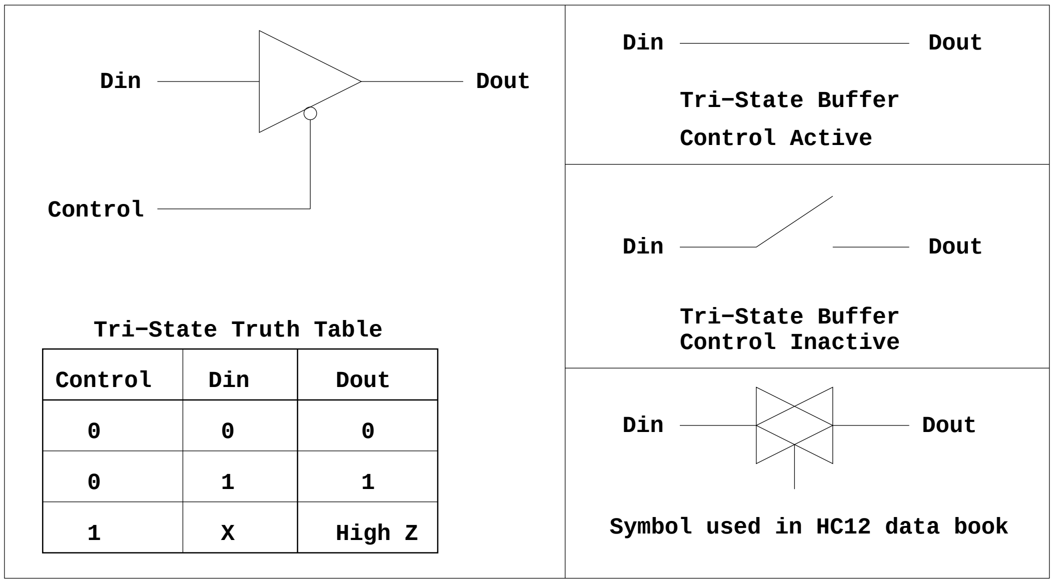
A Simple Parallel Input Port

- 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 HC12 needs it to be there to read
- 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 state: 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

