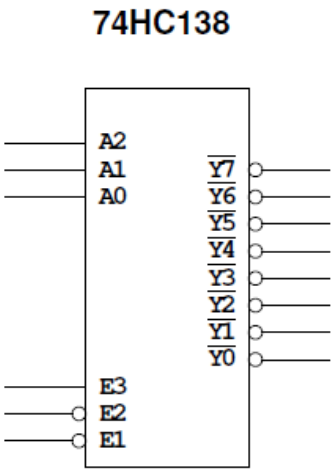
- **The MC9S12 in Expanded Mode – Using MSI logic to build ports**
- Huang Chapter 14

Using Medium-Scale Integration (MSI) Logic To Build An Output Port

• Many designs use standard MSI logic for microprocessor expansion

• This provides an inexpensive way to expand microprocessors

• One MSI device often used in such expansions is a decoder, such as the 74HC138 decoder chip



When E3 high, E2 and E1 are low, one of the outputs of the 74138 will go low A2, A1, A0 determine which output will be low.

A2	A1	A0	Output
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5
1	1	1	Y7

Using MSI Logic To Build An Output Port

• A 74HC138 decoder can be used to select a range of addresses

• The outputs of the 74HC138 are used to select up to 8 different devices

• The enable inputs are used to determine when one of the 8 different devices should be selected

• Because the data part of the memory cycle occurs when E is high, the active high enable input is usually connected to the MC9S12's E clock

• The active low enable inputs are usually connected to highest-order address lines (A15 and A14), or to combinational logic which is driven by highest-order address lines

• The address inputs of the 74HC138 are usually connected to the next highest-order address bits. Sometimes the low-order address input is connected to the R/W line of the MC9S12 to allow separate selection of input and output devices

• For example, on the figure on the next page, A15 and $\overline{A14}$ are connected to the active low enable inputs. This means that one of the outputs will be selected only when A15 and $\overline{A14}$ are low, or for the address range 0x4000 to 0x7fff

• A13, A12 and A11 are connected to the three address inputs of the 74HC138. For example, address Y5 will be low when A13 A12 A11 are {1 0 1}. Thus Y5 will be selected for addresses from **01101**00000000000_2 to **01101**11111111111_2, or from 0x6800 to 0x6fff

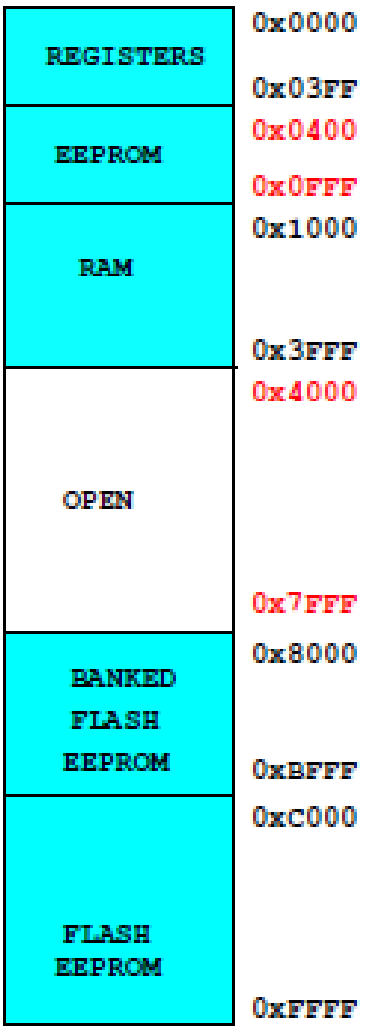
• The Y5 output may be connected to a device which needs only one address

  – In this case, the device will be accessed with any address from 0x6800 to 0x7fff

  – This is called partial address decoding — to save expense, we do not decode all address lines, but only enough to put in the number of devices we need

  – We also need to demultiplex only those address lines we decode, which also saves some expense

