- **The MC9S12 in Expanded Mode – Using MSI logic to build ports**
- Using MSI logic to build an output port
- Using MSI logic to build an input port

## A Simple Parallel Output Port

- We want a port which will write 8 bits of data to the outside

- Such a port is similar to Port A or Port B when all pins are set up as output

- <u>We need some hardware to latch the output data at the time the MC9S12 puts the data on the data bus</u>

- We can use a set of 8 D flip-flops to latch the data

    - The D inputs will be connected to the data bus

    - The clock to latch the flip-flops should make its low-to-high transition when the MC9S12 has the appropriate data on the bus

    - The MC9S12 will access the flip-flops by writing to an address. We must assign an address for the flip-flops

    - We must have hardware to demultiplex the address from the data, and to determine when the MC9S12 is writing to this address
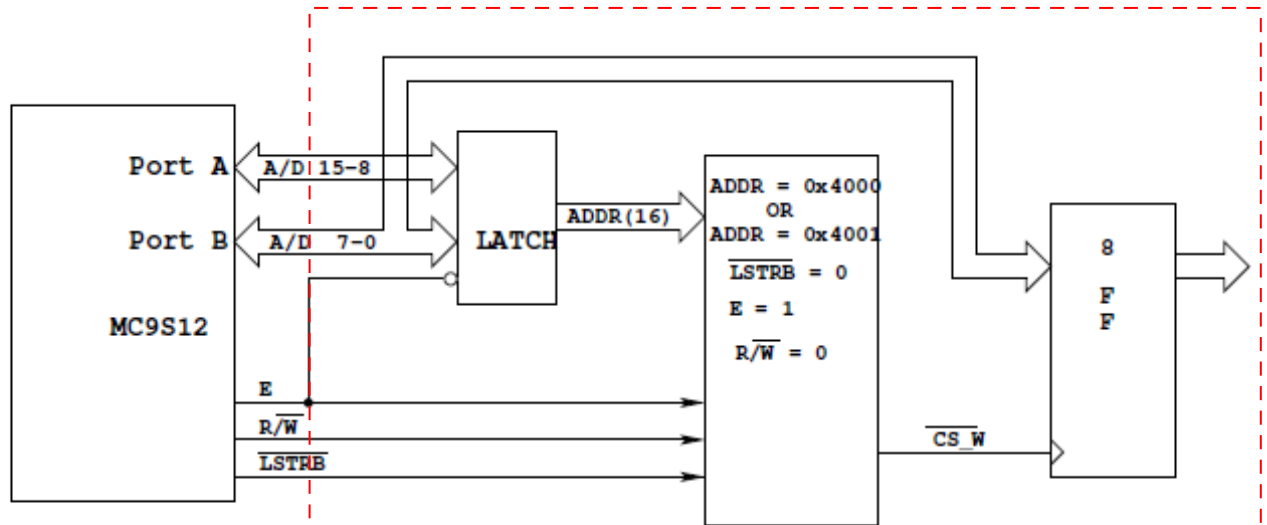
- The 8-bit inputs of the D flip-flops will be connected to 8 bits of the 16-bit address/data bus of the MC9S12

    * If the address of the input is even, we need to connect the flip flop inputs to the even (high) byte of the bus, which is connected to AD15-8 (what was Port A)

    * If the address of the input is odd, we need to connect the flip flop inputs to the odd (low) byte of the bus, which is connected to AD7-0 (what was Port B)
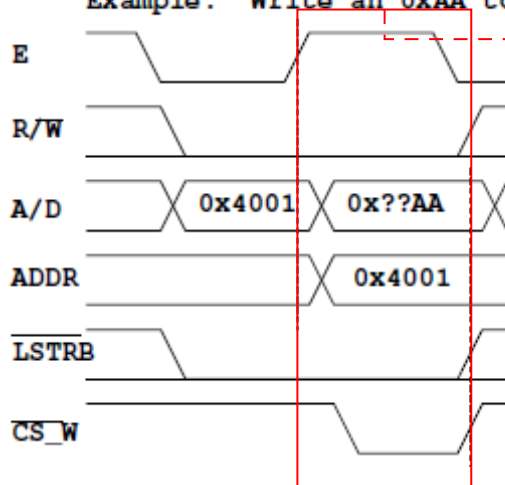
- The hardware should latch the data on the high-to-low transition of the E-clock

