA#9D
8510DEY
8530BNEWW9
8550:WW10RTS
8570:WW11INC#E0
8590JMPWW10
8610:WW12LDA@#2A
8630STA#DE
8650LDA@#81
8670STA#DF
```

```
:Loop
:Change VDU line
:
:
:Loop until all seasons done
:
:Adjust #80,#81
:Store end of text marker
:
:
:
:Display "?"
:
:
:Wait for key to be pressed
:
:
:
:
:
:
:
:Output character
:Clear top 4 bits
:Load X with pointer to work
:space, store low 4 bits
:
:
:
:
:
:Convert input bits to 1 #
:number held in #9D
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:Reset cursor position and
:clear VDU, X points to number
:of lines, Y points to number
:of columns
```



```
8690LDX@4                :
8710:WW13LDY@#15         :
8730:WW14LDA@#20         :
8750STA(#DE),Y           :
8770DEY                  :
8790BPLWW14              :
8810LDA#DE               :
8830CLC                  :
8850ADC@#40              :
8870STA#DE               :
8890DEX                  :
8910BNEWW13              :
8930RTS                  :
8950:WW15JSR#FFED        :Save data to tape routine
8960CLC                  :Print RECORD TAPE and wait for
8970JSR#FC38             :key to be pressed
8990LDA@#83              :Set #80,#81 to contain start
9010STA#81               :address of data bank
9030LDY@0                :
9050STY#80               :
9070:WW16LDA(#80),Y      :Send byte to tape until end of
9080STA#8020             :text found, send visual
9090CMP@#FF              :display
9110BEQWW17              :
9130JSR#FC7C             :
9150INY                  :
9170BNEWW16              :
8934INC#81               :
9210JMPWW16              :
9230:WW17LDA@#FF         :
9250JSR#FC7C             :
9270RTS                  :
9290:KK0SEC              :Load data routine, print PLAY
9310JSR#FC38             :TAPE and wait for key to be
9330LDY@0                :pressed
9350LDA@#83              :
9362STA#81               :
9390STY#80               :
9410:KK1JSR#FBEE         :Get byte from tape until end
9430STA(#80),Y           :of data found, store in data
9450STA#8040             :bank, send visual display
9470CMP@#FF              :
9490BEQKK2               :
9510INY                  :
9530BNEKK1               :
9550INC#81               :
9570JMPKK1               :
9590:KK2RTS              :
9610:EE0LDY@0            :Find end of data routine
9630LDA@#83              :
9650STA#81               :
9670STY#00               :
```

```
9690:EE1LDA(#80),Y          :
9710CMP@#FF                 :
9730BEQEE2                  :
9750INY                     :
9770BNEEE1                  :
9790INC#81                  :
9810JMPEE1                  :
9830:EE2JSRLL34             :Exit with #80,#81 containing
9850RTS                     :address of 1st free location
9860:BB0LDA@#C2             :Load #84,#85 with return
9950STA#85                  :address, #C278 contains RTS
9960LDA@#78                 :
9970STA#84                  :
9980RTS                     :
10040:BB1LDA@#35            :Load #80,#81 with address of
10050STA#81                 :current match storage area
10060LDA@0                  :
10070STA#80                 :
10080RTS                    :
10100:BB2LDA@#81            :Load #81 with H.B of VDU
10110STA#91                 :address
10120RTS                    :
12000];R.                   :
```

PART NO: 2

ENTER IN TEXT AREA: #29
TEXT: 1.4K
VDU: .5K and up to 5.25K as data storage
M/C: -
EXECUTION: RUN

```
10REM POOLS PROGRAM                :
20DIMR3;@=0                        :
30P.$12'"          POOLS           :
PROGRAM"''                         :
40IN."DO YOU HAVE A DATA           :
BASE"$R                            :
50IF CH$R<>CH"Y"G.520              :
60IN."IS YOUR DATA BASE            :
LOADED"$R                          :
70IFCH$R=CH"Y" G.110               :
80P.$12"LOAD DATA BASE"''''        :
90LI.#338C                         :REM KKO
100P.''"DATA BASE LOADED"'         :
110IN."DO YOU WISH TO PREDICT      :
DRAWS"$R                           :
120IFCH$R<>CH"Y"G.220              :
125P.$12'                          :
130IN."HOW MANY MATCHES"Z          :
```



