# 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.

• Sometimes you only want to access one byte at a time. For example,
– ldaa $4001
will access the single byte at address 0x4001.

• To determine whether it should access one byte or two bytes, the 9S12

uses the $\overline{\text{LSTRB}}$ and A0 lines.

– $\overline{\text{LSTRB}}$ low means that the 9S12 is accessing the lower (odd) byte of a sixteen-bit word

– $\overline{\text{LSTRB}}$ high means that the 9S12 is accessing the upper (even) byte of a sixteen-bit word

– A0 low means that the 9S12 is accessing the upper (even) byte

– A0 high means that the 9S12 is accessing the lower (odd) byte

| $\overline{\text{LSTRB}}$ | A0 | Type of Access |
|---|---|---|
| 0 | 0 | 16-bit access of an even address<br>Accesses bytes at even address and subsequent odd address |
| 0 | 1 | 8-bit access of an odd address |
| 1 | 0 | 8-bit access of an even address |
| 1 | 1 | Not allowed on external bus |

• The instruction
– ldaa $4000
accesses the byte at address 0x4000, but doesn't access the byte at address 0x4001. For this access, the 9S12 will put 0x4000 on the bus (A0 = 0, access even byte), and will make $\overline{\text{LSTRB}}$ = 1 (don't access odd byte).

• The instruction
– ldaa $4001
accesses the byte at address 0x4001, but doesn't access the byte at address 0x4000. For this access, the 9S12 will put 0x4001 on the bus (A0 = 1, access odd byte), and will make $\overline{\text{LSTRB}}$ = 0 (access odd byte).

• The instruction
– ldd $4000
accesses the bytes at addresses 0x4000 and 0x4001. For this access, the 9S12 will put 0x4000 on the bus (A0 = 0, access even byte), and will make $\overline{\text{LSTRB}}$ = 0 (access odd byte).

## Simple Parallel Input Port

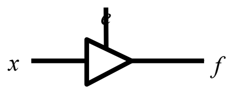
• We want a port which will read 8 bits of data from the outside

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

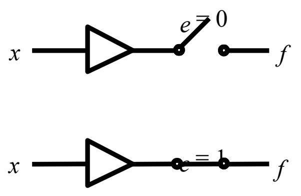
• We need some hardware to drive the input data onto the data bus at the time the 9S12 needs it to be there to read it

• 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.



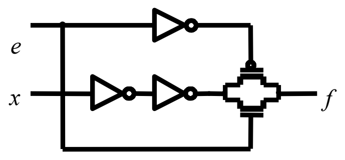
(a) A tri-state buffer



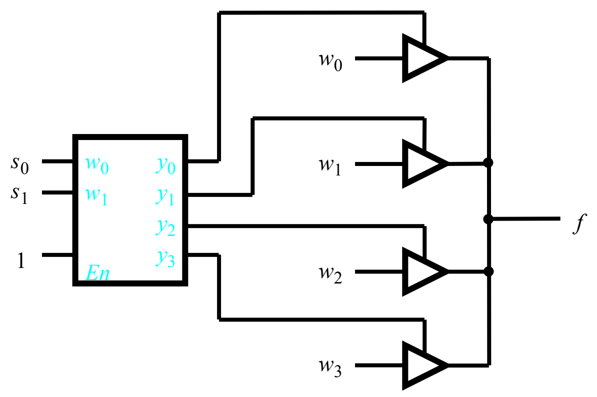
(b) Equivalent circuit

| e | x | f |
|---|---|---|
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(c) Truth table



