MPS Memory Interfacing Lab Exercise

Interfacing Memory Chips on the 6812 Processor Bus

Student's name & ID: ____

Partner's name & ID: ____

Your Section number / TA's name __________________________________

Notes:

You must work on this assignment with your partner.

Hand in a printer copy of your software listings for the team.
Hand in a neat copy of your circuit schematics for the team.

These will be returned to you so that they may be used for reference.

-------------------------do not write below this line------------------------

	POINTS	TA init.
Grade for performance verification (50% max.)		
Grade for answers to TA's questions (20% max.)		
Grade for documentation and appearance (30% max.)		

Grader's signature: ____

Date: ____

Interfacing Memory Chips on the 6812 Processor Bus

GOAL

By doing this lab assignment, you will learn to interface to the M68HC12 expanded mode bus:

1. The Am9128 2048 x 8-bits Static RAM and access it with software written in C.
2. If time permits, the MCM6147 4096 x 1-bit Static RAM.

PREPARATION

- Read Sections 9.1 - 9.3, 9.6 - 9.9 from *Software and Hardware Engineering* by Cady & Sibigtroth. (Read Sections 6.2 and 6.4-6 from *Microcomputer Engineering* by Gene H. Miller.)
- Read 4.6.2, 4.6.3, and 4.14 from the 68HC12 EVB User's Manual
- References: 68HC12 EVB User's Manual, pp. 3-34 & 4-17 to 4-22
  Motorola MC68HC812A4 Technical Summary, pp. 17 & 43 to 47

Hardware design

- Do a top-down design of the bus interface hardware. Select the appropriate IC chips.
- Provide logic expressions for the various control pins, address decoding, etc.
- Use LogicWorks, or another logic simulator, to validate your logic expressions.

Software design

- Do a top-down design of your program. Provide a flowchart or equivalent.
- Write C programs for each module/subroutine. Provide planning documentation for each: flowchart and tests. Specify test inputs and expected output or other indictors of correct operation.
- Integrate your modules to get a C program that is free from syntax errors (i.e., it should compile without error messages and produce a .s19 file).

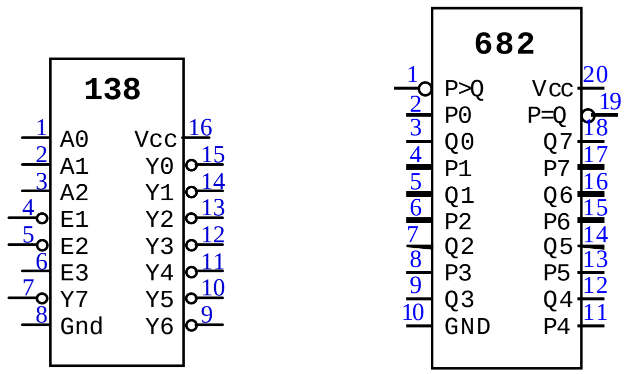
1. INTRODUCTION TO BUS INTERFACING

When the M68HC12 is operated in expanded mode, Port B acts as the Lo-Address Bus (A0-A7), Port A acts as the Hi-Address Bus (A8-A15), and Ports D & C act as the Lo-Data Bus (D0-D7) and Hi-Data Bus (D8-D15).

In addition to the Address and Data buses, the M68HC12 provides several signals that are used to control the bus interface: Enable (or ECLK), $R/\overline{W}$ (Read / Not Write - high $\rightarrow$ read and low $\rightarrow$ write operation), LSTRB (Lo Byte Strobe) and several programmable chip select signals ($\overline{CSP1}$, $\overline{CSP0}$, $\overline{CSD}$, $\overline{CS3}$, $\overline{CS2}$, $\overline{CS1}$, and $\overline{CS0}$). The control pins provide flexibility in configuring memory interfaces so that either memories with 8 bit wide or 16 bit wide data buses may be used.

The LSTRB, R/$\overline{W}$, and chip select signals together with address decoding logic are used to select the time when data is to be put onto the DATA-bus for a processor memory read operation, e.g. a LDAA M instruction, or to get data from the DATA-bus for a processor write operation, e.g. a STAA M instruction. For a memory read instruction, put the data onto the AD bus at the time labeled "put data." That is when both Enable and R/W* are high *and* the correct memory address is detected. For a memory write instruction, get data from the AD-bus at the time labeled "get data." That is when both Enable and R/W* are low *and* the correct address is detected.