```
140W=15                            :
150IF Z<14 W=Z                     :
160LI.#30BD                        :REM LL13
170P.$12"MATCHES MOST              :
LIKELY TO DRAW ARE"'               :
180?#9F=W                          :
190LI.#3193                        :REM LL23
200LI.#FE94                        :
210P.$12                           :
220IN."DO YOU WISH TO              :
CHANGE DATA BASE"$R                :
230IFCH$R<>CH"Y" E.                :
240IN."ADD EXTRA DATA"$R           :
250IFCH$R<>CH"Y" G.310             :
260LI.#33AD                        :REM EE0
270G.a                             :
280P.$12"SAVE DATA"''              :
290LI.#3363                        :REM WW15
300P.'"DATA SAVED"'                :
305G.110                           :
310IN."CHANGE DATA"$R              :
320IFCH$R<>CH"Y"E.                 :
330P.$12                           :
340P.''"        UPDATE DATA        :
"                                  :
350P.''                            :
360IN."TEAM "$#3420                :
370P.''"       SEASON TO BE        :
UPDATED"'                          :
380P.'"   82/83  81/82             :
80/81  79/80"''''                  :
390P.''"PRESS SHIFT ON             :
INDICATION"                        :
400LI.#322C                        :REM SS0
410IF?#98<>4 G.485                 :
420P.$12$#3420" FOUND"'            :
430X=78+?#9F                       :
440P."SEASON                       :
19"X"/19"X+1''                     :
450IN."DRAWS"Z                     :
460IN."AWAYS"Y                     :
470IN."HOMES"X                     :
480LI.#329B                        :REM SS8
485P.''                            :
490IN."MORE"$R                     :
500IFCH$R=CH"Y" G.330              :
510G.280                           :
520!#80=#8300                      :
530aP.$12"       DATA BASE         :
ENTRY    "'                        :
540IN."ENTER NO OF TEAMS"X         :
550                                :
```

```
560F.N=1 TO X                          :
570?#E1=#00                            :
580P.''''''"(11SPACES)                 :
HOMES  AWAYS  DRAWS"                   :
590P.'"--(28 -ve SIGNS)--"             :
600P. "1982/1983"''                    :
"1981/1982"''"1980/1981"''             :
"1979/1980"                            :
610P.$30                               :
620?#DE=#60                            :
630P."(22 SPACES)"$13                  :
640?#E1=#80                            :
650P."TEAM NO"N                        :
660IN." NAME"$#3420                    :
670LI.#32A6                            :REM WWO
680?#DF=#80                            :
690?#DE=#40                            :
700?#E0=0                              :
710N.                                  :
720G.280                               :
```

LINES:-

290-430 Set up VDU and print HOME TEAM.
470-730 Home cursor, set up VDU and print AWAY TEAM.
790-990 Get character from keyboard, recognize delete.
1010-1590 Find team in data bank, return with Y pointing to #D, if not found print TEAM NAME NOT FOUND.
1650-1950 Clear line, delay befor returning.
1970-3090 Set up current match storage area, get HOME TEAM name, find in data bank, store in current match area, calculate probability.
3110-4130 Get AWAY TEAM, find in data bank, store in current match area, calculate probability of match, store factor in current match area,set bit 7 of next location, store end of text marker before returning.
4210-5310 Print matches in order of probability.
5350-5490 Add Y to #80,#81 subroutine.
5510-5750 Calculate return address subroutine.
5770-6390 Flash and move cursor until SHIFT pressed.
6410-6830 Find name in data bank, adjust #82,#83 to contain address of data for year indicated.
6850-6950 Store new data.
6970-8590 Set up data base routine. Store name in data bank, get data as ASCII code, convert to # number and store, store end of text marker on exit.
8610-8930 Clear VDU for next data input.
8950-9270 Save data to tape.
9290-9590 Load data from tape.
9610-9850 Find end of data routine.
9860-9980 Load #84,#85 with address to fetch RTS.
10040-10080 Load #80,#81 with address of data bank.
10100-10120 Load #91 with H.B VDU address.

Screenchase

The computer guides a cursor, you guide a cursor. If you hit your own trail or the computer's trail you will have to sit back and watch the computer cover as many stars as possible. You gain points for every star covered, you lose points for every star the computer covers. You move your cursor with keys 1,2,3 and 4 which will move your cursor Up, Forwards, Down and Backwards respectively.

PAGE 0 LOCATIONS USED:-
#80,#81 Address of Computer's cursor.
#88 - #8F VDU address data.
#90,#91 Used as address of computer's cursor during search for next valid move.
#92 - #95 Computer's valid direction flags.
#9A,#9B Address of Player's cursor.
#9E Flag to indicate player finished.
#9F Direction indicator for player's cursor.
SUBROUTINES USED:-
#FB8A #FE71

ENTER IN TEXT AREA: #29
TEXT: 3K
VDU: 6K
M/C: .47K
EXECUTION: RUN