(d) Implementation



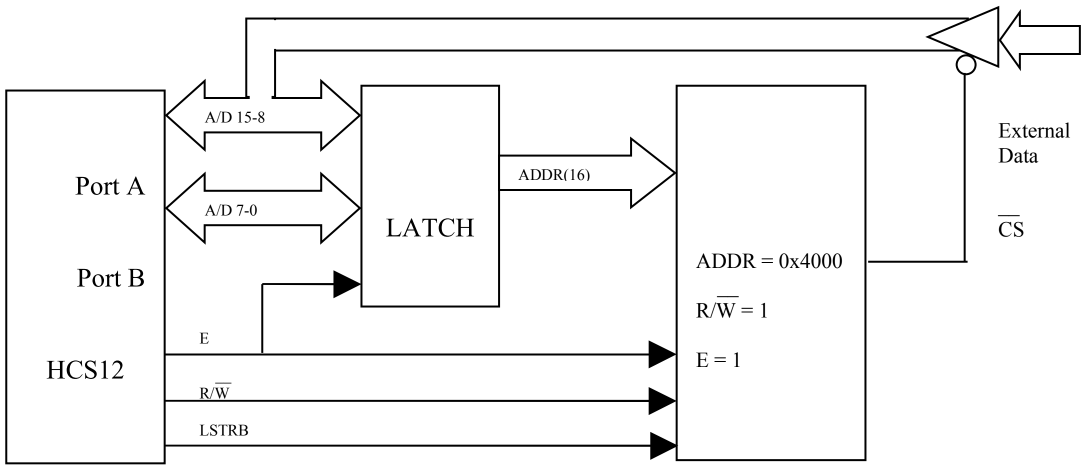
**Multiplexer built using a decoder and tri-state buffers**

## A Simple Parallel Input Port

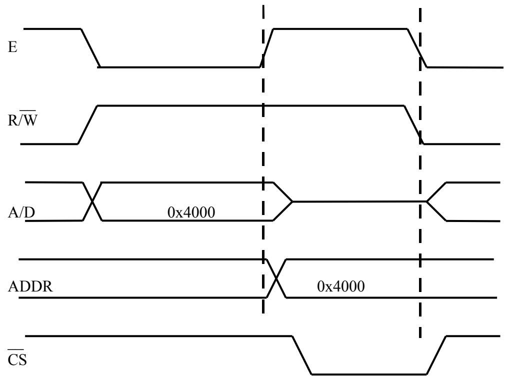
• When should the tri-state buffer be enabled to drive the data bus?
– The 9S12 will access the buffer by reading from an address. We must assign an address for the tri-state buffer
– We must have hardware to demultiplex the address from the data, and to determine when the 9S12 is reading from this address
– 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):



**Example: Read from address 0x4000**



## A Simple Parallel Output Port

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

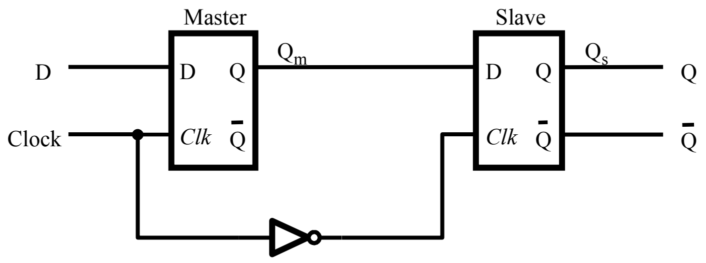
• Such a port is similar to Port A or Port B when all pins are set up as output

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

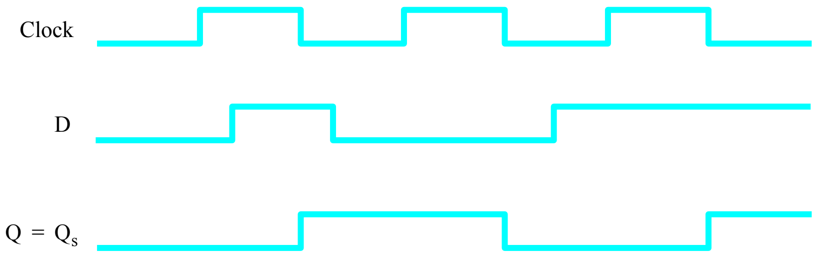
• We can use a set of 8 D flip-flops to latch the data
– The D inputs will be connected to the data bus
– The clock to latch the flip-flops should make its low-to-high transition when the 9S12 has the appropriate data on the bus
– The 9S12 will access the flip-flops by writing to an address. We must assign an address for the tri-state buffer
– 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 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

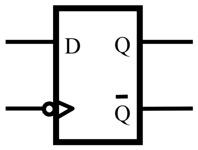
1. The address of the flip-flops is on the address bus
2. The 9S12 is writing to this address
3. The 9S12 is writing the high byte if the address is even, or the low byte if the address is odd
4. E is high