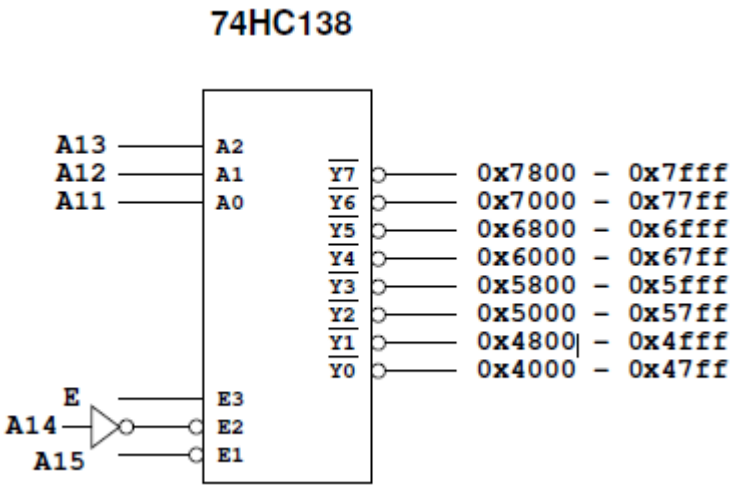
MC9S12 Memory Map
(Expanded Mode)



Can map external device into any unused space

When using Altera for address decoding, can select any desired address.

Can map using less expensive chip like 74HC138.



A Simple Output Port

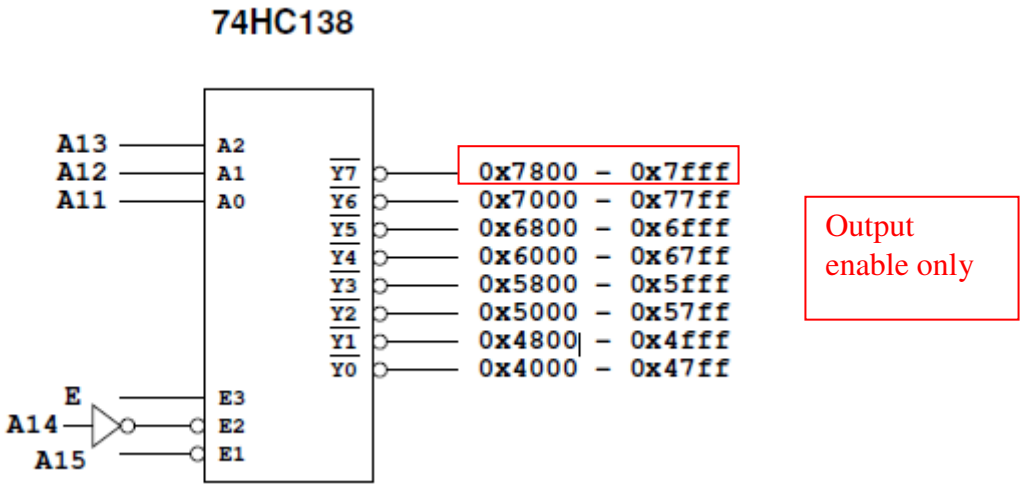
• We will use some MSI chips to implement a simple output port

• We will use a 74AHC374 chip as our output port — this is a chip which has 8 flip-flops

• We need to demultiplex several of the address lines. We will use another 74AHC374 chip to do this

