The 9S12 in Expanded Mode - External Ports

Huang Chapter 14

Accessing External Memory and Ports on the 9S12 in Expanded Mode

• In expanded mode, the 9S12 has a multiplexed 16-bit address and data bus.

• With a 16-bit address bus, the 9S12 can access 2^{16} = 65, 536 bytes of data

• With a 16-bit data bus, the 9S12 can access 16 bits (two bytes) in a single bus cycle

• In expanded mode, the 9S12 uses Port A and Port B as the multiplexed address/data bus

• Timing is controlled by the E clock

• When the E clock is low, the 9S12 places the address on the multiplexed bus
– Port A is used for address bits 15-8
– Port B is used for address bits 7-0

• When the E clock is high, the 9S12 uses the multiplexed bus for data bus:
– Port A is used for the even byte
– Port B is used for the odd byte

For example, if accessing the sixteen-bit word at address 0x4000 (the bytes at addresses 0x4000 and 0x4001), Port A will access the byte at address 0x4000, and Port B will access the byte at address 0x4001.

Simple Parallel Input Port

• We want a port which will read 8 bits of data from the outside

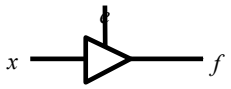
• Such a port is similar to Port A or Port B when all pins are set up as input

• We need some hardware to drive the input data onto the data bus at the time the 9S12 needs it to be there to read it

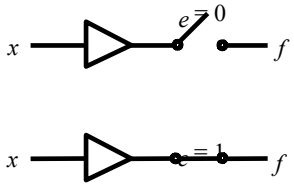
• The hardware needs to keep the data off the bus at all other times so it doesn't interfere with data from other devices

• A tri-state buffer can be used for this purpose
– A tri-state buffer has three output states: logic high, logic low, and high impedance (high-Z)

– In high-Z state it is like the buffer is not connected to the output at all, so another device can drive the output
– a tri-state output acts like a switch — when the switch is closed, the output logic level is the same as the input logic level, and when the switch is open, the buffer does not change the logic level on the output pin
– A tri-state buffer has a control input which, when active, drives the input logic levels onto the output pins, and when inactive, opens the switch.



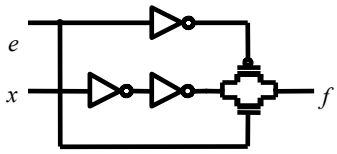
(a) A tri-state buffer



(b) Equivalent circuit

e	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

(c) Truth table



(d) Implementation

A Simple Parallel Input Port

• When should the tri-state buffer be enabled to drive the data bus?

– The 9S12 will access the buffer by reading from an address. We must assign an address for the tri-state buffer

– We must have hardware to demultiplex the address from the data, and to determine when the 9S12 is reading from this address

– The 8-bit input will be connected to 8 bits of the 16-bit address/data bus of the 9S12

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

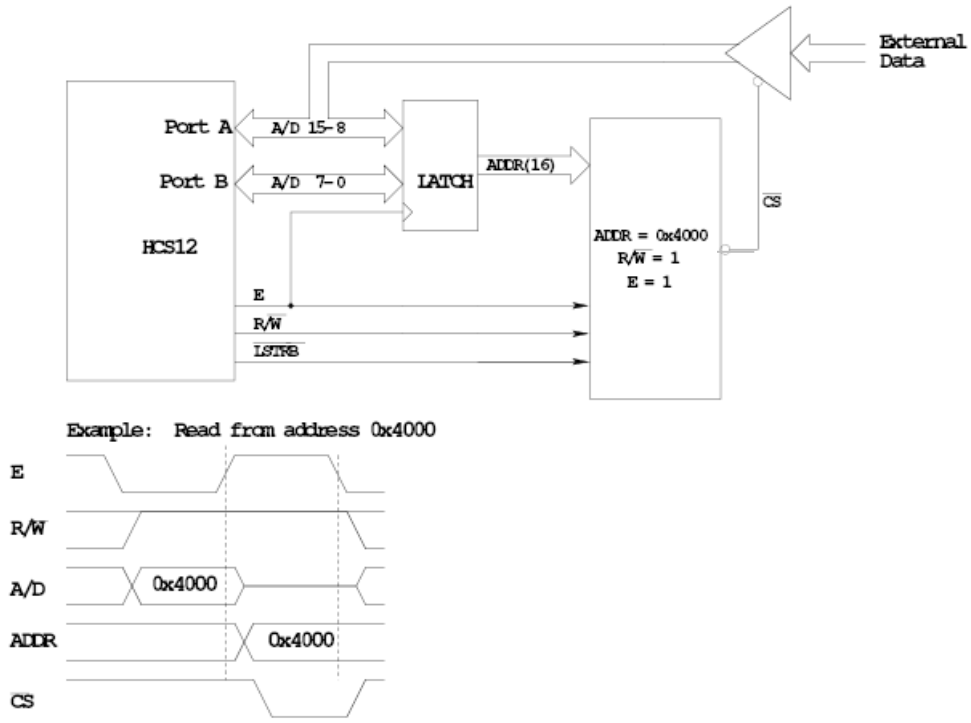
- If the address of the input is odd, we need to connect the output of the buffer to the odd (low) byte of the bus, which is connected to AD7-0 (what was Port B)