The EVB board uses only a single 74LS32 quad OR gate and the programmable chip select pins to detect which 8K-byte address block is addressed. The chip select pins CS0-CS3 are also available for other purposes such as incorporating the "put data" or "get data" conditions with partial address decoding. A 74LS682 8-bit Magnitude Comparator can also be used to detect specific address bit patterns. The pin-outs for these two chips are shown next.



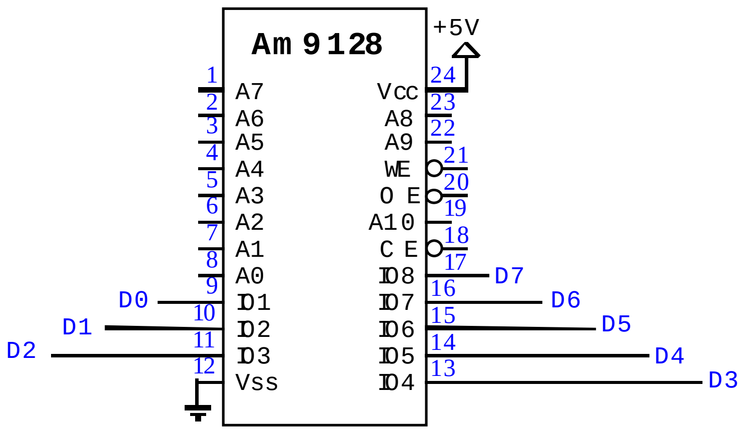
Pin-outs for the '138 and '682.

2. INTERFACING STATIC RAM

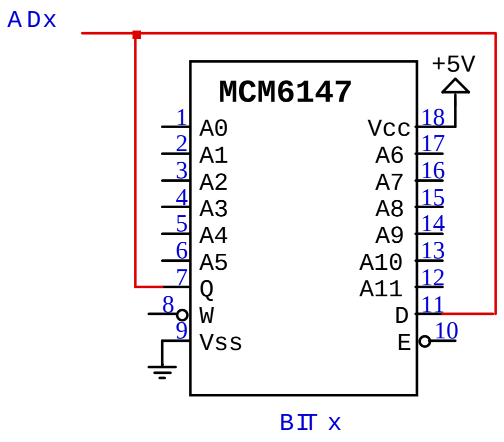
In this lab exercise, two kinds of static RAM chips are used. The first kind is the Am9128, a 2K x 1-byte memory chip. It has 11 address pins (A0-A10), 8 bi-directional input-output pins, (IO1-IO8) and 3 control pins. The control pins are Chip Enable (CE*), Output Enable (OE*), and Write Enable (WE*) and are all active low. Notice the 8 input-output pins are connected to the 8 DATA-bus pins. The control pins are driven by the "put data" and "get data" conditions and the address decoding of the address bits A15-A0.

The second kind of memory chip is the MCM6147, a 4K x 1-bit device. It has 12 address pins (A0-A11), 1 input pin (D), 1 output pin (Q), and 2 control pins. The control pins are Chip Enable ($\overline{E}$) and Write

($\overline{W}$) and are both active low. Notice both input D and output Q pins are connected together to the same DATA-bus pin. The control pins are driven by the "put data" and "get data" conditions and the address decoding for the 4 high order address bits A15-A12.



Pinout for the Am9128.



Pinout for the MCM6147

This lab uses the M68HC12 EVB as the processor to which you will be adding external memory. The EVB is configured for Normal Expanded Wide Mode (BKGD = MODB = MODA = 1). This means that in this mode 64 KBytes of memory can be accessed over a 16 bit wide data bus. All memory must be configured for wide data bus access using Port C and Port D as the 16 data bus bits and Port A and Port B as the 16 address bus bits. The LSTRB, CSD, and A0 are used to select the chip and R/$\overline{W}$ to select the operation. All the bus interface signals are available on the two EVB 60-pin connectors (J8 and J9). The pin assignments are given in Figures 4-5 and 4-6 and details given in Tables 4-5 and 4-6 in the 68HC12 EVB User's Manual (included at the end of this lab).