• We will use a 74AHC138 decoder chip to select the output port

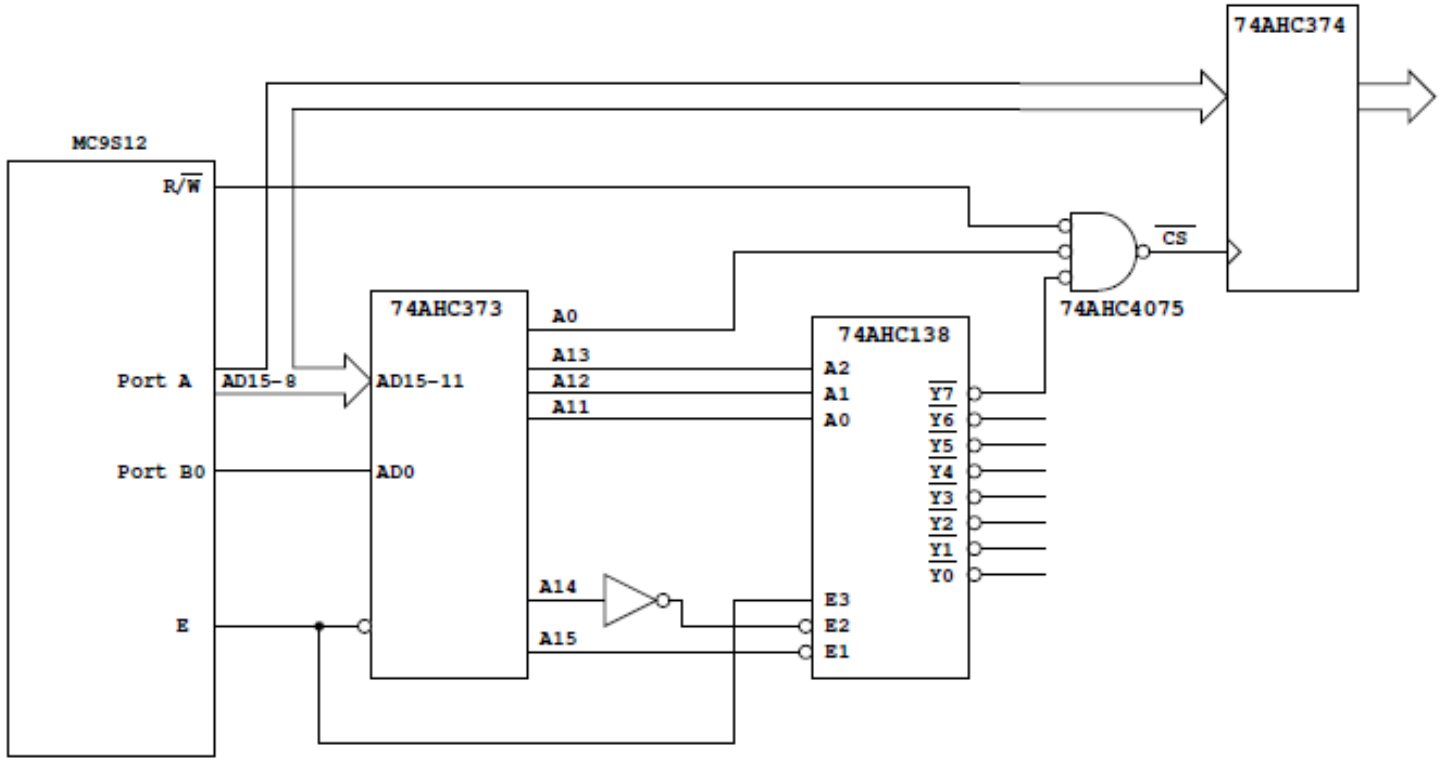
• For this example we will select the output port based on output Y7 from the 74AHC138. For the connections shown of the next page, A15 low and A14 must be high, and A13, A12 and A11 must be high to select output Y7. Thus, the output port will be selected for addresses in the range 0x7800 to 0x7fff



• The address port needs to be connected to 8 bits of the address/data bus. We must connect it to either the high (even) byte (AD15-8) or the low (odd) byte (AD7-0). For this example we will connect it to the high byte

• We need to select the output port only when we write to an even address in the range 0x7800 to 0x7fff. In addition to address Y7 being active (low), the R/W line must be low (to indicate a write) and A0 must be low (to indicate an even address). We will use a 3-input OR gate to do this.

• Now, a write to any even address between 0x7800 and 0x7fff will write data to the output port.



Simple output port for the HC12.

When address is between 0x7800 and 0x7fff and E is high, Y7 will go low

On a write to an even address in this range, CS will go low

Data on AD15–8 will be latched into port when CS goes high (when E goes low)

