Lab 19 Memory Interface and Memory Testing

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- Interface a static RAM chip to the 6812,
- Implement a memory test program.

Review
- Valvano' Chapter 9 on interfacing to the MC68HC812A4,
- Data sheets on the 2K by 8 bit static RAM Am9128 (same as MC6116),

Starter files
- MEMTEST.C

Background

This experiment considers the design of a 2048 byte RAM system. It will serve to demonstrate the general problems involved with static RAM memories. The memory system to be designed in this experiment will involve an Am9128 RAM chip (2048 addresses by 8 bits). This can store a total of 16384 bits and each byte has a unique address. This is called byte addressable. The address decoder will determine which cycles to activate. With minimal cost the decoder will

- Enable the RAM during accesses to the specified address range $7000 to $77FF,
- Disable the Ram during access to other specified modules in the system.

This will conserve power because the power consumed during the disable state is much less than that consumed during the enable state. See also I_{cc} and I_{SB}.

It is important to note that this experiment requires every word of your external RAM to be accessed. The 2K RAM will be placed at any address range using one of the 6812 built-in address decoders. The example software uses CSD and places it from $7000 to $77FF (it actually uses a $7000 to $7FFF decoder.)

Preparation (do this before your lab period)

1: You will draw the circuit that interfaces a 2048 by 8-bit memory, the Am9128, to the 6812. You should use the CSD, which is a built-in address decoder of the MC68HC812A4. Draw a logic diagram for the memory system including pin numbers for all IC's. Show all connections to the MC68HC812A4 bus. On your circuit diagram, label both the 6812 pin name and the Adapt812 H2 connector pin number. The memory bus is available on the Adapt812 H2 connector:

- 6812 Ports A, B contain the 16-bit address A15-A0,
- 6812 Port C contains the bidirectional 8-bit data bus,
- 6812 Port E contains the E clock and R/W,
- 6812 Port F contains the built-in chips selects.

In narrow mode the data is on PORTC, but is labeled as D15-D8 on the Adapt812.

2: Choose the correct number of cycle stretches so that Read_Data_Available overlaps Read_Data_Required and that Write_Data_Available overlaps Write_Data_Required. By using the built-in chip select, CSD, you will be synchronizing the memory control signals with the E clock. Draw the combined READ timing diagram and draw the combined WRITE timing diagram, as described in Chapter 9 of the book.

3: Write a simple program to test the memory at $7000 to $77FF you have just designed. First fill the memory with zeros, then check each location for zero. Next, fill the memory with ones, then check each location for $FF. Report any errors to the screen. You will use this program during the hardware test phase of the lab.

4: Next, write a program to test dynamic RAMs. You will run this program to test the internal static RAM $0800-$0BFF of the 6812. Even though it is a static RAM, it should illustrate the differences between static and dynamic memory testing. You will be testing whether the DRAM can retain information over a few memory refresh cycles. In particular, you will write a known sequence of data values into the memory, wait 2 seconds, and verify if that sequence is still present.

5: Next, write a complex test program to find stuck at errors. You will use this program to test memories with simulated faults. An address can be stuck high, stuck low, or stuck together, as simulated in Figure 19.1.

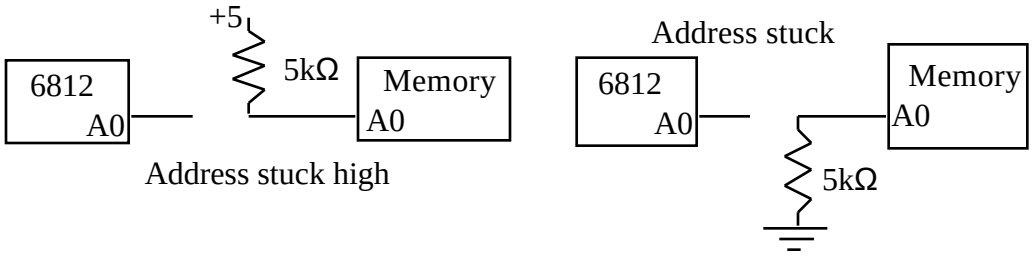

Figure 19.1. Simulations of memory address-stuck-at faults.

Your system should give some information that could be used to determine the type of error (e.g., data bit stuck at one, data bit stuck at zero, address line stuck at one, address line stuck at zero.) *Some of these error types may be impossible to resolve using software tests.* Therefore, do the best you can. A syntax free hardcopy printout of the software is required as part of the preparation. Please provide detailed documentation of the testing algorithm you use as part of the software report. Invent hardware connections similar to Figure 15.1 that create typical memory chip failures. Of particular difficulty is simulating data bit stuck high, data bit stuck low, and data bits stuck together. The 6812 will stop running if you pull data bits high or low (not a good idea).

Procedure (do this during your lab period)

1: Connect the memory system to the microprocessor using the address and data busses and the appropriate memory control signals. In expanded narrow mode, Port C is the data bus (labeled D15-D8) on the H2 connector.