```
10REM SCREEN CHASE5                    :
20DIMZZ59                              :
30F.N=0TO59;DIMP-1;ZZN=P;N.            :
40F.N=1TO2;P=#3800;GOS.a;N.            :
50Y=0                                  :
70CLEAR4                               :
80?#9B=ABSRND%(#98-#80)+#80            :
90?#81=ABSRND%(#98-#80)+#80            :
100?#9A=0                              :
120F.N=0 TO 3;N?#92=1;N.N              :
130N=ABSRND%4                          :
140N?#92=0                             :
145!#21C=#4B00                         :
150!#88=#80009820                      :
160!#8C=#971F7FFF                      :
170LI.#3800                            :
180F.I=1TO100;WAIT;N.I                 :
190P.$12                               :
200X=((?#21D*256)+?#21C)               :
220P."YOUR SCORE IS "X'                :
230IF X>Y G.280                        :
240P.'"HIGH SCORE IS"Y''               :
```

```
250IN."PLAY AGAIN "$L
260IF CH$L=CH"Y" G.70
270E.
280P.''"FAR OUT -- HIGH
SCORE"''
290Y=X
300G.240
500aI
510LDA@#80
520STA#9F
530LDX@#18
540:ZZ1LDA@#5F
550STA#9E
560LDY@#20
570LDA@#18
580:ZZ2STA(#9E),Y
590DEY
600BNEZZ2
610LDA@#7F
620STA#9E
630LDY@#20
640LDA@#3C
650:ZZ3STA(#9E),Y
660DEY
670BNEZZ3
680LDA@#9F
690STA#9E
700LDY@#20
710LDA@#18
720:ZZ4STA(#9E),Y
730DEY
740BNEZZ4
750INC#9F
760DEX
770BNEZZ1
773LDA@#FF
774STA#9E
790:ZZ5JSR#FB8A
795JSRZZ58
800LDY@0
810LDX@8
820:ZZ6LDA@#FF
830STA(#80),Y
840LDA#80
850CLC
860ADC@#20
870STA#80
880DEX
890BNEZZ6
900JSRZZ52
910JMPZZ11
1000:ZZ7LDA#80,X
```

```
:
:
:
:
:
:
:
:
:Set VDU display
:Print star in each block of 8
:lines x 1 byte
:
:Star consists of
:Bits 3,4 set line 4
:Bits 2,3,4,5 set line 5
:Bits 3,4 set line 6
:
:
:
:
:
:
:
:
:
:
:
:
:
:
:Set #9E to indicate valid
:player's cursor
:
:Deduct from score
:Print computer's cursor at
:address in #80,#81
:
:
:
:
:
:
:
:
:
:GOSUB Player's routine
:
:Move address in #80,#81 up by
```