A Simple Input Port

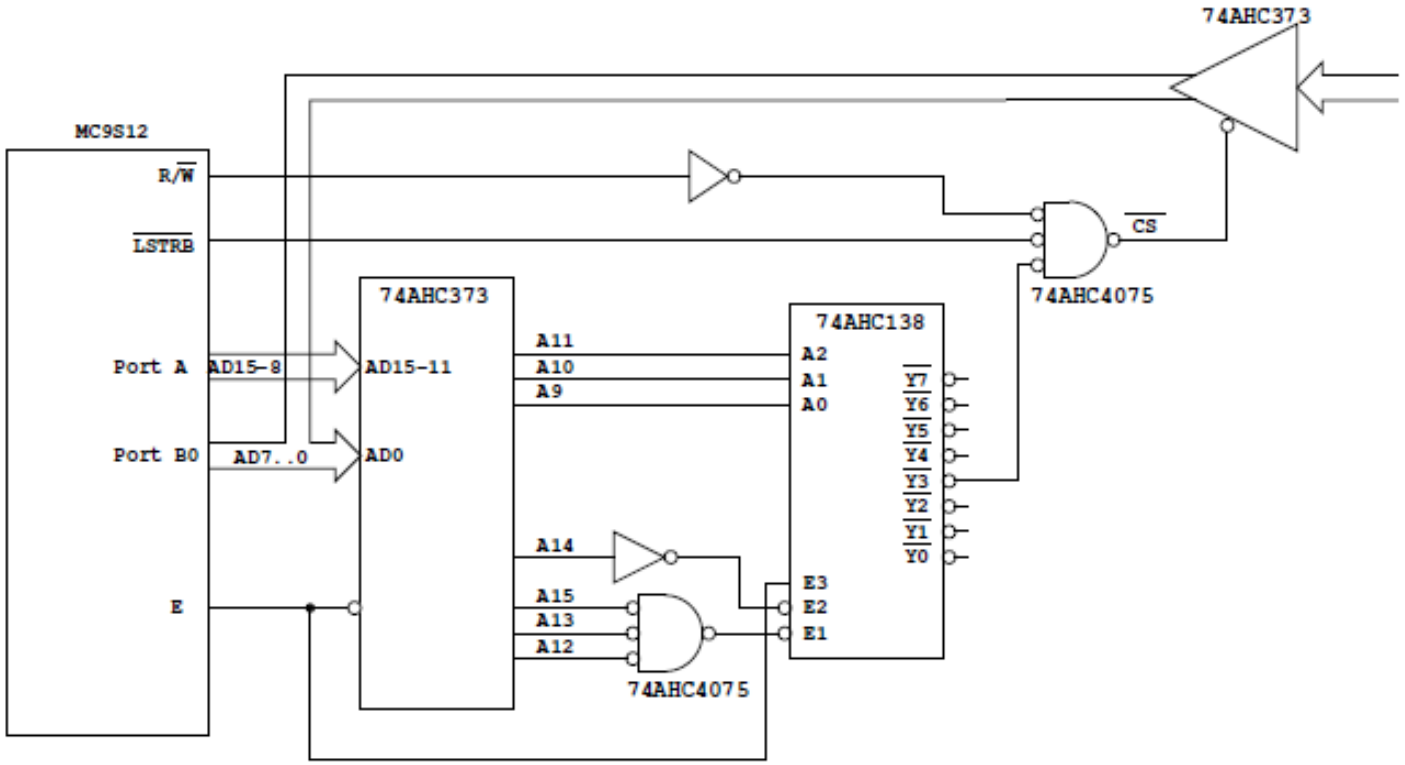
• We will design a simple input port in a similar manner

• We will narrow the range of addresses used by using more high-order address lines to enable the 74AHC138. In the example shown, the address range of a single output is 512 bytes. For the ouput port example above, the range was 2 kB

• For an input port we need a tri-state buffer to drive the input data transparent latch (with tri-state outputs) for our buffer

• In this case we connect the tri-state buffer to the low (odd) data byte

• To access the chip for odd address, we need to select the chip only when $\overline{\text{LSTRB}}$ is low



Simple input port for the HC12.

When address is between 0x4600 and 0x47ff and E is high, Y3 will go low

On a read from an odd address in this range, CS will go low

Data from the tri–state buffer will be driven onto AD7–0. The HC12 will latch the data into internal registers when E goes from high to low

A Simple Input-Output Port

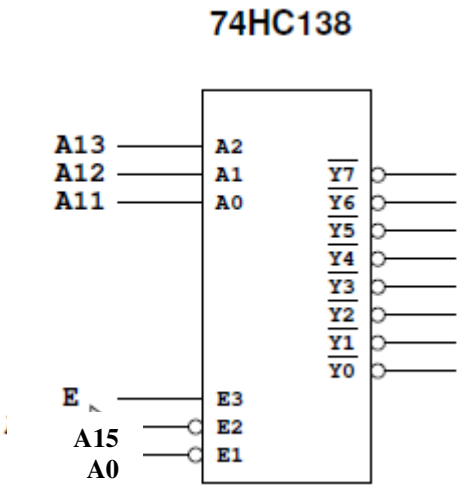
• The next example shows another way to connect the 74HC138 chip to select input and output devices