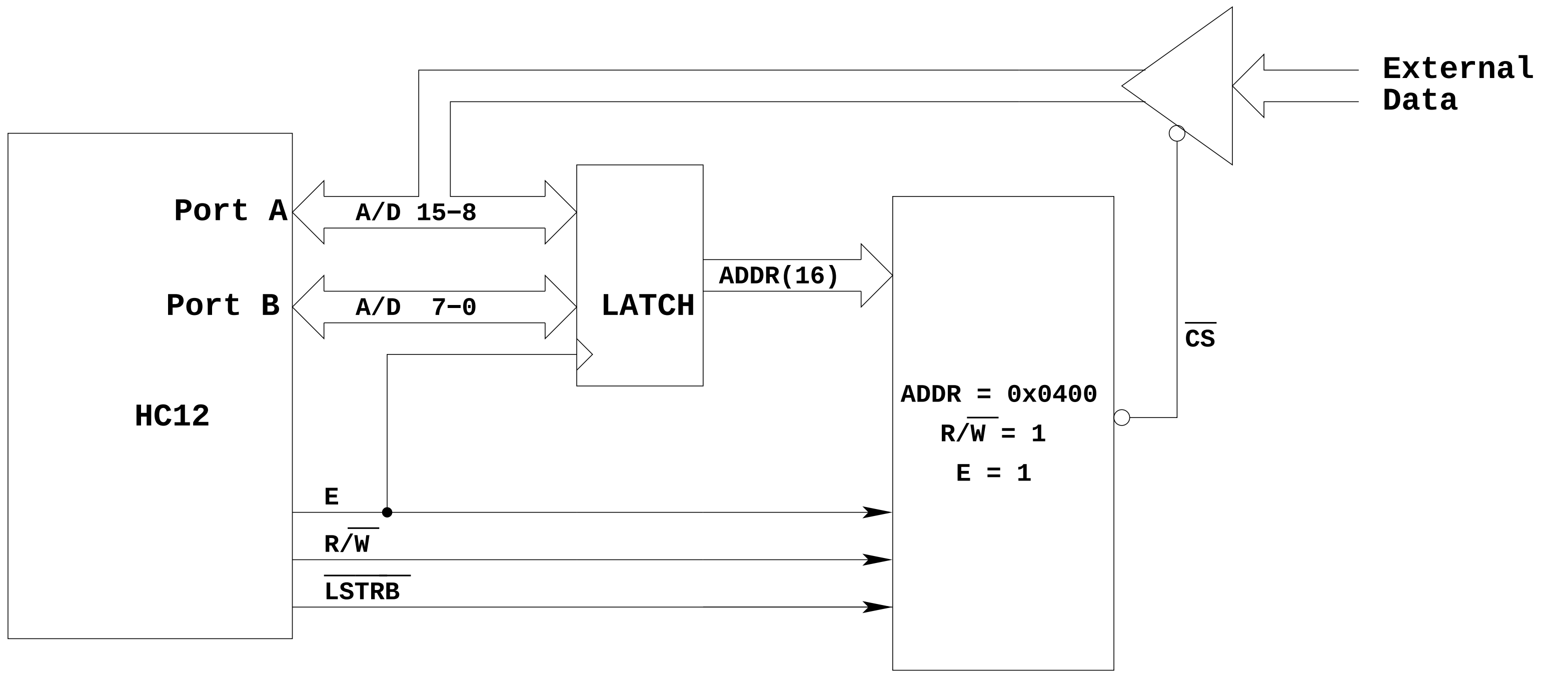


Tri−State Truth Table

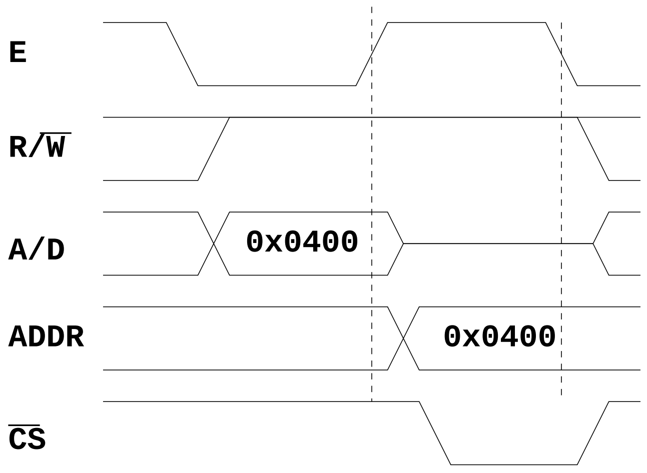
Control	Din	Dout
0	0	0
0	1	1
1	X	High Z

A Simple Parallel Input Port

- When should the tri-state buffer be enabled to drive the data bus?
  - The HC12 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 HC12 is reading from this address
  - The 8-bit input will be connected to 8 bits of the 16-bit address/data bus of the HC12
    - ∗ 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 HC12 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 HC12 is reading from this address
    3. The HC12 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 0x0400 (an even address, or high byte):