```
1010CLC
1020ADC@1
1030CMP#88,X
1040BNEZZ8
1050LDA#8A,X
1060:ZZ8STA#80,X
1070LDA@0
1080STA#94,X
1090JMPZZ5
1200:ZZ9LDA#80,X
1210SEC
1220SBC@1
1230CMP#8C,X
1240BNEZZ10
1250LDA#8E,X
1260:ZZ10STA#80,X
1270LDA@0
1280STA#92,X
1290JMPZZ5
1300:ZZ11LDX@4
1302LDY@0
1305:ZZ12LDA#91,X
1308CMP@#80
1310BCCZZ13
1320SEC
1330SBC@#80
1340STA#91,X
1350:ZZ13DEX
1360BNEZZ12
1400LDX@4
1416:ZZ14LDA#91,X
1420BEQZZ15
1430DEX
1440BNEZZ14
1450:ZZ15 STX#96
1460CPX@4
1470BEQZZ27
1480CPX@3
1490BEQZZ21
1500CPX@2
1510BEQZZ28
1520LDX@0
1530:ZZ16STX#86
1540JSRZZ39
1550:ZZ17STX#87
1560LDX#86
1570LDA#90,X
1580SEC
1585SBC@1
1590CMP#8C,X
1600BNEZZ18
1610LDA#8E,X
```

```
:1 block if X=1, forwards by 1
:block if X=0, check for top or
:extreme right of VDU and
:adjust accordingly, Set
:direction indicator to 0
:
:
:
:Move address in #80,#81 down 1
:line if X=1, backwards by 1
:block if x=0, check and adjust
:if bottom or extreme left of
:VDU, set direction indicator
:to 0
:
:
:
:Clear bit 7, if set, of
:direction indicators
:
:
:
:
:
:
:
:Find 1st direction indicator
:set to 0
:
:
:
:Compare X and goto indicated
:movement
:
:
:
:
:
:
:Find next valid move routine,
:look up if X=1, backwards if
:X=0
:
:
:
:
:
```

```
1620:ZZ18STA#90,X
1630LDX#87
1640LDA(#90),Y
1650CMP@#AO
1660BCSZZ19
1670INX
1680CPX@1
1690BNEZZ17
1700:ZZ19TXA
1710LDX#86
1720STA#92,X
1730CMP@1
1740BEQZZ20
1750JMPZZ29
1760:ZZ20JMPZZ9
1900:ZZ21LDX@O
1910:ZZ22STX#86
1920JSRZZ39
1930:ZZ23STX#87
1940LDX#86
1950LDA#90,X
1960CLC
1970ADC@1
1980CMP#88,X
1990BNEZZ24
2000LDA#8A,X
2010:ZZ24STA#90,X
2020LDX#87
2030LDA(#90),Y
2040CMP@#AO
2050BCSZZ25
2060INX
2070CPX@1
2080BNEZZ23
2090:ZZ25TXA
2100LDX#86
2110STA#94,X
2120CMP@1
2130BEQZZ26
2140JMPZZ29
2150:ZZ26JMPZZ7
2160:ZZ27LDX@1
2170JMPZZ22
2180:ZZ28LDX@1
2190JMPZZ16
2300:ZZ29LDX#96
2310LDA#91,X
2320CLC
2330ADC@#80
2340STA#91,X
2350LDX@4
2360:ZZ30LDA#91,X
```

```
:
:
:
:
:
:
:
:
:
:
:
:
:
:If X = 1 move valid
:
:
:
:Look down if X = 1, look
:forwards if X = 0
:
:
:
:
:
:
:
:
:
:
:
:
:
:
```