– The 9S12 needs the data on the bus on the high-to-low transition of the E-clock
–

– We must enable the tri-state buffer when

1. The address of the buffer is on the address bus
2. The 9S12 is reading from this address
3. The 9S12 is reading the high byte if the address is even, or the low byte if the address is odd
4. E is high

• For example, consider an input port at address 0x4000 (an even address, or high byte):



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

• We need some hardware to latch the output data at the time the 9S12 puts the data on the data bus

• 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 9S12 has the appropriate data on the bus

– The 9S12 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 9S12 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 9S12

- If the address of the output 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)

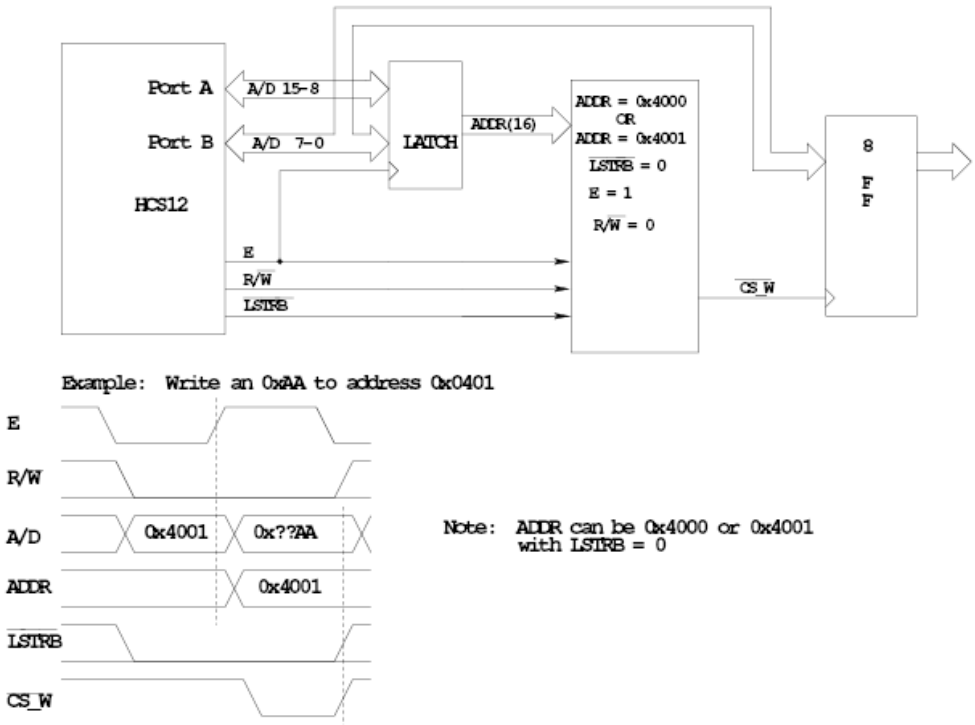
- If the address of the output is odd, we need to connect the flip flop inputs to the odd (low) byte of the bus, which is connected to AD7-0 (what was Port B)

– 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 9S12 is writing to this address
3. The 9S12 is writing the high byte if the address is even, or the low byte if the address is odd
4. E is high