- Our hardware should bring the clock of the flip-flops low when

    **1. The address of the flip-flops is on the address bus**
    **2. The MC9S12 is writing to this address**
    **3. The MC9S12 is writing the high byte if the address is even, or the low byte if the address is odd**
    **4. E-clock is high**

- For example, consider an output port at address 0x4001 (an odd address, or low byte):

Port A    A/D 15-8
Port B    A/D 7-0

MC9S12

LATCH    ADDR(16)

ADDR = 0x4000
OR
ADDR = 0x4001
$\overline{\text{LSTRB}}$ = 0
E = 1
$R/\overline{W}$ = 0

8
F
F

E
$R/\overline{W}$
$\overline{\text{LSTRB}}$

$\overline{\text{CS\_W}}$

Example:  Write an 0xAA to address 0x4001

E

R/W

A/D        0x4001    0x??AA

ADDR             0x4001

$\overline{\text{LSTRB}}$
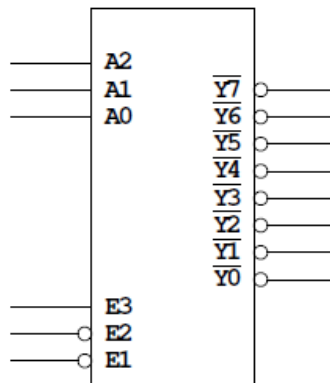
$\overline{\text{CS\_W}}$

Note:  ADDR can be 0x4000 or 0x4001
       with LSTRB = 0

**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

**74HC138**



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.

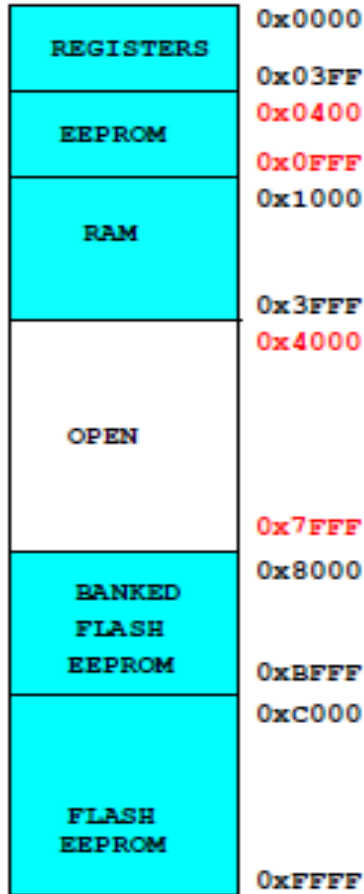| A2 | A1 | A0 | Output |
|----|----|----|--------|
| 0 | 0 | 0 | Y0 |
| 0 | 0 | 1 | Y1 |
| 0 | 1 | 0 | Y2 |
| 0 | 1 | 1 | Y3 |
| 1 | 0 | 0 | Y4 |
| 1 | 0 | 1 | Y5 |
| 1 | 1 | 1 | Y7 |

## 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 is low and A14 is high, 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 **0110 1**000 0000 0000$_2$ to **0110 1**111 1111 1111$_2$, or from 0x6800 to 0x6fff