```
3770CMP@#9B
3790BNEZZ49
3810LDA@#80
3830:ZZ49STA#9B
3850:ZZ50LDY@0
3870LDA(#9A),Y
3890CMP@#A0
3920BCSZZ51
3940JMPZZ40
3960:ZZ51LDA@0
3980STA#9E
4000RTS
4020:ZZ52LDA#9E
4040BNEZZ53
4060RTS
4070:ZZ53JSRZZ56
4080JSR#FE71
4100CPY@#FF
4120BNEZZ55
4140:ZZ54LDA#9F
4160BEQZZ46
4180CMP@3
4200BEQZZ44
4220CMP@2
4240BEQZZ48
4260JMPZZ42
4280:ZZ55TYA
4300SEC
4320SBC@#11
4340CMP@4
4360BCSZZ54
4380STA#9F
4400JMPZZ54
4420:ZZ56LDA#21C
4440CLC
4460ADC@50
4480STA#21C
4500BCCZZ57
4520INC#21D
4540:ZZ57RTS
4560:ZZ58LDA#21C
4580SEC
4600SBC@25
4620STA#21C
4640BCSZZ59
4660DEC#21D
4680:ZZ59RTS
60001;R.
```

```
:
:
:
;
:Check new position, if block
:has been set then set player
:finished flag
:
:
:
;
:
:See if player finished
:
:if so return
:Adjust score
:See if key pressed
:
:If so GOTO ZZ55
:Otherwise get current
:direction and go specified
:routine
:
:
:
;
:Key pressed routine, convert
:to 0-3, if greater than,
:ignore and GOTO ZZ54,
:otherwise change direction
:indicator and GOTO ZZ54
:
:
:Player's score increase
:routine
:
:
:
:
:
:Player's score decrease
:routine
:
:
:
:
:
:
```


Subroutines Used

#C2F2	Interpreter entry point.
#C4F6	Later entry point to above to keep loops intact.
#C589	Convert hex number in #16,#25,#34,#43 to decimal and print to VDU in specified field width.
#C9D8	Standard ON ERROR routine.
#CD0F	Get characters from K/B, store in buffer. Exit on RETURN or when buffer full.
#CD14	As above but will not move cursor.
#F7D1	Prints following codes as a string until first code with bit 7 set is found.
#F7FD	Print a space.
#F802	Print number in Accum as two hex numbers.
#F87E	Convert ASCII code in Accum to hex low four bits.
#FB83	Wait for X/60 secs.
#FB8A	Wait for 1/10 sec.
#FBEE	Get byte from tape.
#FC38	Print RECORD TAPE if carry clear, PLAY TAPE if carry set.
#FC7C	Save byte to tape.
#FD1A	Sound bell.
#FD69	Clear VDU and home cursor.
#FE52	Print ASCII code in Accum to VDU.
#FE71	Scan keys, return with key number in Y.
#FE94	Wait for key to be pressed and return with ASCII code in Accum.
#FFED	Output a carriage return and line feed.

M/C Instructions

ADDRESSING MODES:-

A	Accumulator
#2800	Absolute
#90	Zero page
@#FF	Immediate
#2800,X	Absolute,X
#2800,Y	Absolute,Y
(#80,X)	Indirect,X
(#80),Y	Indirect,Y
#90,X	Zero page,X
#90,Y	Zero page,Y
(#9F)	Indirect

ADC — ADD WITH CARRY

```
ADC#2800
ADC#90
ADC@#FF
ADC#2800,X
ADC#2800,Y
ADC(#80,X)
ADC(#80),Y
ADC#90,X
```

AND — LOGICAL AND

```
AND#2800
AND#90
AND@#FF
AND#2800,X
AND#2800,Y
AND(#80,X)
AND(#80),Y
AND#90,X
```

ASL — ARITHMETIC SHIFT LEFT

```
ASLA
ASL#2800
ASL#90
ASL#2800,X
ASL#90,X
```

BCC — BRANCH ON CARRY FLAG CLEAR

BCS — BRANCH ON CARRY FLAG SET

BEQ — BRANCH IF EQUAL TO ZERO

BIT — COMPARE MEMORY WITH ACCUMULATOR

BMI — BRANCH ON MINUS

BPL — BRANCH ON PLUS

BRK — BREAK

BVC — BRANCH ON OVERFLOW CLEAR

BVS — BRANCH ON OVERFLOW SET

CLC — CLEAR CARRY FLAG

CLD — CLEAR DECIMAL FLAG

CLI — CLEAR INTERRUPT FLAG

CLV — CLEAR OVERFLOW FLAG

CMP — COMPARE TO ACCUMULATOR