The M68HC12 is an extremely flexible chip that can be configured in hardware for many different modes depending on the I/O and memory requirements of the target application. The single chip mode needs no external memory and all the address and data lines are available for use as on-chip ports. The expanded mode can be programmed for a fixed 64 KB address space (16 bit address bus) referred to as the Expanded mode, or up to a 4 MB address space (22 bit address bus) referred to as the Memory Expansion Windows mode. The expanded/expansion modes can be configured for an 8 or 16 bit wide memory data bus referred to as narrow or wide mode. In the wide mode (used on the EVB), the LSB address bit A0 is not used in address decoding since only even addresses are used and with the high byte of the 16 bit data bus pointing to the following odd byte address. Two 4 KB chips must be wired in parallel with one chip being the low byte and one chip being the high byte of each even address. Refer to Figure 9-15 in the text. Since data memory (RAM) is being configured in this experiment, chip selection will be done using $\overline{\text{CSD}}$ rather than $\overline{\text{CSP0}}$. The chip select pins on the 68HC12 have been designed to eliminate or reduce the amount of glue logic needed to interface peripherals to the processor. Complete address decoding is possible for devices using less than 16 address lines when mapped to specific locations. $\overline{\text{CSD}}$ has been programmed on the EVB to decode 8 KB devices. [Refer to Chip Select Registers CSCTL0 ($003C) and CSCTL1 ($003D).] Since the 2 RAM chips used here provide only 4 KBytes, some extra address decoding will be necessary.

Two other control signals are used for timing and selection of the RAM chips. Figure 4-3 of the 68HC12 EVB User's Manual shows how to use $\overline{\text{LSTRB}}$ (LSTRB* in diagram), $\overline{\text{CSD}}$ (CS* in diagram), and A0 to generate the correct chip select and output enable signals for both the narrow and wide modes, even though this lab uses only the wide mode. Remember that you will need to modify this circuit to perform full address decoding on the 4 KB memory block.

3. LAB EXERCISE

This lab exercise has two parts. The first part is to interface two Am9128 static RAM chips to the M68HC12 (expanded) bus. The optional second part is to interface 4 MCM6147 static RAM chips to form a 4K x 1-nibble memory array.

1. Start the chip address space at location $2000. This uses some of the available 8 KB of address space on the EVB between the 4 KB of on-chip EEPROM from $1000 to $1FFF and the 16 KB of program RAM space from $4000 to $7FFF. Verify that the memory works as desired by using D-Bug12 commands to access locations within the block. Execute the D-Bug12 commands with the external RAM powered up and without the +5 V power attached. If the memory chip is correctly interfaced, the commands should be able to change values at memory locations in the range when the RAM is on and fail if the RAM is powered down. Make sure you turn in all complete logic circuit diagrams.

   Design the hardware top-down. Then validate the operation of the chips bottom-up. You can do this without the computer being connected. Thoroughly test each chip before integrating them. This way you can correct any hardware errors as you are assembling the modules.

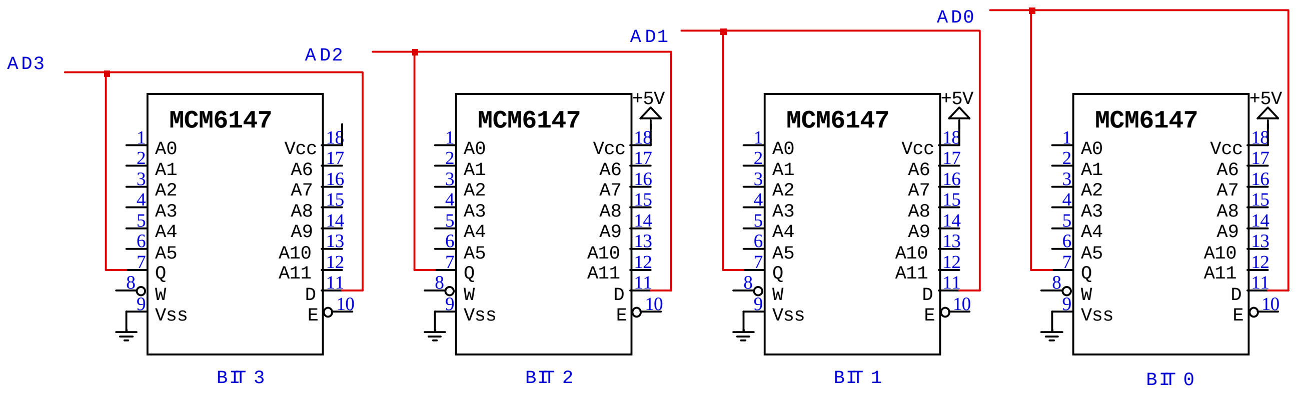
   Integrating and debugging working modules is much simpler than trying to debug an ensemble of untried chips. Your grade will depend on how well you follow these guidelines.

