CORSHAM

Corsham Technologies, LLC
www.corshamtech.com
617 Stokes Road, Suite 4-299
Medford, NJ 08055

SS-50C 1 MB RAM Board

Introduction

One of the really cool things about the SWTPC 6809 CPU board is the ability to extend the 64K memory map to a full megabyte by use of Dynamic Address Translation (DAT). I'm not convinced many people actually had more than 64K back when SWTPC was making their CPU board, but when I designed the Corsham Tech 6809 one of the features added was 128K of RAM.

It seemed natural that people with either our CPU board or the original SWTPC board would want to use the extended memory scheme, so it didn't take long for us to design this 1 MB board.

There were no existing memory boards to model the design after so this was done based on looking at the CPU board schematics and the crude description of how DAT worked. A number of customers have confirmed this board works well in OS/9 systems, as well as our XMEMTEST extended memory test program.

Features

- Fully compatible with the DAT design.
- Supports A16-A19.
- If not all four extended address bits are desired, any of them can be ignored.
- Each of the sixteen 64K memory banks can be enabled/disabled via switches.
- RAM chips are static. No refresh logic.

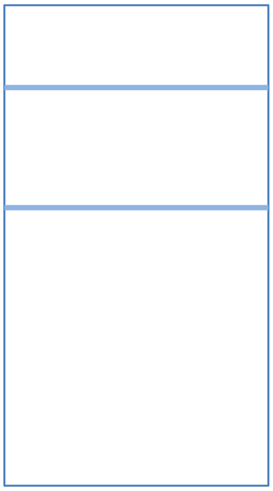
Address Bit Enables

SW3 on the lower left corner of the board select which of the four extended address bits to honor. These are not enable/disable per-se, they select which of the four extended address lines are used in determining if the on-board RAM is to be used.

We strongly suggest having all four of these switches ON/ENABLEd.

Bank Select Switches

SW1 and SW2 select which banks this board responds to. Notice that if a bank is enabled then the entire 64K in that bank is available. Of course, the configuration of the DAT registers on the CPU board determines which of the sixteen 4K blocks in each bank is mapped to the main memory map, and where in memory it is mapped.


Offset 1800
At address F800
Offset 1000
At address F000 if SW3 is set to 4K, not present in 2K
Offset 0000
Lower 4K is never mapped to memory

Dynamic Address Translation

This section was pulled from our 6809 CPU board manual. It's not really important to understand how DAT works unless you are writing your own programs using the extended memory, but DAT is not straightforward so it seemed that including the documentation here too might be helpful to some.

You don't really need to read this section unless you plan on writing software that uses the extended memory, in which case it's good to understand how SWTPC mapped 1 MB of address space into a processor with only 64K of address space. They did this with Dynamic Address Translation, or DAT. DAT uses 16 RAM locations to map a 16 bit address from the processor into a 20 bit address space.

The top four address lines (A12 to A15) are used as address select lines to 16 bytes of memory. The lower 4 bits of each address map to A12 through A15. The upper 4 bits are A16 to A19.

The top page of memory (FF00 to FFFF) is always mapped to the top 256 bytes of the EPROM. When SBUG starts, it loads up the DAT registers to map 56K of memory from 0000 to DFFF.

Addresses FFF0 to FFFF are the write-only DAT registers. If you read those locations you'll get the contents of EPROM, not the DAT registers. Each register maps one 4K block of memory:

Address	Block	Default value
FFF0	0xxx	0F
FFF1	1xxx	0E
FFF2	2xxx	0D
FFF3	3xxx	0C
FFF4	4xxx	0B
FFF5	5xxx	0A
FFF6	6xxx	09
FFF7	7xxx	08
FFF8	8xxx	07
FFF9	9xxx	06
FFFA	Axxx	05
FFFB	Bxxx	04
FFFC	Cxxx	03
FFFD	Dxxx	02
FFFE	Exxx	01
FFFF	F000	00

That's as clear as mud, right? Okay, the value written into the registers is the inverse of the value for the lower 4 bits, and the true value for the upper 4 bits. Still not clear, I know, so let's take an example:

FFF0	0xxx	0F

The value 00001111 (binary) is written into the register. When the upper four bits of the address (A12 to A15) are 0000, the entry above is used. The inverse of the lower four bits of DAT register at FFF0 is 0000 (since it has 00001111). So the values for A12 to A15 put onto the bus will be 0000.

So how do we use that? Well, let's assume you want to load and use two programs that are both start at address 0000 hex. You can select bank 0's memory by writing 0F to FFF0 and load the first program.

Now there are multiple ways to put another block of memory at address 0xxx. You can map another block from bank zero, such as moving the memory currently at

8000 down to 0000 by writing 07 hex to FFF0. The inverse of 7 (0111) is 8 (1000), so now when any address with 0000 as the top four bits is selected, the top four bits put onto the address bus will be 1000.

Another way is to use bank 1 so that all of bank 0's memory remains in place. To do this, put the value 0001 in the top 4 bits by writing 1F to FFF0. Now bank 1 will be selected for all 0xxx addresses.

Load up your second program to 0000 and you're set! To select the initial program again, write 0F to FFF0.

Revision History

Version	Changes
1	Initial release.
2	Fixed a few minor problems.

Parts List

Part	Number	Description
PCB	1	Printed Circuit Board (Corsham Tech)
J1	5	Molex 09-52-3101
C1	1	4.7uf, 25v electrolytic capacitor
C2, C3, C4, C5	4	.1 uf disc capacitor
R1, R2	2	680 ohm ¼ watt
R3, R4, R5, R6	4	10K
SW1, SW2	2	8 position DIP switch
SW3	1	4 position DIP switch
U1	1	7805 5 VDC regulator
U2	1	74159 decoder
U3	1	74LS04N inverter
U4, U5	2	HM628512 512K static RAM
	1	24 pin wide IC socket for U2
	1	14 pin IC socket for U3
	2	32 pin IC sockets for U4 and U5