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



An Output Port Which Can Be Read

• Suppose we set up the 9S12 Port B for output, and we write a number to Port B

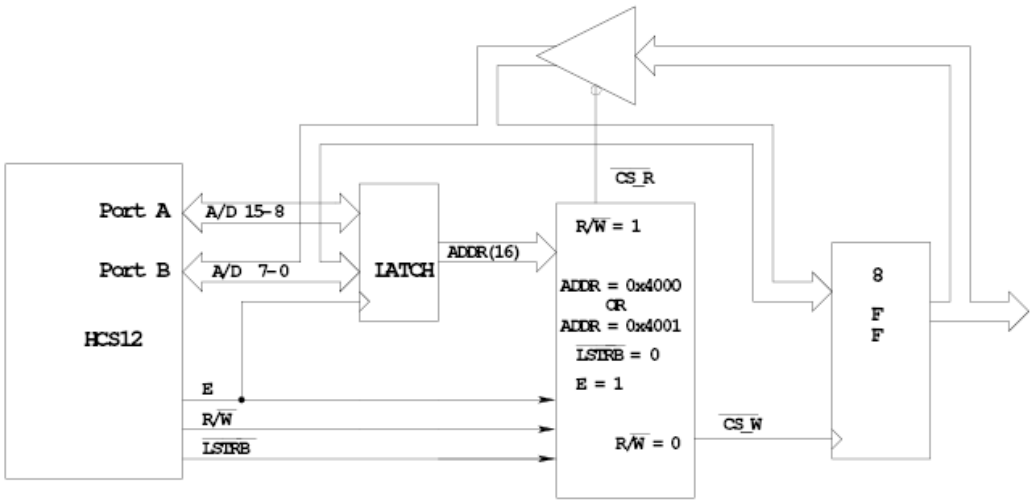
• When we read from Port B, we will read back the number we wrote

• This is a useful diagnostic

• We can make our output port have this same performance by connecting the output of the flip-flops back into the data bus through a tri-state buffer

• We should enable this tri-state buffer when the 9S12 is reading from the address of the output port

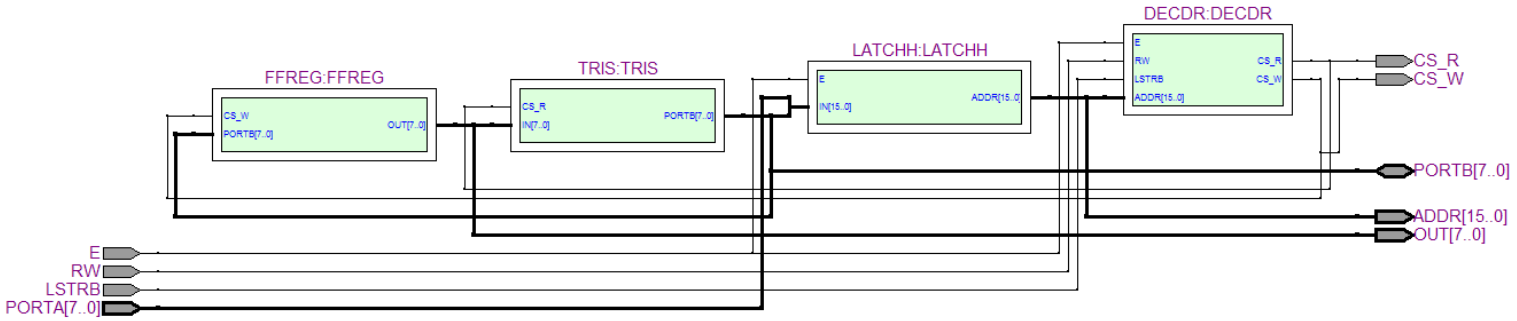
• For example, consider the output port at address 0x4001:



Writing to address 0x4001 will bring CS_W low.
On the high−to−low transition of E, CS_W will go high, latching the data into the flip−flops

Reading from address 0x4001 will bring CS_R low
This will drive the data from the flip−flops onto the data bus
The HC12 will read the data on the flip−flops on the high−to−low transition of the E−clock

The RTL diagram of the Input-Output Port



A simulation of the Input-Output Port