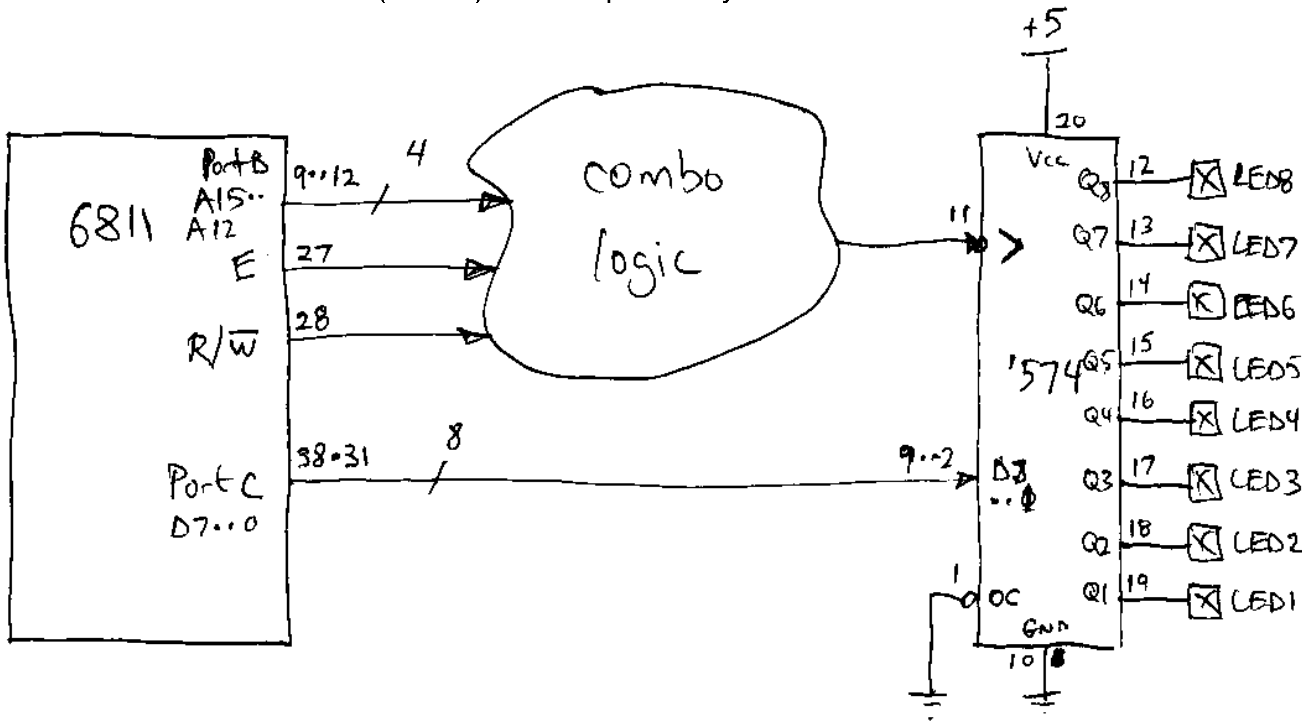
Assignment 6: Memory-Mapped Address Decoding and the Stack

Overview

This assignment has one lab-based design problem, one paper-based design problem, and one paper-based exercise. Please refer to the handout "Appendix B" as background and reference.

Problem 6-1: Memory-Mapped Output Latch.

An 8-bit output latch is to be mapped into the HC11's address space. Any HC11 memory writes to the address range 0x5000 to 0x5FFF (inclusive) should cause the data byte written from the HC11 onto the data bus (Port C) to be captured by the latch.



Detail

The circuit above illustrates the method. A block of combinational logic (to be designed by you) accepts as input the 4 higher -order address bits (from the HC11 Port B), the E clock, and the R/~W signal. The combinational logic should decode these signals and produce a negative-true enable signal when:

- the high four bits of the address are 0x5, and
- the E clock is high (it's a positive-true signal), and
- the R/~W is write (i.e., as a negative-true signal, it is 0).

Your combo logic circuit should generate a true signal (0 output) when all of these conditions are satisfied, and a false signal (1 output) otherwise.

Wire your circuit's output to the clock input of HC574 latch chip as illustrated. Connect the HC11 PORTC data bus to the inputs of the HC574. Wire the latch's outputs to LEDs. **Make sure to wire the latch's pin 1 output control enable to ground!**

Test Code

The test program `expanddemo.s` is provided to help you test your design. When bootloaded into the HC11, the code puts the HC11 into expanded mode (enabling the address bus, data bus, and R/~W signals), and then writes sequential numbers 0, 1, 2, 3, ... 255, 0, 1, etc. to memory address 0x5000. In between each memory write, the piezo line is toggled, producing a click. There is about a quarter-second pause between each memory write/piezo click.

When you attempt to bring up your circuit, **make sure you have Port A 4 (pin 4 of the HC11) connected to the piezo!** You should then hear a continuous *click...click...click...* while the `expanddemo.s` program is running. Make sure you hear the clicking! If you don't, the program isn't running and your design has no chance of working!