(a) Circuit



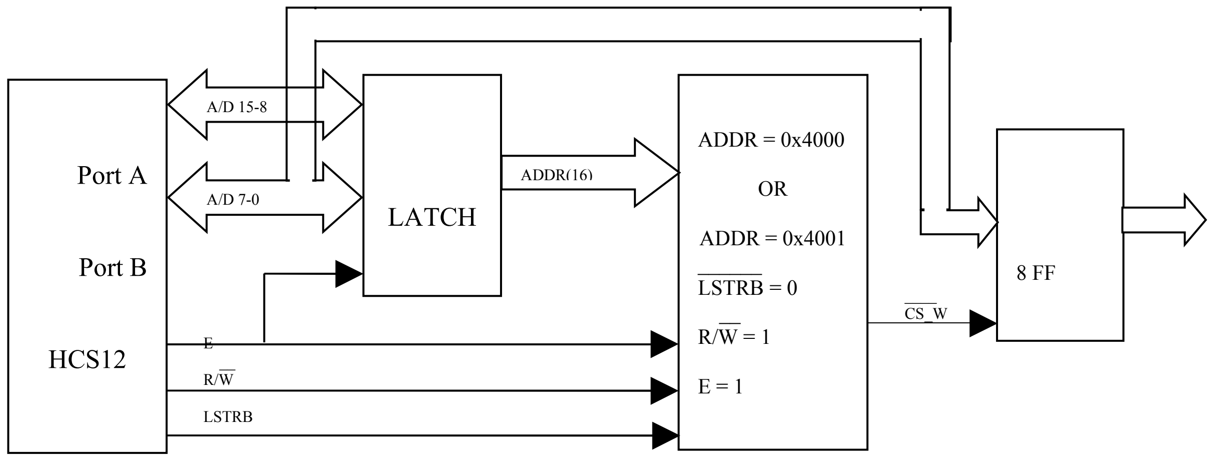
(b) Timing diagram

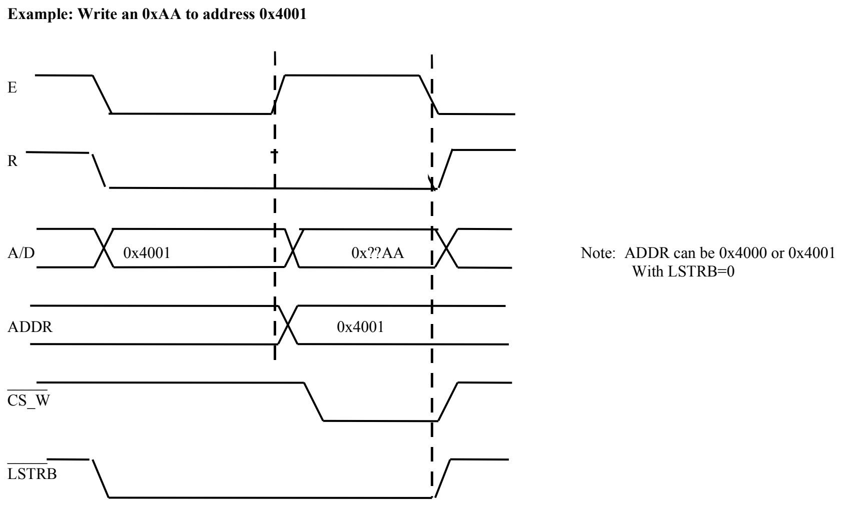


(c) Graphical symbol

Master-slave D flip-flop

• 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 A for output, and we write a number to Port A