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

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

> – This is called <u>partial address decoding</u> — 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
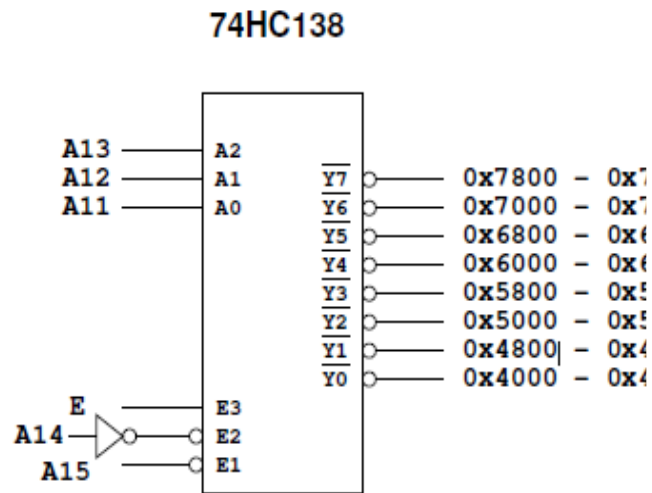
MC9S12 Memory Map
(Expanded Mode)

```
              0x0000
  REGISTERS
              0x03FF
              0x0400
  EEPROM
              0x0FFF
              0x1000
  RAM

              0x3FFF
              0x4000


  OPEN


              0x7FFF
              0x8000
  BANKED
  FLASH
  EEPROM
              0xBFFF
              0xC000


  FLASH
  EEPROM
              0xFFFF
```

Can map external device into any unused space

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

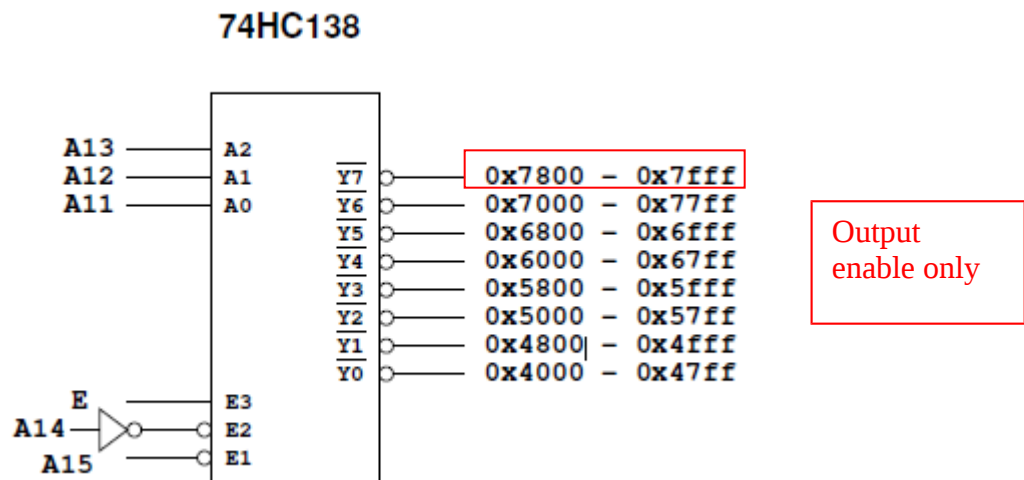Can map using less expensive chip like 74HC138.

**74HC138**

```
A13 ──── A2
A12 ──── A1          Y7 ○── 0x7800 - 0x7
A11 ──── A0          Y6 ○── 0x7000 - 0x7
                     Y5 ○── 0x6800 - 0x6
                     Y4 ○── 0x6000 - 0x6
                     Y3 ○── 0x5800 - 0x5
                     Y2 ○── 0x5000 - 0x5
                     Y1 ○── 0x4800 - 0x4
                     Y0 ○── 0x4000 - 0x4

  E  ──────── E3
A14 ─▷○─────○ E2
A15 ───────○ E1
```

