

Computer with N bit address bus can access  $2^N$  bytes of data

Computer with M bit data bus can access M bits of data in one memory cycle

Value on address bus tells memory which location computer wants to read (write)

Control lines tell memory when computer wants to read (write) data, and if access is read or write

## Address, Data and Control Buses

- A microprocessor system uses address, data and control buses to communicate with external memory and memory-mapped peripherals
- The address bus determines which memory location to access
- The control bus specifies whether the memory cycle is a read (into microprocessor) or a write (out of microprocessor) cycle, and specifies timing information for the cycle
- The data bus contains the data being transferred during the memory cycle
- For example, consider the following simple MC9S12 program, which continuously increments the contents of address 0x0400:

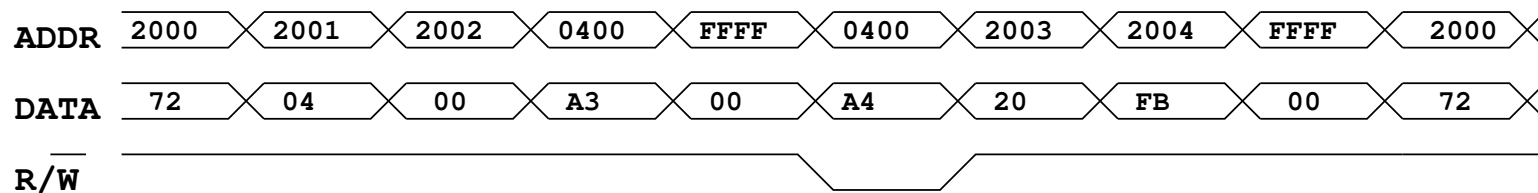
```
        org    0x2000  
  
loop:   inc    0x0400  
        bra   loop
```

- The program is stored in memory starting at memory location 0x2000
- The MC9S12 Program Counter starts at address 0x2000
- The MC9S12 reads the first instruction, `inc 0x0400`, located in address 0x2000 through 0x2002
- The MC9S12 then reads the contents of memory location 0x0400, takes an internal memory cycle to increment the value, then writes the new value out to address 0x0400
- The MC9S12 then reads the next instruction, `bra 0x2000`
- The MC9S12 takes one memory cycle to load the program counter with the new value of 0x2000, and to clear its internal pipeline, then reads the instruction at 0x2000 to figure out what to do next

### The MC9S12 address, data and control buses (simplified)

- Note: The following diagram assumes that the MC9S12 accesses one byte at a time
- The MC9S12 actually accesses two bytes (16 bits) at a time, when it can
- What actually occurs on the MC9S12 bus is a little more complicated than what is shown below

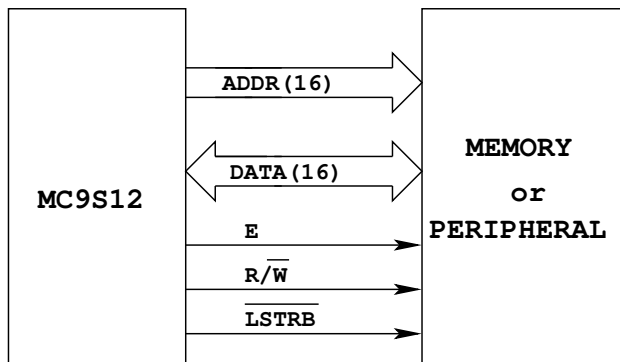
### MC9S12 ADDRESS, DATA AND CONTROL BUS (SIMPLIFIED)



```

                .org 0x2000          2000: 72
loop:          inc 0x0400           2001: 04 } inc 0x0400
                bra loop            2002: 00 }
                bra loop            2003: 20 } bra 0x2000
                bra loop            2004: FB }

```



MC9S12 has 16 bit address bus - can access 65536 bytes

1024 bytes = 1 kB

65536 bytes = 64 kB

MC9S12 has 16 bit data bus - can access 16 bits (2 bytes)  
at a time

For example, the instruction `LDX $0900`  
will read the two bytes at address \$0900 and \$0901

Sometimes MC9S12 only accesses one byte -- e.g., `LDAA $0900`  
The MC9S12 accesses only the byte at address \$0900

$\overline{R/W}$  tells memory (or peripheral) if MC9S12 is reading or writing

$\overline{R/W}$  high => read

$\overline{R/W}$  low => write

E tells memory when MC9S12 is reading (writing) --  
synchronizes data accesses

$\overline{LSTRB}$  (together with  $\overline{ADDR0}$ ) tells memory if MC9S12 is accessing one or two bytes

### The MC9S12 Memory Map

- The MC9S12 has address regions occupied by internal memory and peripherals
- A diagram showing which address regions are used is called a memory map
- Here is a memory map of the MC9S12DP256 with no added memory or peripherals

0x0000	Registers	1 KB
0x03FF		
0x0400	EEPROM	3 KB
0x0FFF		
0x1000	User RAM	11 KB
0x3BFF		
0x3C00	D-Bug 12 RAM	1 KB
0x3FFF		
0x4000	Flash EEPROM	16 KB
0x7FFF		
0x8000	Banked Flash EEPROM	16 KB
0xBFFF		
0xC000	D-Bug 12 Flash EEPROM	16 KB
0xFFFF		

