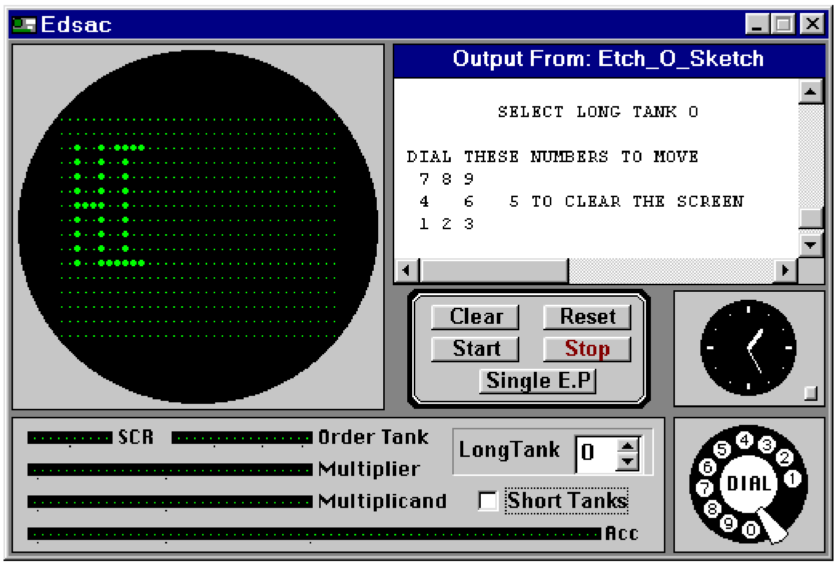
Chris Jenkins 9801604

Etch-O-Sketch

A drawing program that bares more than a passing resemblance to that children's favourite, Etch-A-Sketch. The user can draw a picture on Long Tank 0 by moving a 'cursor'. This cursor can be moved in steps of one vertically, horizontally and diagonally. The screen may also be cleared.

Example output from Etch-O-Sketch:



```
       SELECT LONG TANK 0

DIAL THESE NUMBERS TO MOVE
 7 8 9
 4   6    5 TO CLEAR THE SCREEN
 1 2 3
```

Structure of the program tape:

T 56K

M3

space PK

T 56K

PLOT_POINT

space PZ

T 128K

CLS

space PZ

MASTER

EZPF

Two custom subroutines were written for Etch-O-Sketch:

Name	***Purpose***	***Location***
PLOT_POINT	Plots a point at the specified location	56
CLS	Clears long tank 0	128

PLOT_POINT:

PLOT_POINT plots a point which can be seen by switching the CRT display to long tank 0. The X parameter is placed into the location L + 50 and the Y parameter into location L + 51, where L is the location into which PLOT_POINT is loaded. The coordinate origin is located at the bottom right of the screen, with the x axis pointing to the left and the y axis pointing upwards. This was chosen to make the implementation of the subroutine easier.

The subroutine plots a point by taking the long int 1 (stored in location 8M) and shifting it to the left x times to give a long int pattern that is stored in location 2M. It then checks to see if the bit in long tank 0 that we wish to set has already been set. It does this by ANDing the pattern in 2M with the appropriate row of long tank 0. If the result is non-zero then the bit has already been set and control is returned to the caller. If the result is zero then the pattern is added to the appropriate row of long tank 0 and returns control to the caller.

