Life on the PIC

by Jonathan Woodrow, February 2011

Abstract

Cellular automata work by taking a grid of ‘cells’ and applying the same action to each cell in turn. Using the greater processing power of the 16 bit PIC micros Jonathan demonstrates the cellular automata action using a variety of visually impressive methods printed to an EB058 128x128 Graphical LCD. Read on to find out how you could create life on the PIC.

Requirements

Software:

- Professional licence of Flowcode v4 for dsPIC/PIC24

Hardware:

- dsPIC/PIC24 EB064 Multiprogrammer board
- EB058 Graphical LCD board

Cellular automata work, at their simplest, by taking a grid of 'cells' and applying the same action to each cell in turn. Each cell is connected to others in some way and is capable of reading the state or value of its neighbours, writing back to itself and (if you want to cheat) writing to its neighbours.

Unfortunately the number of cells that require processing each frame in a typical automation requires more processing power than most 8-bit PICs are capable of, especially when each cell also needs to be made suitable for output, usually visually.

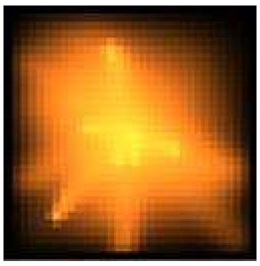
A complex automation could easily contain several Kb of memory for the cell array and needs to be capable of processing thousands of cells in a fraction of a second to manage a decent frame rate.

With the 16-bit PIC the memory and processing power now exists to produce an automation of quite a high resolution, and the addition of a 128x128 LCD display allows these automata to easily be displayed graphically.

Simple plasma

One of the simplest automata is a ‘plasma’ demo. The typical automation for plasma displays is, for each cell, to average the cell with its neighbours and write back to the original.

The code provided here has an additional optimisation that the LCD is drawn to only if the plasma cell has changed during that frame. This allows the LCD to run at a reasonable speed producing a massive boost for the program.



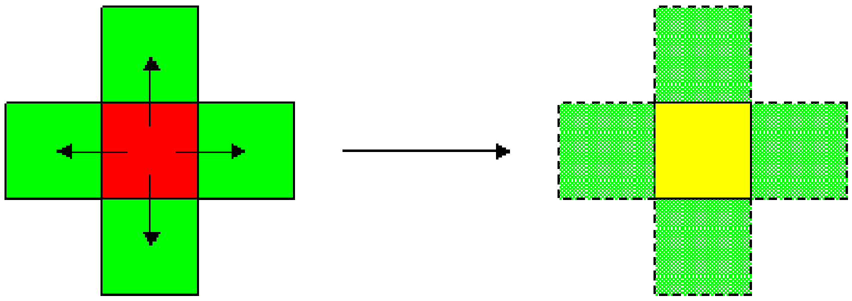
Fire plasma

All that is then required is a method to ‘stimulate’ the plasma, and the way this is done can have wide ranging effects on what the plasma is perceived to be.

Two demonstrations of this are provided here, ‘rain’ and ‘fire’. They both work on exactly the same principle, and in fact all that has changed in the programs is the colour of the plasma and the method to stimulate it.

Both programs work by taking a 2D array of cells and, for each cell, averaging the value of the neighbours then averaging this with the cell itself, storing that value back, so:

```
cell = (mean(neighbours) + cell) / 2
```


Blending plasma

Creating a Petri-dish

Now the basic plasma is done, we can build on this by adding more functionality into the cell activity. The aim is to create a series of cells that interact randomly but give off the impression of some rudimentary intelligence.

Each cell is represented by a single byte – 8 bits. We can split these bits up to represent different functionality, and do so as follows:

Sync	Species			Life-force / energy			
0	1	2	3	4	5	6	7

If all the bits are set (0xFF) then the cell is assumed to be ‘solid’. If the cells energy is zero, it is assumed to be vacant.

This allows our Petri-dish to contain 7 different ‘species’ of life. Each cell has an energy that may be depleted during interactions with other cells or restored when ‘resting’.

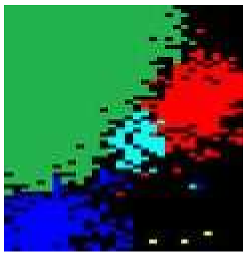
An additional bit is necessary for frame-synchronisation. For this program, we resort to the cheat mentioned above in that cells not only write to themselves, but also their neighbours. In order to prevent the same ‘life-form’ from being processed more than once each frame, we toggle the ‘sync’ bit each frame on each cell and only process cells whose bit does not match a global sync bit.

For this to be a cellular automation, we need to define rules applied to each cell in the grid one at a time. For this demo, these rules are:

If a cell has not been processed (sync != global sync) and is a 'species' cell with energy:

- Mark the cell as synchronised (regardless of the action)
- Take a random point adjacent to, diagonal to or the same as the current cell.
- If that point is the same as the current cell, 'regenerate' any lost energy.
- If that point is solid, exit (do nothing).
- If that point is empty, move to that cell and synchronise the destination.
- If that point is occupied by a different species, half both cells energy and exit.
- If that point is occupied by the same species, half both cells energy select a random point and 'spawn' the cell into it if that point is empty

By 'synchronise', it is meant that the sync bit of the cell is set to that of the global sync bit.



Cells then need to have some colouring mechanism. The simplest here is to use the primary colours (red, green, yellow, blue, magenta and cyan) for each species – the final available species is reserved as a 'virus' cell – see right

This results in a somewhat more complex system than the simple averaging plasma and will cause species to 'breed' and encroach on other species territory.

The final stage in this program is to 'stimulate' the plasma, in this case randomly spawning new clusters of life every now and again.

As an extension, plugging a switch array into port BH on the board provides some additional control:
· The bottom switch causes spawning to increase.
· The bottom two switches cause a 'genetically engineered' virus to be spawned (using species 0).

This virus species has slightly different rules to the other cells, and will 'assimilate' neighbours to its own species, but combust on contact with a cell of the same species. This causes the virus to annihilate species then burn out.

Edit: For those interested, here are the automation Flowcodes. They are all for the 33FJ128GP802 chip, but this is easily changed and there is nothing specific to that chip in the code. Matrix has just received stock of Flowcode for Pic16, so if you haven't already, get your orders in!

Further reading

Below are some links to other resources and articles on related subjects, and technical documentation relating to the hardware used for this project...

Flowcode:	http://www.matrixmultimedia.com/flowcode.php
Eblocks:	http://www.matrixmultimedia.com/eblocks.php
Learning Centre:	http://www.matrixmultimedia.com/lc_index.php
User Forums:	http://www.matrixmultimedia.com/mmforums
Product Support:	http://www.matrixmultimedia.com/sup_menu.php