Lab 10 – Serial RAM
November 5, 2015

In this lab you will use the Serial Peripheral Interface (SPI) system on the HCS12 microcontroller to store data to and read data from a serial static RAM (the Microchip 23LC512). You will use the serial communications system (SCI) to enter the data to be stored on the SRAM, and display the data that is read back.

1 Prelab Questions (complete prior to coming to lab)

Read this handout and do the following:

1. Create a schematic for the connections between the HCS12 and the 23LC512 SRAM chip, indicating the names and numbers of the pins.
2. Following the flowchart that was given in the lecture notes, write a C module that writes a byte to the SRAM at a given address.
3. Following the flowchart that was given in the lecture notes, write a C module that reads a byte from the SRAM from a given address.

2 Connections to SRAM

We will use the SPI system on the HCS12 to interface to the 23LC512 SRAM chip. Make the connections between the SPI pins on the HCS12 and the SRAM (don’t forget power and ground on the SRAM). As discussed in lecture, we won’t use the SS (slave select) pin from the HCS12 to drive the CS (chip select) input on the SRAM, but instead will use pin PT0 from the HCS12. Show your connections to the instructor and then connect the power supply.

![SPI Mode](image)

Figure 1 Pinout for the Microchip 23LC512 SRAM. We won’t use pin 3. Tie pin 7 (HOLD) to +5V.

Power the SRAM with +5V \( (V_{\text{sharp}}) \). As described in lecture, one of the SPI signals from the MCU is hardwired to the speaker on the SSMI board. When you run your program you will hear clicking and popping from the speaker. This is caused by the MOSI pin connected to the transistor that drives the speaker. Regardless, the SPI continues to function properly.
3 Writing and reading a byte

Write a C program that writes one data byte to the SRAM; for example, to address 0. Your program should then read the byte back from that address and store it into a global variable in your program. Run the program and inspect the value of that global variable, and verify that the byte that was read back is the same as the byte you wrote out to the SRAM. Use the oscilloscope to capture the SCK, MOSI, MISO and CS signals. Discuss what you see on these signals and point out where the instructions, address, and data are sent.

Sign-Off 1: Demonstrate your program and data waveforms to the instructor. Discuss the send and received data to/from the SRAM.

4 Writing and reading multiple bytes

Now modify your program to get characters from the serial communications interface (SCI). You will have to use the methods that you used in the previous lab; namely, putting the MCU into “run” mode after loading the program, and then using the PuTTY terminal program on the PC to send and receive characters.

When your program receives a character from the SCI port, it should echo it back to the SCI port (so you can see what you are typing) and then write it to the next address in the SRAM. It should continue doing this until you hit the ENTER key on the keyboard (recall what ASCII character is generated by the ENTER key).

Then your program should read the stored characters from the SRAM and send them to the SCI port. As a result, you should see on the PuTTY terminal screen, the exact sequence of characters echoed back, that you originally typed in.

One final note – it will be helpful to output the ASCII line feed (LF) character after the ENTER key is read in and echoed back. This will make the PuTTY terminal screen easier to read.

The pseudo code for the program is show below in Figure 2.

Sign-Off 2: Demonstrate your program to the instructor.

Include in your report:

1. The C language program, along with a corresponding flowchart or pseudo code.
2. An explanation of the circuit and program.
3. Oscilloscope traces from “Writing and Reading a Byte”.

The pseudo code for the program is show below in Figure 2.
Initialize phase locked loop
Initialize SCI system
Initialize SPI system
Enable interrupts

Loop forever
  addr = 0
  Do
    Get a character from the SCI port
    Echo the character back to the SCI port
    Write the character to the SRAM at address addr
    Increment addr
  Until the character is the ASCII code for the ENTER key
  Send a line feed character to the SCI port

  addr = 0
  Do
    Read the character from the SRAM from address addr
    Increment addr
    Send the character to the SCI port
  Until the character is the ASCII code for the ENTER key
  Send a line feed character to the SCI port
End loop

Figure 2  Pseudo code.
Lab 10: Serial EEPROM

Name: ________________________________ Name: ________________________________

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Initials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign-Off 1</td>
<td>Writing and Reading a Byte</td>
<td></td>
</tr>
<tr>
<td>Sign-Off 2</td>
<td>Writing and Reading Multiple Bytes</td>
<td></td>
</tr>
</tbody>
</table>

5 Rubric

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>20 pts</th>
<th>Pre-Lab</th>
<th>5 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>C code with flowchart/pseudocode</td>
<td>/ 10</td>
<td>Questions</td>
<td>/ 5</td>
</tr>
<tr>
<td>Circuit and program Explanation</td>
<td>/ 7</td>
<td><strong>Demonstrations</strong></td>
<td>20 pts</td>
</tr>
<tr>
<td>Scope Traces</td>
<td>/ 3</td>
<td>Sign-Off 1: Writing/Reading Byte</td>
<td>/10</td>
</tr>
<tr>
<td>Composition</td>
<td>5 pts</td>
<td>Sign-Off 2: Writing/Reading Multiple Bytes</td>
<td>/10</td>
</tr>
</tbody>
</table>

Total / 50 pts