Annotated Listing for PLOT_POINT:

```
              GK            set θ parameter

              T 47K         set M parameter - local variables and parameters live here

              P 50θ

              TZ

       0      A 16M         plant return link

       1      T 44θ

       2      XF

                            first, shift the pattern by x steps to get the right bit pattern to
                            draw onto the screen

118->  3      T 10#M        pattern = INITIAL_PATTERN

       4      A 8#M

       5      T 2#M

       6      XF

       7      A 0M          i = x - 1

       8      S 8M

       9      U 4M

       10     XF            if i = 0 then don't execute the loop body at all

       11     G 20θ

                            loop

19->   12     T 10#M

       13     A 2#M         left shift pattern by one

       14     L 0D

       15     T 2#M
```

```
        16      A 4M            decrement i

        17      S 8M

        18      U 4M

        19      E 12θ           loop while i is not -ve

11->    20      XF

                                now, check if there is already a point in the place that we want to
                                draw to - if there is one, then we don't need to draw another one

        21      T 10#M          take the initial SMC instruction opcode and add y*4 to it

        22      A 1M

        23      L 1F            acc = y*4

        24      U 15M

        25      A 12M

        26      T 27θ

        27      H 0D            SMC point 1 - masked_target = the target memory ANDed with pattern

        28      C 2#M

        29      U 6#M

        30      G 43θ           if masked_target is -ve then we don't need to do anything, so jump to
                                the end of the subroutine

        31      S 8#M           if masked_target > 0 then jump to the end of the subroutine

        32      E 43θ

                                now draw the pattern onto long tank 0 by adding pattern onto the
                                appropriate memory location

        33      T 10#M          take SMC point 2 and add y*4 to its initial value

        34      A 15M

        35      A 13M

        36      T 40θ

        37      A 15M           take SMC point 3 and add y*4 to its initial value

        38      A 14M

        39      T 42θ

                                target = target + pattern

        40      A 0D            SMC point 2

        41      A 2#M

        42      T 0D            SMC point 3

30,32-> 43      T 10#M          clear the accumulator so that we jump back to the caller

        44      EF

        45      XF              padding

        46      XF

        47      XF

        48      XF

        49      XF
```

```
M 0     P 0F          x - the caller should set this

1       P 0F          y - the caller should set this

2       P 0F          pattern (long int) - variable used to manipulate the pattern that
                      will

3       P 0F          be added to the screen

4       P 0F          i - used as a counter

5       P 0F          filler

6       P 0F          masked_target (long int ) - used to store the result of masking off
                      all bits

7       P 0F          other than the one that we are about to set to see if that bit is
                      already set

8       P 0D          INITIAL_PATTERN (long int constant) = 1. This pattern is left-shifted
                      to

9       P 0F          get the pattern that we draw onto long tank 0

10      P 0F          two words to clear the acc into when we want to clear it

11      P 0F

12      H 0D          the initial value of SMC point 1

13      A 0D          the initial value of SMC point 2

14      T 0D          the initial value of SMC point 3

15      P 0F          y*4 kept here

17      U 2F          to help in planting return link
```

CLS:

CLS clears long tank 0 by clearing the accumulator and then then transferring the contents of the accumulator to each long word in long tank 0, one at a time. Control is then returned to the caller.

Annotated Listing for CLS

```
        GK

0       A 19θ         plant return link

1       T 18θ

2       T D

3       T 2D

4       T 4D

5       T 6D

6       T 8D

7       T 10D

8       T 12D

9       T 14D

10      T 16D

11      T 18D

12      T 20D

13      T 22D

14      T 24D

15      T 26D
```

```
       16      T 28D
       17      T 30D
       18      E F            return link
       19      U 2F           constant to help planting return link
```

Master Routine:

The master routine does the majority of the work. It clears long tank 0 and then enters the main loop.

In the main loop, a point is plotted at the coordinates (x, y) where x is stored in the location 0M and y is stored in the location 1M. The routine then stops and waits for the user to dial input on the rotary dial. Depending on the number dialled, the routine either moves the cursor in the desired direction (and therefore increments or decrements x and/or y), or clears the screen by calling CLS. The routine then loops back to the start of the main loop.