• We connect A0 to one of the active low enable inputs. The 74HC138 will select a device only when A0 is low — only for even addresses

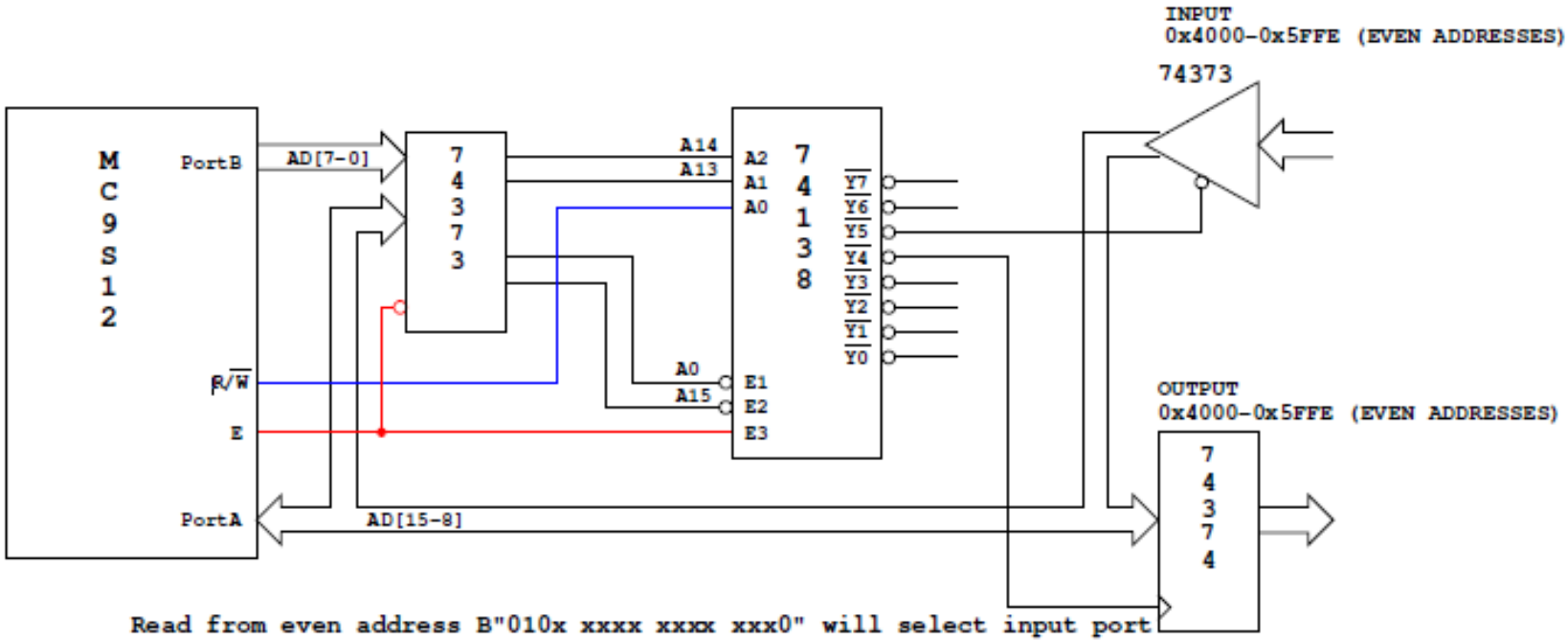
• We connect R/W to the low-order address input of the 74HC138

– To select Y0, Y2, Y4 or Y6, R/W must be low. Thus, the devices selected by Y0,Y2, Y4 and Y6 are ouput devices (devices the MC9S12 writes to).

– To select Y1, Y3, Y5 or Y7, R/W must be high. Thus, the devices selected by Y1,Y3, Y5 and Y7 are input devices (devices the MC9S12 reads from).



Simple Input and Output Ports



Read from even address B"010x xxxx xxxx xxx0" will select input port

Write to even address B"010x xxxx xxxx xxx0" will latch data in output port