Let's take a look at the `expanddemo.s` program:

```
;;; expanddemo.s
;;; puts HC11 in expanded mode,
;;; writes sequentially increasing values to address 0x5000,
;;; and delays and clicks on PA4 between each write
;;; Fred G. Martin / UML CS 2002

PORTA=  0x1000
HPRIO=  0x103c                  ; highest priority bit int and misc reg

        ldaa HPRIO              ; read the reg
        oraa #0x20              ; set MDA bit -> expanded mode
        staa HPRIO              ; write it back

        clrb                    ; init b at 0
loop:
        stab 0x5000             ; write b to target
        incb                    ; inc it for next time

        ldaa PORTA
        eora #0x10              ; toggle piezo bit (click)
        staa PORTA

        ldy #0                  ; delay for 65k loops
delay:  dey
        bne delay

        bra loop                ; again
```

After the definitions of the PORTA and HPRIO registers, the first three lines of code set the MDA bit (bit 5) in the HPRIO register. This puts the HC11 into expanded mode. From this point forward, and memory accesses outside of either the internal RAM (0x0000 to 0x01FF) or the register bank (0x1000 to 0x103F) are accomplished using PORTB, PORTC, the R/~W line, and the address strobe (AS) signals.

Next, accumulator B is used as a counter, and is cleared to zero to begin. The core of the program follows. Accumulator B is written to address 0x5000. This causes an external memory access using the HC11 lines just mentioned. Accumulator B is incremented for next time.

Next the piezo line is toggled, causing a click. This is done by loading the A accumulator from PORTA, exclusive-OR'ing the piezo pin, and storing back.

Next, a long delay loop (about a 1/2 second) is executed using register Y as a 16-bit counter down from 0 and back again.

This process repeats ad infinitum.

Please Note!

It is possible to design a circuit that "works" but is NOT correct. For example, if you build a circuit that says "true" when the high nybble of the address is both 0x5 and 0x6, the circuit will "work" (that is, you will see the values written to address 0x5000). But this circuit will also respond to address 0x6000, and it should not!

So make sure you are decoding *only* the address range specified in the assignment.

To Turn In:

6-1a. Draw a schematic diagram of your circuit that decodes the address from the HC11. Make sure your schematic includes:

- The HC11 chip itself, which signals from it you are using, and the pin numbers of those signals. Inside the box for the HC11, you should have the names of the signals (e.g., A15); outside the box, you should have the pin number (e.g., 8).
- Logic level diagrams for other chips that you are using. E.g., if you use an inverter, draw the diagram for an inverter, not the 14-pin box that the inverter lives in. *See `http://www.cs.uml.edu/~fredm/courses/91.305/files/schematic-hints.pdf` for more details on how to properly draw a schematic.*
- The 74HC574 latch. Show all signals on the 'HC574 chip, including the fact that its outputs are wired to LEDs so you can see their state. (In effect, reproduce the drawing of the 'HC574 from this handout.)

6-1b. Provide a brief written explanation of your circuit, explaining how your circuit accomplishes the decoding task.

6-1c. After your circuit is working, download the file `mystery5000.rel` to your HC11. Describe the pattern of lights that you see when running this file.

6-1d. Answer the following. The demo program and the mystery test program both write the address 0x5000 to talk to your latch. But this is not the only address that your latch will respond to, because the design only does partial decoding of the full 16-bit memory address. *What is the full range of addresses (in hex) over which the latch will respond?*

Problem 6-2: 8K RAM for 68HC11

Assume you have an 8K static RAM, the CY6264. (The first page of the CY6264 data sheet is appended to this handout, and the full data sheet PDF is available from the course web site.)

This RAM is to be mapped into the 68HC11's address space in the range 0x8000 to 0x9FFF.

Design a circuit to accomplish this, and draw a schematic to represent it.

Your schematic should include the 68HC11, a 74HC373 transparent latch, the CY6264 static RAM chip, and any other gates or components you deem necessary.

Please note:

1. You do not need to build the circuit, just draw the schematic.
2. Include a brief (one-paragraph) written description of your design.
3. Make sure to use only parts that are available in your kit.

Problem 6-3: Understanding Operation of the 68HC11 Stack.

Consider the following 68HC11 program, which loaded into memory beginning at location 0.

Your task is to specify (1) the contents of the stack and (2) the values in the SP (stack pointer) and X registers at three different points of the program execution.

```
  mem     object               line #        instruction                  comment
location  code bytes
                                 1 ;;; stackdemo.s
                                 2
  0000 8E 01 FF                  3 start:  lds #0x1ff                ; set stack ptr
                                 4
  0003 CE 10 00                  5         ldx #0x1000               ; initialize X
  0006 3C                        6         pshx                      ; put X on stack
                                 7
  0007 BD 00 0D                  8         jsr delay                 ; call subr
                                 9
  000A 38                       10         pulx                      ; restore X
                                11
  000B 20 FE                    12 done:   bra done                  ; all done
                                13
  000D CE 01 F4                 14 delay:  ldx #500
  0010 09                       15 delaylp: dex
  0011 26 FD                    16         bne delaylp
  0013 39                       17         rts
```

Use the diagrams below to fill in your answers. The first one is done for you. Please note:

- All values are assumed hexadecimal; please omit the “0x” prefix.
- Each memory address for the stack holds one byte, but the SP and X registers are each two bytes wide.
- “after line 5” means: after the instruction at line 5 has executed, but before the subsequent instruction has been run.
- If a value is indeterminate, mark it with a dash.

after line 5:

STACK

m em addr	contents
0x01FB	—
0x01FC	—
0x01FD	—
0x01FE	—
0x01FF	—
SP reg	01FF
X reg	1000

after line 6:

STACK

m em addr	contents
0x01FB	
0x01FC	
0x01FD	
0x01FE	
0x01FF	
SP reg	
X reg	

after line 8:

STACK

m em addr	contents
0x01FB	
0x01FC	
0x01FD	
0x01FE	
0x01FF	
SP reg	
X reg	

after line 10:

STACK

m em addr	contents
0x01FB	
0x01FC	
0x01FD	
0x01FE	
0x01FF	
SP reg	
X reg	