```
CMP#2800
CMP#90
CMP@#FF
CMP#2800,X
CMP#2800,Y
CMP(#80,X)
CMP(#80),Y
CMP#90,X
```

CPX — COMPARE TO X

```
CPX#2800
CPX#90
CPX@#FF
```

CPY — COMPARE TO Y

```
CPY#2800
CPY#90
CPY@#FF
```

DEC — DECREMENT MEMORY

```
DEC#2800
DEC#90
DEC#2800,X
DEC#90,X
```

DEX — DECREMENT X

DEY — DECREMENT Y

EOR — EXCLUSIVE-OR

```
EOR#2800
EOR#90
EOR@#FF
EOR#2800,X
EOR#2800,Y
EOR(#80,X)
EOR(#80),Y
EOR#90,X
```

INC — INCREMENT MEMORY

```
INC#2800
INC#90
INC#2800,X
INC#90,X
```

INX — INCREMENT X

INY — INCREMENT Y

JMP — JUMP TO ADDRESS

```
JMP#2800
JMP(#9F)
```

JSR — JUMP TO SUBROUTINE

LDA — LOAD ACCUMULATOR

```
LDA#2800
LDA#90
LDA@#FF
LDA#2800,X
LDA#2800,Y
LDA(#80,X)
LDA(#80),Y
LDA#90,X
```

LDX — LOAD X

```
LDX#2800
LDX#90
LDX@#FF
LDX#2800,Y
LDX#90,Y
```


LDY — LOAD Y

```
LDY#2800
LDY#90
LDY@#FF
LDY#2800,X
LDY#90,X
```

LSR — LOGICAL SHIFT RIGHT

```
LSRA
LSR#2800
LSR#90
LSR#2800,X
LSR#90,X
```

NOP — NO OPERATION

ORA — INCLUSIVE-OR

```
ORA#2800
ORA#90
ORA@#FF
ORA#2800,X
ORA#2800,Y
ORA(#80,X)
ORA(#80),Y
ORA#90,X
```

PHA — PUSH ACCUMULATOR ON STACK

PHP — PUSH PROCESSOR STATUS ON STACK

PLA — PULL ACCUMULATOR FROM STACK

PHP — PULL PROCESSOR STATUS FROM STACK

ROL — ROTATE LEFT

```
ROLA
ROL#2800
ROL#90
ROL#2800,X
ROL#90,X
```

ROR — ROTATE RIGHT

```
RORA
ROR#2800
ROR#90
ROR#2800,X
ROR#90,X
```

RTI — RETURN FROM INTERRUPT

RTS — RETURN FROM SUBROUTINE

SBC — SUBTRACT WITH CARRY

```
SBC#2800
SBC#90
SBC@#FF
SBC#2800,X
SBC#2800,Y
SBC(#80,X)
SBC(#80),Y
SBC#90,X
```

SEC — SET CARRY FLAG

SED — SET DECIMAL MODE

SEI — SET INTERRUPT DISABLE

STA — STORE ACCUMULATOR IN MEMORY

```
STA#2E00
STA#90
STA#2800,X
STA#2800,Y
STA(#80,X)
STA(#80),Y
STA#90,X
```

STX — STORE X IN MEMORY

```
STX#2800
STX#90
STX#90,Y
```

STY — STORE Y IN MEMORY

```
STY#2800
STY#90
STY#90,X
```

TAX — TRANSFER ACCUMULATOR INTO X

TAY — TRANSFER ACCUMULATOR INTO Y

TSX — TRANSFER STACK POINTER INTO X

TXA — TRANSFER X INTO ACCUMULATOR

TXS — TRANSFER X INTO STACK POINTER

TYA — TRANSFER Y INTO ACCUMULATOR

PONTOON...
THE COMPUTER CHEATS!!

It looks at your score, if greater or equal it exits as it has won, if less it looks at the next card, adds that to it's score and as long as this score is less than 22 it displays the card. Only if the next card will bust will the computer allow you to win.