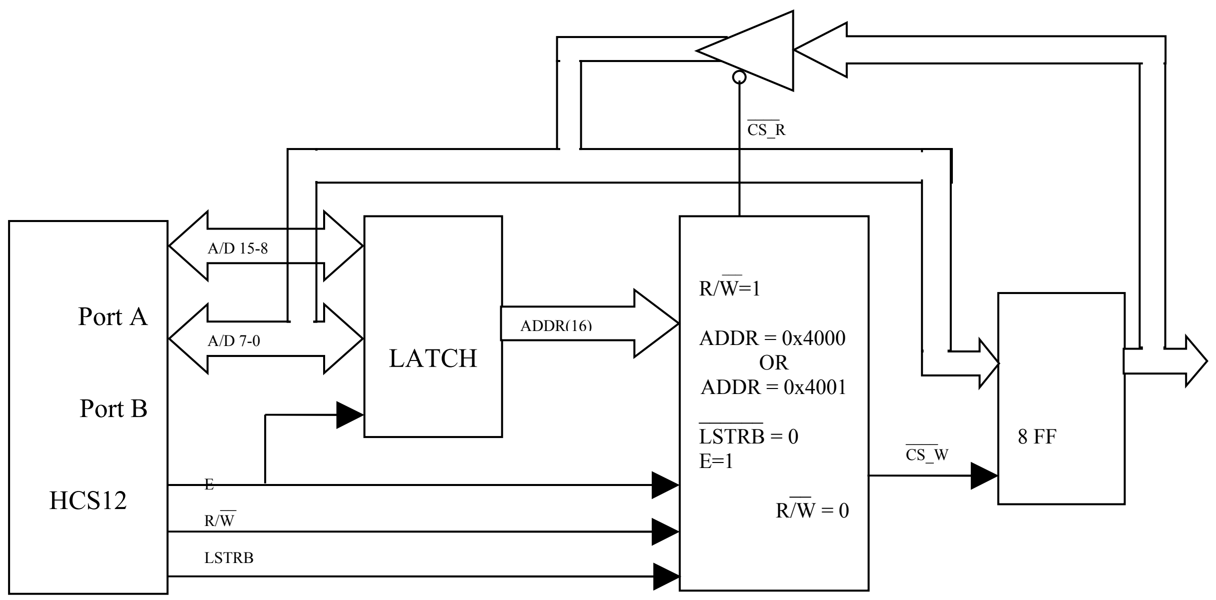
• When we read from Port A, we will read back the number we wrote

• This is a useful diagnostic

• We can make our output port have this same 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 HCS12 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

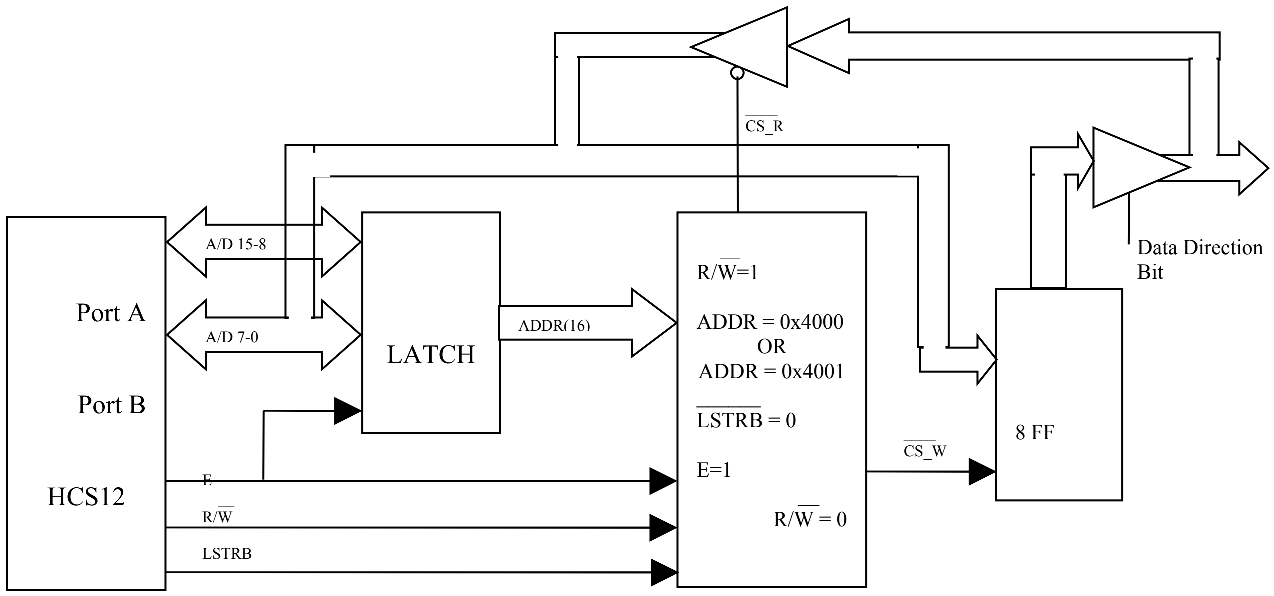
• To do this we need a data direction bit (at another address), and a tri-state buffer on the outputs of the flip-flops

