**Release: software post 27**

I realised something this morning. My expression evaluator seems to work fine, except for one little flaw. I’ve forgotten it allows terms using character constants (e.g. ‘B’ which evaluates to its ASCII value, 66).

So I fixed that dozy mistake. Irritatingly, it made the expression evaluation loop just too big, so I had to use a couple of interim jumps – this is because the SC/MP can only jump backwards or forwards 128 bytes (without using XPPC) which means if you have a routine longer than that you break it up or you have jumps in the middle of code whose sole purpose is to transfer to the top and bottom of code. (If anyone ever reads the code, these are marked by being called \_\_EError2, \_\_EEError3, \_\_EEError4 and so on , all these will jump through a chain and end up at \_\_EEError).

The problem with the XPPC alternative is it breaks P3 (and takes 7 bytes).

Otherwise, things move on. I’ve written code for CALL, CLEAR, END, GOTO and NEW, so far, which leaves IF, IN, OS, PR, RUN, LIST and LET (which is optional). None of these are massive because I’ve done the ground work.

I have split code into sections, so the stuff relating to how the program is stored in memory (haven’t decided yet, may use the original …..) is all in one file ; hence GOTO uses a subroutine which doesn’t actually exist, and NEW is a call to a dummy as well. I will probably put IN and PR (the INPUT and PRINT equivalents) in their own file too. IN especially is slightly odd, and will require some reading of the 8080 code to figure out exactly how it works – when you type a string in it says it only includes spaces in quotes … does it really mean this ? I’d guess it’s using the same input routine as the command line. One difference between the original and this one is that it doesn’t mind spaces, whereas the original only allows it in a few areas – character constant, comment and after IF … I think anyway ☺