2. Write a short C program to check the correct operation of the memory chips. Do this by writing and then reading the pattern $AA at all 4096 locations starting at $2000. To be certain that address decoding is being performed correctly, you may want to be certain that other address values outside the block are NOT being affected by writes to values in the block. Next write and read the pattern $55 at all 4096 locations. Record any memory locations that fail to read the pattern that is written. Use a buffer of size 256 for the errors. If the buffer becomes full, empty the buffer to the screen and continue. Otherwise, display the buffer contents when the program finishes. Note that using pointers in C is a convenient way to access specific memory locations.

3. Repeat this test for another Am9128 memory chip. If you get the same error pattern, check your wiring for errors. It is highly unlikely that two chips have failed in exactly the same manner.

OPTIONAL

1. Start the address of the nibble memory array at the end of the Am9128 address space. Verify that the nibble memory works correctly by copying a list of 30 hexadecimal digits into the memory array beginning at the first address and using only even numbered addresses. Display side-by-side the original hexadecimal digits and the ones copied to the nibble memory array. If the array is interfaced correctly, the numbers should match. A partial schematic for the nibble memory array is shown here. Note, except for the data pins, the address and control pins are all connected the same. That is all A0s are connected together, all $\overline{W}$s, etc.

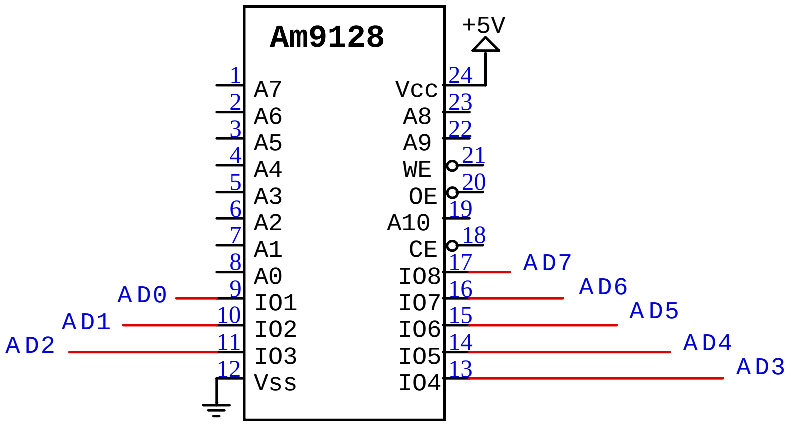


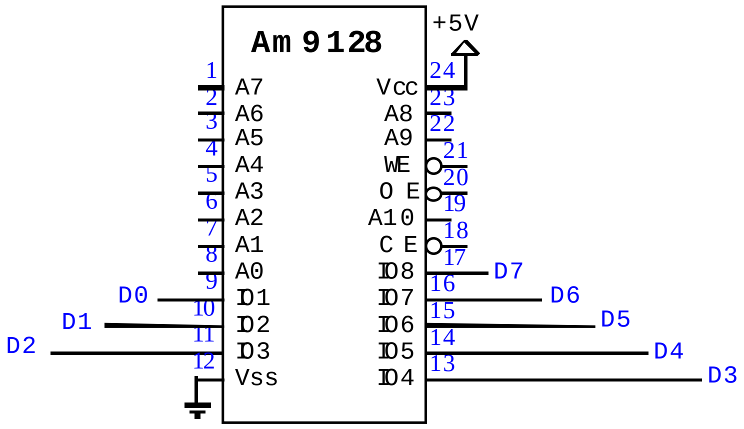
Partial schematic for 4-bit memory array.

2. Again, design the hardware top-down. Then validate the operation of the chips bottom-up. A single bit can be tested with D-Bug12 before the full nibble is wired.

ADDITIONAL NOTES:

1. Use the signals on the EVB J8 & J9 sockets.
2. Use CSD and address decoding on A12 - A15 to select the \$2000-\$2FFF address.
3. Use the R/$\overline{W}$ signals to control $\overline{WE}$ and $\overline{W}$.


Am9128
+5V
A7
A6
A5
A4
A3
A2
A1
A0
IO1
IO2
IO3
Vss
Vcc
A8
A9
WE
OE
A10
CE
IO8
IO7
IO6
IO5
IO4
AD0
AD1
AD2
AD7
AD6
AD5
AD4
AD3


Am 9128
+5V
A7
A6
A5
A4
A3
A2
A1
A0
IO1
IO2
IO3
Vss
Vcc
A8
A9
WE
OE
A10
CE
IO8
IO7
IO6
IO5
IO4
D0
D1
D2
D7
D6
D5
D4
D3