• The data direction bit is simply a flip-flop which is set or cleared by the 9S12

• 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 set for 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 0x4001. The direction of the port is determined by a data direction bit at address 0x4002:



The data direction bit is the output of a FF which was written to at another address of the HCS12
For example, it could be Bit 4 of address 0x4002

Writing 0 to Bit 4 of address 0x4002 disables the output tri-state buffer
When we write to address 0x4001 we will latch the data into the flip-flops
This data will not be drive onto the external pins
When we read from address 0x4001, we will read what an external device drives onto the pins

Writing a 1 to Bit 4 of address 0x4002 enables the output tri-state buffer
When we write to address 0x4001 we will latch the data into the flip-flops
This data will be driven onto the external pins
When we read from address 0x4001 we will read the data latched into the flip-flops

**Review for Exam 3**

**A/D Converter**

• Power-up A/D converter (ATD1CTL2)
• Write 0x05 to ATD1CTL4 to set at fastest conversion speed and 10-bit conversions
• Write 0x85 to ATD1CTL4 to set at fastest conversion speed and 8-bit conversions
• Select number of conversions in a sequence (ATD1CTL3)
• Select type of conversion sequence and the analog channels sampled (ATD1CTL5)
– Right/left justified
– signed/unsigned
– Continuous Scan vs. Single Scan
– Multichannel vs. Single Channel conversions
• How to tell when conversion is complete - ATD1STAT0 register
• How to read results of A/D conversions - ATD1DR[7 − 0] (8-bit left justified conversions)
• How to read results of A/D conversions - ATD1DR[7 − 0] (8-bit right justified conversions)
• How to read results of A/D conversions - ATD1DR[15 − 6] (10-bit left justified conversions)
• How to read results of A/D conversions - ATD1DR[9 − 0] (10-bit right justified conversions)
– Be able to convert from digital number to voltage, and from voltage to digital number (need to know VRH and VRL).
• How long does it take to make a conversion?

**SPI**

• Pins used – SCLK, MOSI, MISO, SS
• Difference of use in Master and Slave mode
• SPI0CR1 Register
– Enable SPI
– Master or Slave
– Enable interrupts
– Clock polarity
– Clock phase
– Automatically operate SS for single-byte transfers
– LSB or MSB first
• SPI0CR2 Register
– Get into bidirectional mode
– Enable or disable the output buffer on the data pin in bidirectional mode
• SPI0BR Register — Set speed (master only)
• SPI0SR Register — SPIF and SPTEF flag - clear SPIF flag by reading SPI0SR, then read SPI0DR; clear the SPTEF flag by reading SPI0SR, then write the SPI0DR.
• SPI0DR Register — shift register – master starts transfer by writing data to SPI0DR

**Interfacing**

• Getting into expanded mode — MODA, MODB, MDOC pins or MODE Register
• PEAR Register — enable ECLK, LSTRB, R/W on external pins
• Ports A and B in expanded mode
– Port A – AD 15-8 (Port A is for data for high byte, even addresses)
– Port B – AD 7-0 (Port B is for data for low byte, odd addresses)
• E clock
– Address on AD 15-0 when E low, Data on AD 15-0 when E high
– Need to latch address on rising edge of E clock
– On write (output), external device latches data on signal initiated by falling edge of E
– On read (input), HCS12 latches data on falling edge of E
– E-clock stretch, turn/on- off EEPROM - MISC register
• R/W Line
• LSTRB line
• Single-byte and two-byte accesses
– 16-bit access of even address – A0 low, LSTRB low – accesses even and odd bytes
– 8-bit access of even address – A0 low, LSTRB high – accesses even byte only
– 8-bit access of odd address – A0 high, LSTRB low – accesses odd byte only
– A0 high and LSTRB high never occurs on external bus.