2: First test your interface with the following simple program (you could also configure the `MODE=0x3B;` to see internal memory accesses on the external bus):

```
/* filename  MemTest.C  Jonathan W. Valvano,
   PortT is output,
   External RAM from 0x7000 to 0x77FF*/
#include "HC12.h"
#define memory *(unsigned char *)(0x7000)
char *pt;
void main(void){ unsigned int i;
  COPCTL=0;    // disable COP
  DDRT=0xFF;   // PortT is output
  MODE=0x33;   // special exp narrow
  MODE=0x33;   // special exp narrow
/* bit 765 001  Special expanded narrow
   bit 4 ESTR=1  E stretch during external
   bit 3 IVIS=0  No Internal bus activity visable
   bit 1 EMD =1  Port D access as external device
   bit 0 EME =1  Port E access as external device */
  PEAR=0x2C;
/* bit 7 ARSIE=0  PE7 is general I/O
   bit 6 PLLTE=0  no PPL test
   bit 5 PIPOE=1  enable pipe
   bit 4 NECLK=0  show E
   bit 3 LSTRE=1  low strobe enable
   bit 2 RDWE =1  R/W enable */
  CSCTL0=0x10;  // enable CSD
  CSCTL1=0;     // $7000 to $7FFF
  CSSTR0=0x3F;  // stretch 3 E clocks
  for(i=0x7000;i<=0x77FF;i++){
        pt=(char*)i;
        (*pt)=0; // clear memory
  }
  while(1){
    memory=0xC3;  // write data to external RAM
    PORTT=memory; // read from external RAM
```

```
}};
// no vectors because program started from BDM
```

To download and run a program in special expanded narrow mode, follow the steps

1) Within ICC12 edit, compile the program using the regular single chip segmentation
   - RAM at 0x0800-0xBFF (globals start at 0x0800, SP initialized to 0x0C00)
   - EEPROM at 0xF000-0xFFFF

2) With power off to the Adpat812 and BDM boards
   set MODA=0, MODB=0 jumpers (usual settings)

3) Power up Adapt812 and BDM

4) Start BDM software

5) Download S19 record in the usual way

6) Execute **firm**, **reset**, **b** until 6812 is halted (**r** command works) and the **status** returns **C0**

7) Verify the processor is in special mode
db 000b should give **1B** for *special single chip mode*
(program will change it to **33** *special expanded narrow mode*)

8) Verify the EEPROM is at $F000-$FFFF
db 0012 should be F1

9) Start your program from the debugger (do not hit the reset button[1])
g f000

In this manner, the memory system decoder can be checked to verify that it is functional. In these tight loops, a pulse should be received from the $\overline{CE}$ line, during accesses to the memory. Observe $\overline{CE}$ with the following signals in a pairwise fashion:

- $\overline{CE}$ and E
- $\overline{CE}$ and A14
- $\overline{CE}$ and R/W
- $\overline{CE}$ and your other RAM signals, $\overline{OE}$, $\overline{WE}$, and other interesting and appropriate signals

on the oscilloscope using the dual channel capability of the scope. Trigger on the falling edge of $\overline{CE}$. Using the dual channel capability, verify the proper relation between E, A14, R/W and your address decoder. Record the timing diagram in your notebook. In this way you should be able to verify the timing diagram created as part of the lab preparation.

3: For the initial check, store data in the desired locations and read the data back using the BDM monitor. This should simplify the debugging procedure. In any case, different data should be loaded into locations $7000, $7001, $7002, etc. Only after all the data is loaded should the initial location be examined to determine if the correct data was loaded. This should identify gross errors in the memory interface.

4: Load the simple memory test program to verify that the complete RAM system is operating correctly.

[1] Hitting the reset button will place the processor in normal single chip mode. You need special mode.

5: Load the complex memory test program. As part of the experimental evaluation, disconnect various data and/or address bits and report how well your software distinguished these errors.

6: Load and execute a program that will change A0 and D0 during successive memory cycles. Observe the program execution with a Logic Analyzer or multiple channel scope. Draw the read and write timing diagrams as actually measured in your notebook. Include the following signals:

6812: E, R/W, A0, D0

9128: $\overline{CE}$, $\overline{WE}$, $\overline{OE}$

Include data available and data required. Which one did you measure and which one did you derive? Verify that all of the 6812 and RAM timing restrictions are met. Repeat for several write cycles. Explain the differences in the READ and WRITE cycles as observed on the Logic Analyzer and/or scope. Record in your report the actual READ and WRITE cycle timing as seen using the logic analyzer. Compare and contrast the two methods (oscilloscope and logic analyzer) of measuring the actual READ and WRITE cycle timing diagrams.

Checkout (show this to the TA)

You should be able to demonstrate using BDM that your memory system can be read from and written into. Next, you should run your test program with various data and/or address bits disconnected.

Hints

1) If your Am9128 does not work, you can modify the Lab assignment to use any available static RAM. Let your TA know if you plan to change chips.
2) Please don't put broken RAM chips back in your bag. Mark bad chips as destroyed.
3) If the logic analyzer's don't work, try to do something reasonable with the scope, otherwise you may skip the logic analyzer components.
4) The hardware configuration is very similar to the 68HC812A4/MCM60L64 RAM example in the book, so if you are having trouble refer to this example.