**Release: software for post 12.**

The machine pictured is a UK machine called the “Nascom 1” from around the late 1970s.

Many of these machines hadn’t quite got as far as the built in BASIC interpreters of the Pet/Apple/Tandy era, they had ROM monitors. (The Nascom was a Z80 based m/c with a TV display)

I’ve done some work on a monitor, which hopefully I’ve remembered to upload to github.

I thought I might do something a bit different, rather than just have simple commands to set bytes, run programs etc. \*if\* it will fit I will add in a sort of pseudo command line assembler so you can type things like

**LD @P1 47**

Into the command line and it will assemble the code (C5 47) for you. I got the idea from another British SC/MP machine, the Scrumpi 3, which had the option of setting label values, so you could calculate jump offsets etc. automatically. It’s a very basic assembler, obviously it doesn’t do forward references which is the big minus point ☺ However, it would take a lot of the hassle out of entering programs in machine code ; it’s a kind of half way house between the Nascom1 or Apple 1 monitors and a proper assembler.

Much depends on the space. My original design only allowed 2k of ROM on the basic machine – the idea is that this machine should work with 2k ROM, 128 bytes of RAM and the video display, so it might not fit. I shall see ☺

As a move towards this, I have written code to convert typed input into 15 bit numbers (representing up to 3 x 5 bit ASCII characters) which I can use in a look up table, which can direct the monitor either to execute code, or to assemble it. If the baby assembler doesn’t fit, it’s a rather long winded but harmless way of decoding the command line.

e.g.

ASCII A L Z

Hex 41 4C 5A

5-Bit 01 0C 1A

Binary 00001 01100 11010

15 bit 000010110011010

If I’ve got the space, I’ll create a little disassembler as well. But I suspect it won’t fit. Can’t have everything ☺

When I come to write the HLL, I’ll have to rewrite the I/O as well. This one is designed for 128 bytes of RAM as a minimum – this is the minimum set up of the machine. This means, because of the write only VDU, I cannot have scrolling (I need to save all the characters output to scroll), so what the monitor does is jump back to the top from the bottom, keeping the current line clear as it goes.

All the code from here is original. I had a couple of sheets of code, one of which was the basic monitor which I copied and tweaked slightly. The other one is just the middle of something, no real idea what, and it’s useless on its own. I think most of the code was thrown because it just looked like writing , maybe ?