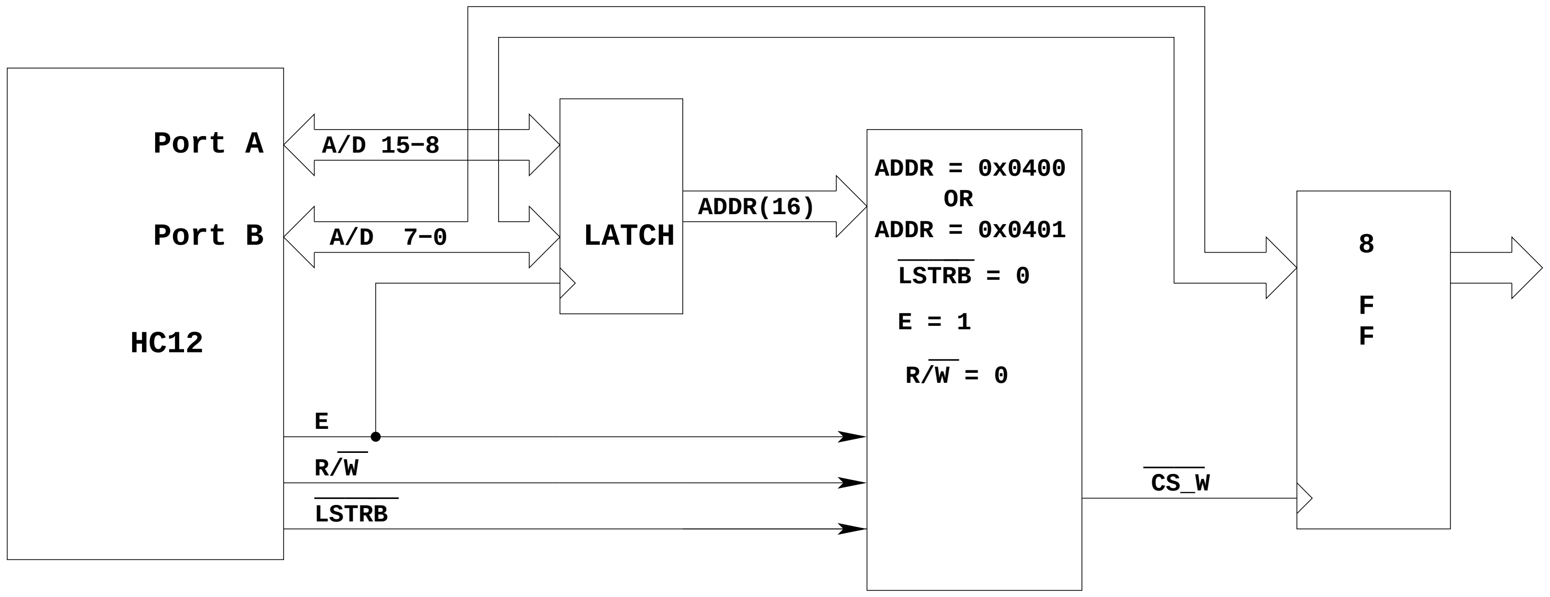


Example: Read from address 0x0400

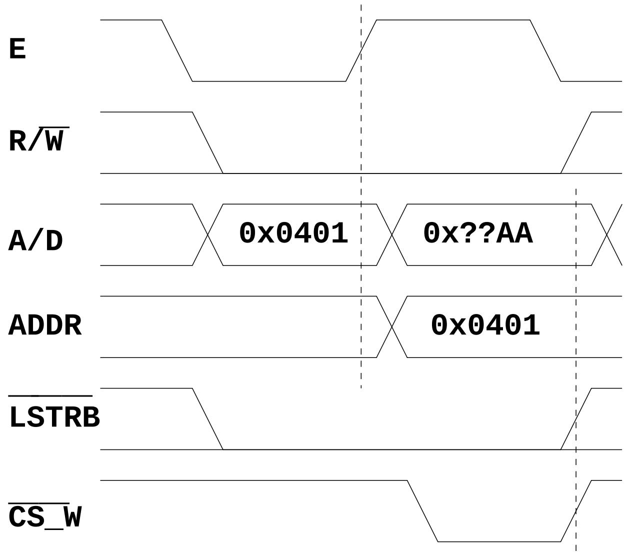


A Simple Parallel Output Port

- 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 HC12 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 HC12 has the appropriate data on the bus
  - The HC12 will access the flip-flops by writing to 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 HC12 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 HC12
    - ∗ 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 HC12 is writing to this address
    3. The HC12 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 0x0401 (an odd address, or low byte):


HC12
Port A
Port B
A/D 15–8
A/D 7–0
LATCH
ADDR(16)
ADDR = 0x0400
OR
ADDR = 0x0401
$\overline{\text{LSTRB}}$ = 0
E = 1
R/$\overline{\text{W}}$ = 0
E
R/$\overline{\text{W}}$
$\overline{\text{LSTRB}}$
$\overline{\text{CS\_W}}$
8
F
F