Want to select external device when E is high – Connect E to E3

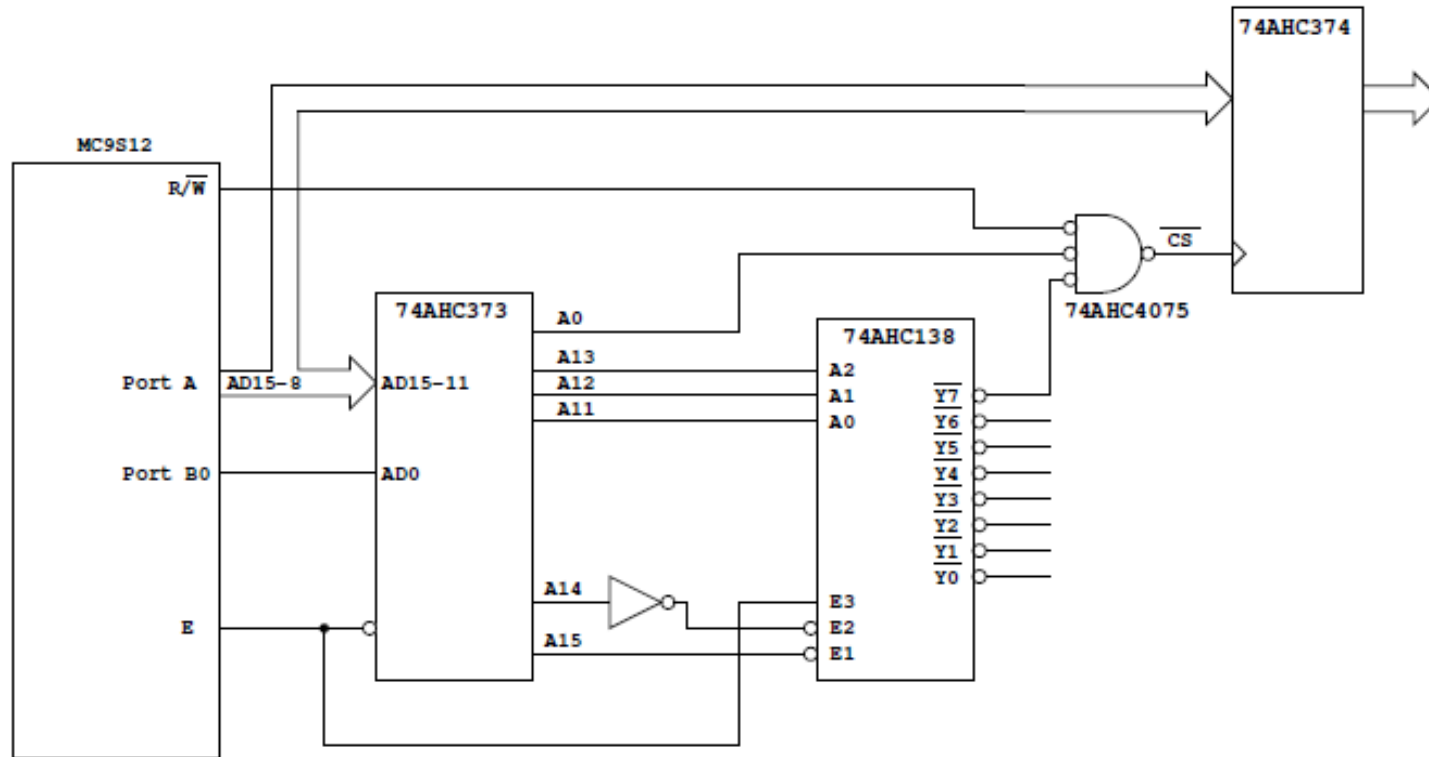Will use A15 and A14 as the other enables. This will divide memory into ¼ of total.

| A15 | A14 | Chip enabled? |
|-----|-----|---------------|
| 0 | 0 | 0x0000 - 0x3fff |
| 0 | 1 | 0x4000 - 0x7fff |
| 1 | 0 | 0x8000 - 0xbfff |
| 1 | 1 | 0xc000 - 0xffff |

## 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

74HC138

```
A13 ———— A2
A12 ———— A1        Y̅7̅ o——— 0x7800 - 0x7fff
A11 ———— A0        Y̅6̅ o——— 0x7000 - 0x77ff
                   Y̅5̅ o——— 0x6800 - 0x6fff          Output
                   Y̅4̅ o——— 0x6000 - 0x67ff          enable only
                   Y̅3̅ o——— 0x5800 - 0x5fff
                   Y̅2̅ o——— 0x5000 - 0x57ff
                   Y̅1̅ o——— 0x4800 - 0x4fff
                   Y̅0̅ o——— 0x4000 - 0x47ff

      E ———— E3
   A14 —▷o—o E2
   A15 ———o E1
```

• 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.

