Creating a 4-digit LCD Counter:

Concepts used in this program:

- Algorithms
- Clones
- Conditional looping (repeat until)
- Conditionals (IF)
- Conditions: YES/NO questions
- Costume-based animation
- Lists
- Logic operators (AND, OR, NOT)
- Nested looping
- Procedures with inputs
- Relational operators (=, <, >)
- Simple looping (repeat, forever)
- Stopping scripts
- String operations (join, letter, length of)
- String variables
- Synchronization using broadcasting
- User input (ASK)
- Variables - local/global scope
- Variables – numbers
- Variables as sliders

LCD Counter:

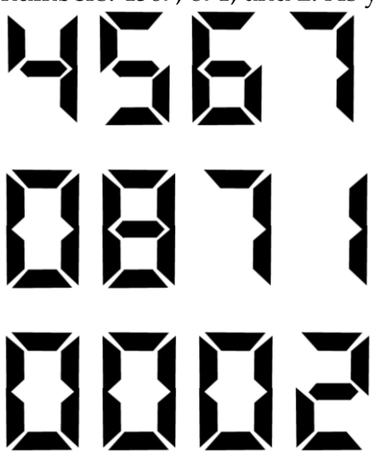
Demo video: https://youtu.be/eNVOWmxLdFk

You have seen this type of counters at gas stations. See below for an example:

A single digit in this display looks like this:

It consists of 7 LCD bars which turn ON or OFF to show digits 0 to 9. In the picture above, all 7 bars are ON to display the digit 8.

In this program, we want to create a 4-digit counter that will count from 1 to whatever number (less than 10000) the user provides. In the picture below the counter displays 3 different numbers: 4567, 871, and 2. As you can see, if the number doesn't require all 4 digits we show 0s.



Version 1:

This version will have the following features:

- Use a 7-bar display to show a single digit 0 thru 9.

Use a 7-bar display to show a single digit 0 thru 9:

Each digit consists of 7 bars which when all lit up show the figure of 8. By turning ON a subset of these bars, we can show numbers 0 to 9.

If we number the bars in some sequence 1 thru 7 and encode it in a 7-bit pattern, it's a matter of simply mapping each digit to the pattern. We will take the clockwise sequence starting with the top bar and ending with the bar in the center. In this sequence, 1 would map to 0110000, 4 would map to 0110011, and so on. See below the complete map:

0: 1111110
1: 0110000
2: 1101101
3: 1111001
4: 0110011
5: 1011011
6: 1011111
7: 1110000
8: 1111111
9: 1111011

The algorithm to display a digit would be as follows:

Algorithm show digit N (0 to 9):

```
Get the 7-bit mapping for N. (All mappings will be stored in a list)
Repeat 7
      Scan the mapping left to right
      If bit is 0
            Use the "invisible" costume for the bar
      Else
            Use the "visible" costume for the bar
      End-if
End-repeat
```

Create a mechanism to light up a digit on Scratch screen:

Each of the 7 bars would be a separate sprite. Each bar would have fixed X and Y positions relative to some point on the screen – for example, the X and Y of the top bar.

Lighting up a bar would be done by sending a broadcast and a "status" variable. The broadcast message would indicate the bar in question, e.g. 1, 2, 3, etc. Each bar would process its own message and look at the "status" variable to decide whether to turn ON or OFF.

Program: LCD-counter-1: https://scratch.mit.edu/projects/317241592/

Version 2:

This version will add the following features:

- Ask user for a number and convert it to a 4-digit string.
- Create a mechanism to display 4 digits in Scratch:

Design:

Ask user for a number and convert it to a 4-digit string.

Since we have 4 LCD digits, we need to pad smaller numbers with 0s. For example, 5 would be shown as 0005, 254 would be shown as 0254, etc.

Algorithm to convert N to a 4-digit string:

```
Input: 0 <= N <= 9999
Output: string scount
Copy N to scount as it is
If N has 4 digits
      Do nothing
If N has 3 digits
      Prepend a "0" to scount
If N has 2 digits
      Prepend a "00" to scount
If N has 1 digit
      Prepend a "000" to scount
```

Once we have a 4-digit string, we can simply examine each digit left to right, get its "bit mapping" and light up its LCD bars. To indicate which digit is being worked on, we can use a global variable "current position".

Create a mechanism to display 4 digits in Scratch:

We will need 4 sets of each bar – which can be accomplished using clones. Each set will contain 7 bars. Each set will belong to a position: during clone creation, set "myposition" to "current position" (e.g. 1, 2, 3, 4).

Algorithm for clone creation phase:

- Hard-code X for "bar1" for 1st digit. X of "bar1" of subsequent digits would be relative to this X.
- Hard-code Y for "bar1". Y would be the same for "bar1" of all digits.
- Set "current position" to which digit we are creating, i.e. 1, 2, 3, or 4).
- Upon creation:
    - Each clone will use a position relative to "bar1" within that digit. This relation would be identical for a particular bar in all digits.
    - Each clone with set private variable "myposition" of every bar to "current position".

During the "lighting" phase (see above), each bar would examine which digit is being lit up by examining "current position". It will process the message only if its own digit is being lit up.

Program: LCD-counter-2: https://scratch.mit.edu/projects/317242071/

Version 3:

This version will add the following features:

- Implement a counter that counts from 1 to N (user supplied).

Implement a counter that counts from 1 to N:

Algorithm for the counter:

```
We want to count from 1 to N.
I = 1
Repeat N
      Scount = use the algorithm to convert I to 4-digit string
      For each digit D in Scount (left to right):
            Set "current position" (to 1, 2, 3, etc.)
            Light up digit D (use the algorithm above)
      End
      I = I + 1
End-repeat
```

Program: LCD-counter-final: https://scratch.mit.edu/projects/317242226/

Version 4:

You will notice that our counter is very slow. It counts almost like a clock although we haven't used any delay (wait) statements. To count up to 100, my program takes about 92 seconds!

The reason for this is that we go about lighting up each digit separately and that too one after the other. Each bar in each digit is examined and is turned ON or OFF. This results in a very slow counter.

If we make the following change, the counter should speed up considerably:

- Each digit is independent of the others, so, we can light up digits concurrently instead of one after the other.

Process each digit in parallel:

To achieve this, we will split the overall work of displaying a 4-digit number in the following two steps:

- Inform each set of LCD bars (that belong to a particular digit) which digit (0 thru 9) it is supposed to display. This we will one after the other since we have the scan the 4-digit number.
- Inform each set of LCD bars to go ahead and do all the work to display the digit. This we will do in parallel.

Algorithm to set a digit for each set of bars (for example, say 5 at position 1):

```
Set current position = 1, set current digit = 5
Send broadcast to all

Each clone upon receiving the broadcast:
If myposition = current position
      Set mydigit = current digit
```

Algorithm to display all 4 digits:

```
Send a general broadcast
Every bar on receiving the broadcast:
Get map for "mydigit"
If your bit is 1
      Turn on
Else
      Turn off
```

With this change, my program takes about only 3.5 seconds to count up to 100!

Program: LCD-counter-final-optimized: https://scratch.mit.edu/projects/317242443/

Author: Abhay B. Joshi (abjoshi@yahoo.com)

Last updated: 2 July 2019