**Release: RAM/ROM Board design**

The design for the expansion RAM/ROM board is fairly simple. Like many designs of its type, it’s also very repetitive.

The basic design allows for up to 4kb of ROM, made up of up to 4 MCM6830 chips (there’s that Motorola thing again), and 32 128 x 8 MCM6810 chips (ditto). I probably should have designed them using 1k x 1 SRAMs as looking at 1970s prices they were a lot cheaper. Probably would be better with 2708 EPROM rather than 6830 ROMs. This would actually require more logic for decoding, one of the advantage of the 68x0 series is they have quite a lot of chip select pins built in – the design makes use of this feature.

This design won’t work I think, because the processor couldn’t cope with all this without some buffering (this applies to the whole thing, I haven’t considered fanout at any time. I probably didn’t know it existed). But aside from that, it seems to be okay.

U1’s job is to latch the upper 4 address lines (A12-A15) on NADS strobe. IU4Aa detects this as being 0000 and this is the “ZeroPage” signal which is used to enable all the addresses on the CPU board. As there could be theoretically more than one 4k RAM/4k ROM board this would have to be a wire, or I’d have to add a diode-OR gate to the CPU board. In reality at this time though, this board would be horribly expensive to fill up with RAM, even if I redesigned it using cheaper 2102 1k x 1 SRAMs.

U4B selects the page this memory appears on. This is done using A12,A13 and A14 or their inverses, this is a handy by product of using the 7475. A15 = 1 is for the ROM, A15 = 0 is for the RAM.

So if your page selection gate (U4B) is set to page 5 (101) then :

* The 4k of ROM will be at D000-DFFF (1 101 xxxx xxxx xxxx)
* The 4k of RAM will be at 5000-5FFF (0 101 xxxx xxxx xxxx)

The 4k of RAM is divided into 512 byte blocks of 4 x 128 x 8 each. One of these is shown. Their chip selects are : the current page signal, one of a combination of A7 and A8 and their inverses connected to chip selects (see the A7:x A8:x text above each RAM chip and track what the A7/A8 bus lines are connected to), and a page selection output from U3 decoded from A9,A10,A11. This itself is only enabled if A15 is low via U3 pin 5 (otherwise all the U3 outputs are driven high)

The ROM decoding is simpler. It uses for chip selects for each 1k ROM – the current page (e.g. A12,13,14), A15 must be logic 1, and a combination of A10 and A11 or their inverses. The same usage as A7 and A8 in the RAM area.

The schematic only shows 1k ROM and 512 bytes of RAM, but there are just 3 more ROMs and 7 more banks of 6810 RAM chips to fill it up, same connections, varying just on which output of U3 or which of A10/Not A10 A11/Not A11 is connected.

A slight oddity is because page zero already exists, the logical place to put the expansion board is in page 1 (so the RAM is 1000-1FFF which is continuous from the RAM on the CPU board, and the RAM is 9000-9FFF).

You can still get at the missing 4k of ROM (8000-8FFF) by putting the board to plug in at Page 0. This will work, but you can’t then put the RAM in as well, because it will collide with the 0000-0FFF ROM, RAM and I/O.

If you have multiple boards the design is a bit wasteful, the decoding and so on is repeated. I think to grasp this you have to look at chip prices then. 7400 et al weren’t far off the price they are now – they are cheaper but it’s the same sort of price (in 1978 a 7400 cost 13p).

RAM/ROM is incredibly expensive. A 62256 (32k x 8 SRAM, which is enough for the entire RAM memory of this machine, is about £10 or so. In 1978 a single 6810 RAM was £4.00), and money is now about 3.5 x the value then, so the single 128x8 RAM is more expensive than the 32k x 8 SRAM chip.

If we fully populate one of these boards and connect it to page 1 (e.g. the AND gate has A12, Not A13 and Not A14), we have the following memory map. (This is the standard layout because it makes the CPU Board RAM contiguous with the expansion RAM).

Well, except it isn’t. The SC/MP upper 4 bits work like a page register, in that if you have a pointer register pointing to say 0FFF and you add one to it, it wraps round to 0000 rather than going to 1000.

This makes some things horribly messy. A consequence of this is the HLL RAM will be split between that on the expansion board, which will be used for screen storage, variables etc. and that on the expansion RAM board.

|  |  |
| --- | --- |
| **Address** | **Contents** |
| 0000-07FF | 2k Monitor ROM (Write to Video circuitry) |
| 0800-0BFF | Keyboard (or toggle switch) input |
| 0C00-0FFF | CPU Board RAM |
| 1000-1FFF | Expansion RAM |
| 2000-8FFF | Unused |
| 9000-9FFF | Expansion ROM |
| A000-FFFF | Unused |