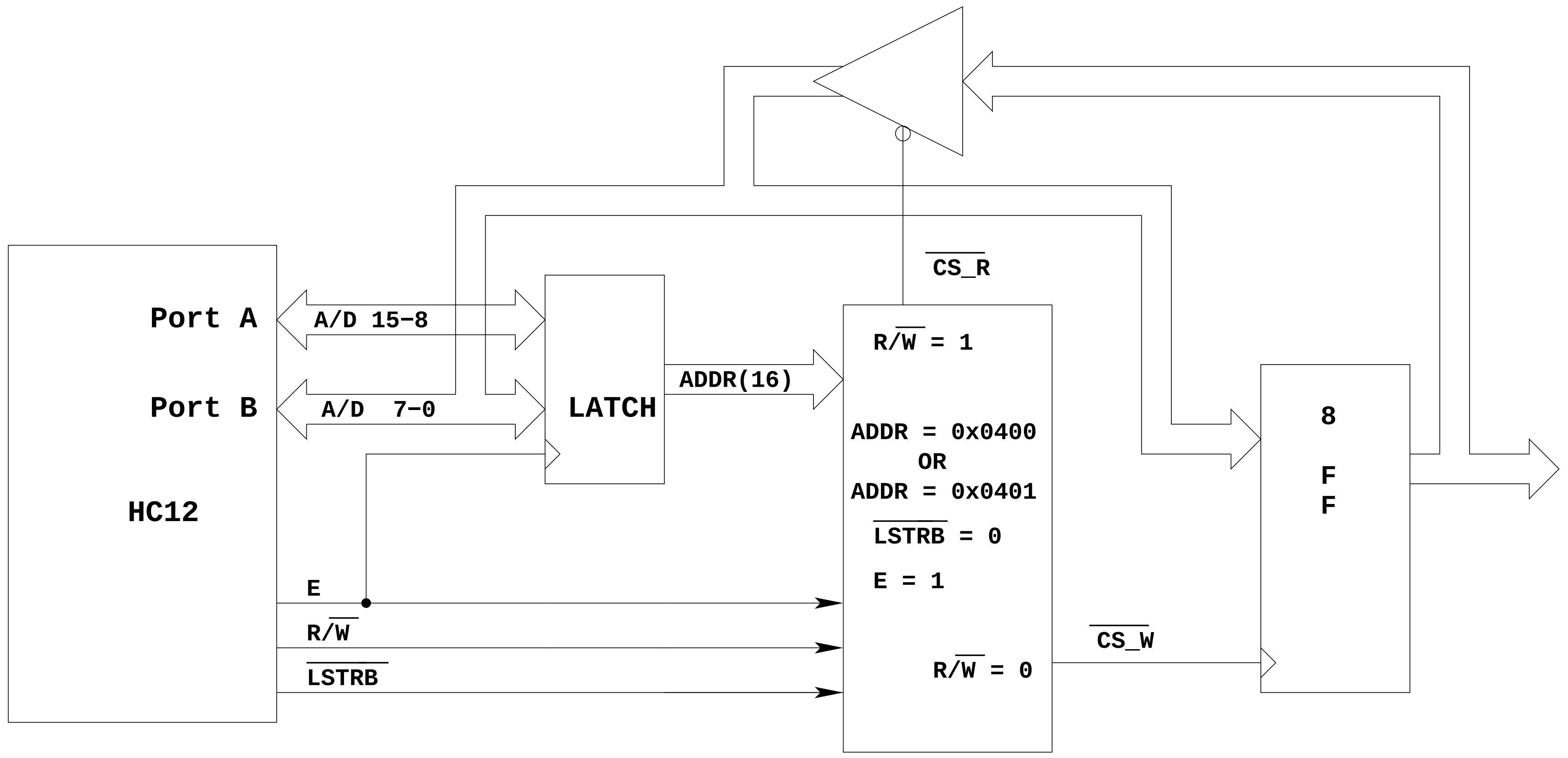
Example: Write an 0xAA to address 0x0401



Note: ADDR can be 0x0400 or 0x0401 with LSTRB = 0

An Output Port Which Can Be Read

- Suppose we set ut the HC12 Port A for output, and we write a number to Port A
- When we read from Port A, we will read back the number we wrote
- This is a useful diagnostic
- We can make our output port have this same behaviour 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 HC12 is reading from the address of the output port
- For example, consider the output port at address 0x0401:



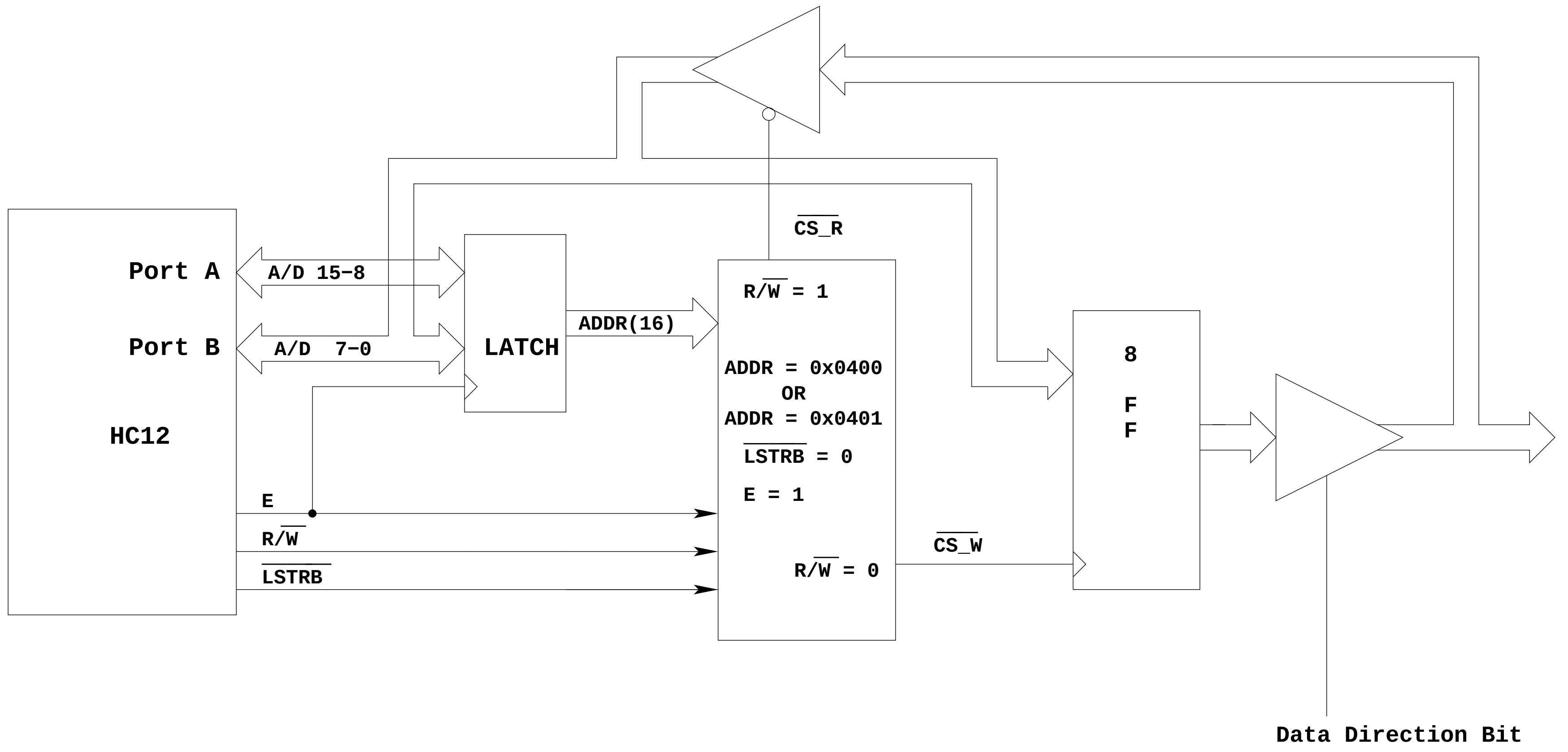
Writing to address 0x0401 will bring CS_W low.

On the high−to−low transition of E, CS_W will go high, latching the data into the flip−flops

Reading from address 0x401 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

An Input-Output Port

- Like Port A, we can make a port be either input or output
- For simplicity, we will make all bits inputs or all bits outputs rather than allowing any individual bit to be either an input or an output
- To do this we need a data direction bit (at another address), and a tri-state buffer on the outputs of the flip-flops
- The data direction bit is simply a flip-flop which is set or cleared by the HC12
- When the data direction bit is cleared, the data from the output flip-flops will be removed from the external pins
  - When we read from the port, we will read the logic levels on the pins put there by external logic
- When the data direction bit is set, the data from the output flip-flops will be removed from the external pins
  - When we write to the port, we will drive the data from the flip-flops onto the external pins
  - For example consider an I/O port at address 0x0401. The direction of the port is determined by a data direction bit at address 0x0402:



The data direction bit is the output of a flip−flop which was written to at another address of the HC12
For example, it could be Bit 4 of address 0x0402

Writing a 0 to Bit 4 of address 0x0402 disables the output tri−state buffer
- When we write to address 0x0401 we will latch the data into the flip−flops
- This data will not be driven onto the external pins
- When we read from address 0x0401, we will read what an external device drives onto the pins

Writing a 1 to Bit 4 of address 0x042 enables the output tri−state buffer
- When we write to address 0x0401 we will latch the data into the flip−flops
- This data will be driven onto the external pins
- When we read from address 0x0401 we will read the data latched into the flip−flops