**74AHC374**

MC9S12

R/W̄

74AHC373

74AHC138

74AHC4075

CS̄

Port A    AD15–8    AD15–11

A0
A13    A2    Y7̄
A12    A1    Y6̄
A11    A0    Y5̄

Port B0    AD0    Y4̄
Y3̄
Y2̄
Y1̄
Y0̄

A14    E3
E    E2
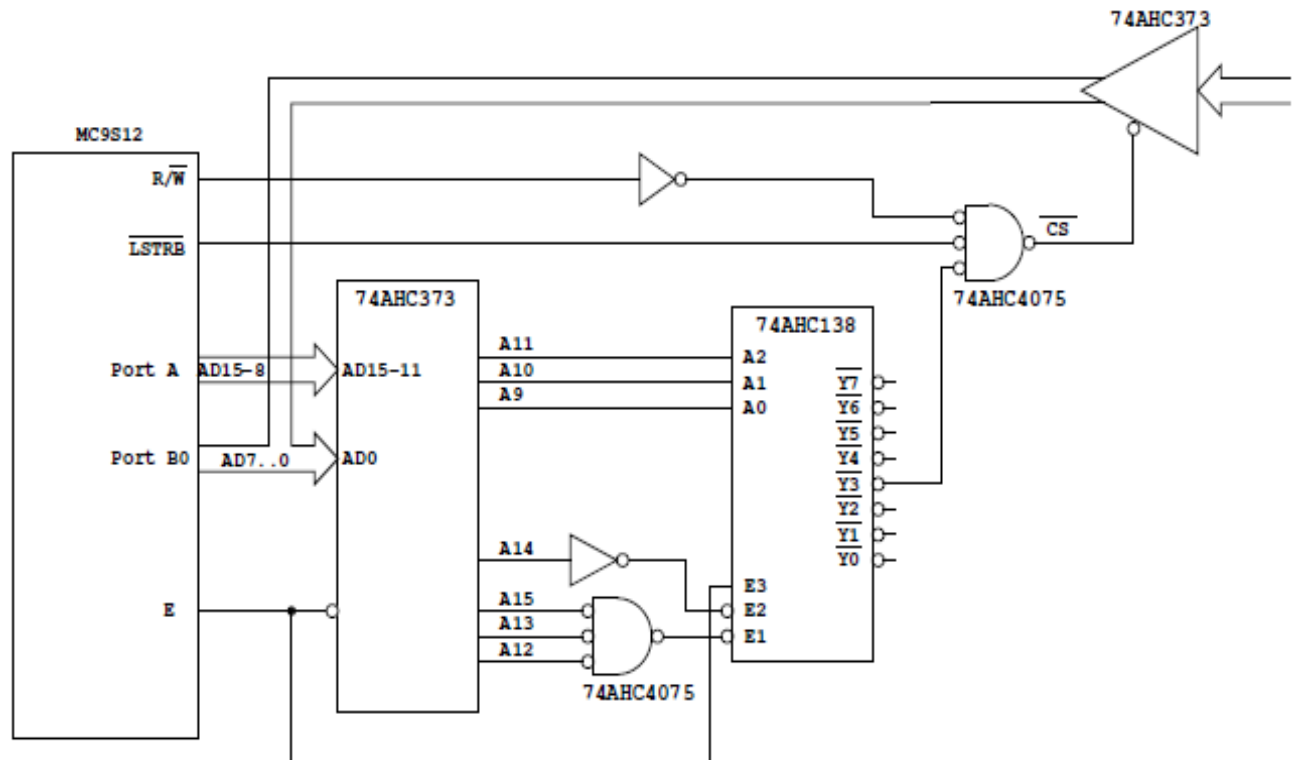A15    E1

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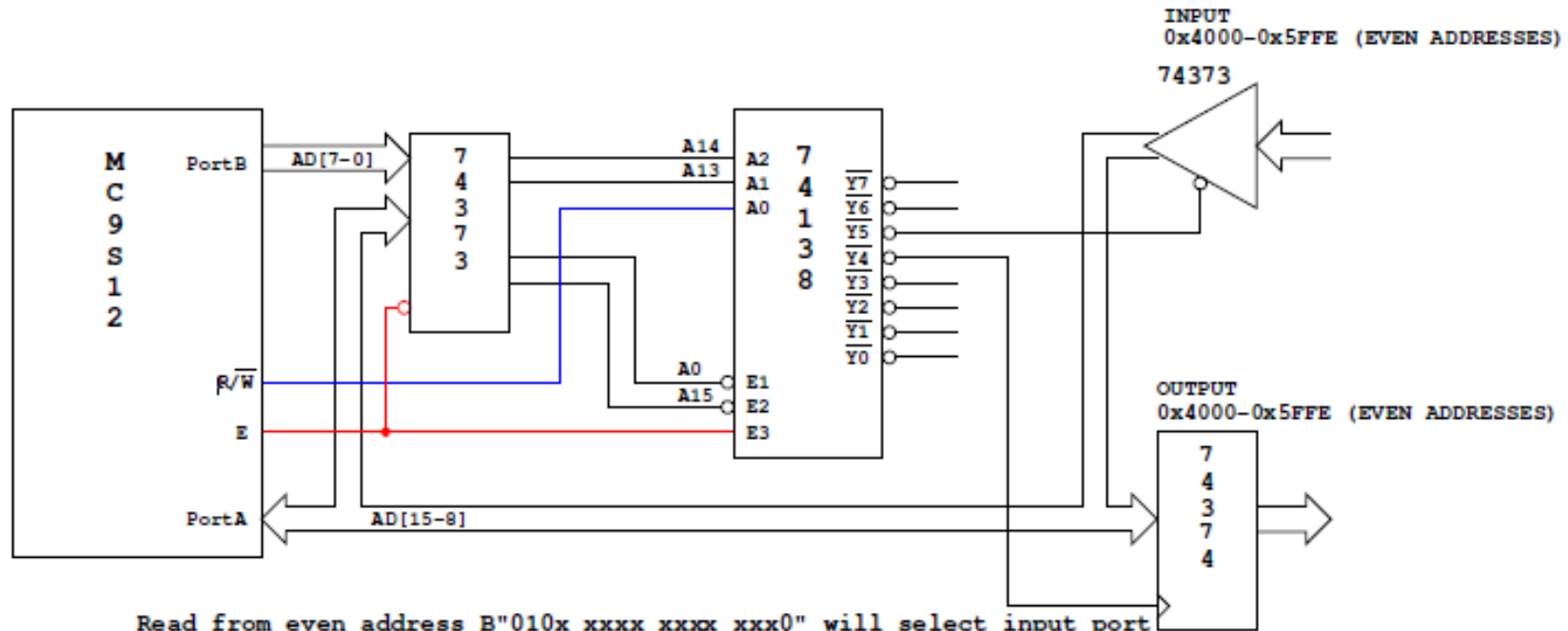
On a read from an odd address in this range, CS will go low

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 <u>ouput devices</u> (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 <u>input devices</u> (devices the MC9S12 reads from).

## Simple Input and Output Ports

INPUT
0x4000–0x5FFE (EVEN ADDRESSES)

74373

| | |
|---|---|
| M C 9 S 1 2 | PortB |

AD[7–0]

7 4 3 7 3

A14
A13

A2
A1
A0

7 4 1 3 8

$\overline{Y7}$
$\overline{Y6}$
$\overline{Y5}$
$\overline{Y4}$
$\overline{Y3}$
$\overline{Y2}$
$\overline{Y1}$
Y0

R/$\overline{W}$

E

A0   E1
A15  E2
     E3

PortA

AD[15–8]

OUTPUT
0x4000–0x5FFE (EVEN ADDRESSES)

7 4 3 7 4

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