Annotated Listing for Master Routine

```
               GK
               T 47K          set M parameter
               P 119θ
               TZ
               T 45K          set H parameter - this points to PLOT_POINT
               P 56F
               TZ
       0       XF
       1       A 1θ           clear long tank 0 using CLS
       2       G 128F
                              main_loop
CLS->  3       T 2M           clear  acc
       4       A 0M           plot a point at (x, y)
       5       T 50H
       6       A 1M
       7       T 51H
       8       A 8θ
       9       G 0H           call PLOT_POINT
PLOT_POINT->
16, 36->
       10      T 2M           clear acc and wait for input
       11      ZF
       12      U 3M           store the input
       13      XF
       14      R 0D           divide by two to get the value entered
       15      S 4M
       16      G 10θ          input = 0 => no user input so re-read the input
       17      S 4M
       18      G 37θ          input = 1 => move down and to the left
       19      S 4M
```

```
        20      G 45θ           input = 2 => move down
        21      S 4M
        22      G 50θ           input = 3 => move down and to the right
        23      S 4M
        24      G 58θ           input = 4 => move left
        25      S 4M
        26      G 63θ           input = 5 => clear screen
        27      S 4M
        28      G 67θ           input = 6 => move right
        29      S 4M
        30      G 72θ           input = 7 => move up and left
        31      S 4M
        32      G 80θ           input = 8 => move up
        33      S 4M
        34      G 85θ           input = 9 => move up and to the right
        35      S 4M
        36      G 10θ           input = 10 => user pressed 0, which isn't allowed => re-enter input
18->    37      T 2M            move down and to the left
        38      A 0M            increment x (remember, the x axis points to the left)
        39      A 4M
        40      T 0M
        41      A 1M            decrement y
        42      S 4M
        43      T 1M
        44      E 93θ           break out of this block
20->    45      T 2M            move down
        46      A 1M            decrement y
        47      S 4M
        48      T 1M
        49      E 93θ           break
22->    50      T 2M            move down and to the right
        51      A 0M            decrement x
        52      S 4M
        53      T 0M
        54      A 1M            decrement y
        55      S 4M
        56      T 1M
        57      E 93θ           break
24->    58      T 2M            move to the left
        59      A 0M            increment x (remember, the x axis points to the left)
        60      A 4M
        61      T 0M
        62      E 93θ           break
26->    63      T 2M            clear the screen
        64      A 64θ           clear long tank 0 using CLS
        65      G 128F
        66      E 93θ          break
28->    67      T 2M           move to the right
        68      A 0M           decrement x
        69      S 4M
        70      T 0M
        71      E 93θ          break
30->    72      T 2M           move up and to the left
        73      A 0M           increment x (remember, the x axis points to the left)
        74      A 4M
        75      T 0M
        76      A 1M           increment y
        77      A 4M
        78      T 1M
        79      E 93θ          break
32->    80      T 2M           move up
        81      A 1M           increment y
        82      A 4M
        83      T 1M
        84      E 93θ          break
34->    85      T 2M           move up and to the right
        86      A 0M           decrement x
        87      S 4M
        88      T 0M
        89      A 1M           increment y
        90      A 4M
        91      T 1M
        92      E 93θ          break

                               check that x and y are still within the bounds of our 'screen'
44, 49, 57, 62, 66, 71, 79, 84, 92->
        93      T 2M           if x >= 35 then x = 34
        94      A 0M
        95      S 5M
        96      G 100θ
        97      T 2M
        98      A 7M
        99      T 0M

96->    100     T 2M           if x < 0 then x = 0
        101     A 0M
        102     E 105θ
        103     T 2M
        104     T 0M
```

```
102->  105     T 2M            if y >= 16 then y = 15

       106     A 1M

       107     S 6M

       108     G 112θ

       109     T 2M

       110     A 8M

       111     T 1M


108->  112     T 2M            if y < 0 then y = 0

       113     A 1M

       114     E 117θ

       115     T 2M

       116     T 1M


114->  117     T 2M            loop to main_loop

       118     E 3θ


       0M      P 8F            x = 16

       1       P 4F            y = 8

       2       P 0F            dump accumulator here

       3       P 0F            user input stored here

       4       P 0D            =1

       5       P 17D           =35

       6       P 8F            =16

       7       P 17F           =34

       8       P 7D            =15
```