### The Expanded MC9S12 Memory Map

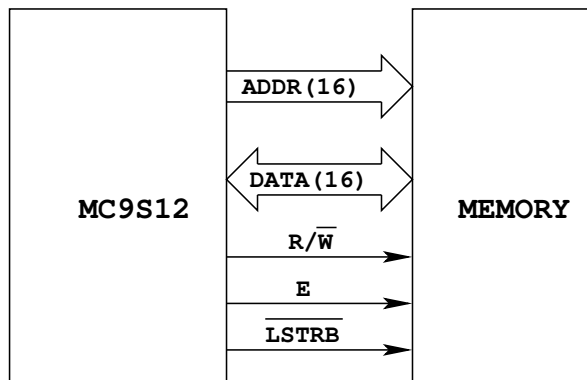
- We will add external peripherals to the MC9S12
- First, we will disable the Flash EEPROM at address 0x4000 through 0x7FFF (which we are not using anyway)
- Here is a memory map of the MC9S12DP256 with the peripherals we will add
- The peripherals will be put at 0x4000 and 0x4001

0x0000	<b>Registers</b>	<b>1 KB</b>
0x03FF		
0x0400	<b>EEPROM</b>	<b>3 KB</b>
0x0FFF		
0x1000	<b>User RAM</b>	<b>11 KB</b>
0x3BFF		
0x3C00	<b>D-Bug 12 RAM</b>	<b>1 KB</b>
0x3FFF		
0x4000	<b>Unused Space</b>	<b>Use address 0x4000 – 0x4001 for external peripherals</b>
0x7FFF		
0x8000	<b>Banked Flash EEPROM</b>	<b>16 KB</b>
0xBFFF		
0xC000	<b>D-Bug 12 Flash EEPROM</b>	<b>16 KB</b>
0xFFFF		



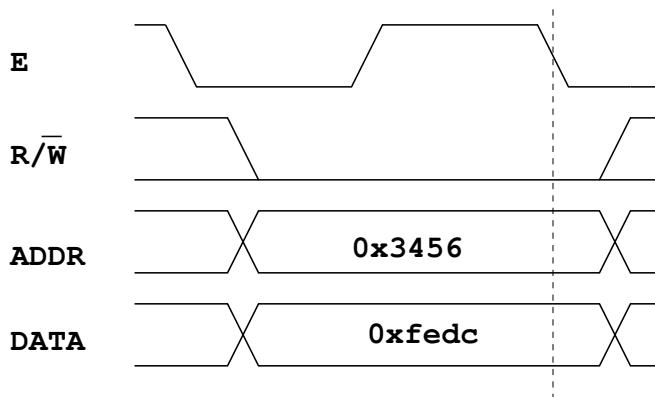
### Simplified MC9S12 Write Cycle

- When the MC9S12 writes data to memory it does the following:
  - It puts the address it wants to write to on the address bus (when E-clock goes low)
  - It puts the data it wants to write onto the data bus
  - It brings the Read/Write ( $R/\bar{W}$ ) line low to indicate a write
  - The MC9S12 expects the external device at the given address will latch the data into its registers data on the falling edge of the E-clock



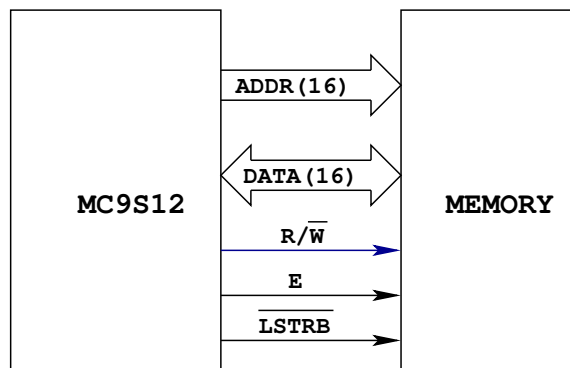
**WRITE:** MC9S12 puts address on address bus  
 puts data on data bus  
 brings  $R/\bar{W}$  low  
 Memory latches data on falling edge of E clock

**Example:** Write 0xfedc to address 0x3456 & 3457



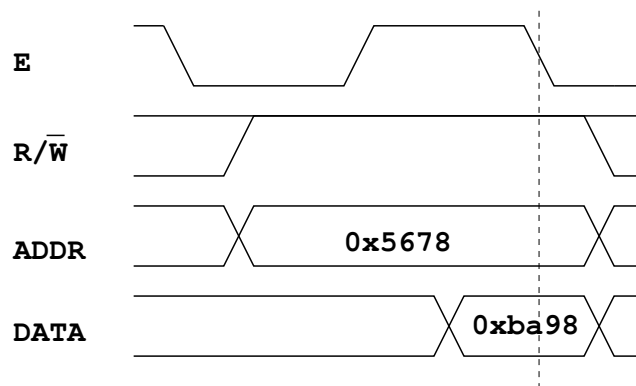
### Simplified MC9S12 Read Cycle

- When the MC9S12 reads data from memory it does the following:
  - It puts the address it wants to read from on the address bus (when E-clock goes low)
  - It brings the Read/Write ( $R/\overline{W}$ ) line high to indicate a read
  - The MC9S12 expects the external device at the given address will put data on the data bus
  - On the falling edge of the E-clock, the MC9S12 latches the data into its internal register



**READ:** MC9S12 puts address on address bus  
 brings  $R/\overline{W}$  high  
 Memory puts data on data bus  
 MC9S12 latches data on falling edge of E clock

**Example:** Read from address 0x5678 & 0x5679



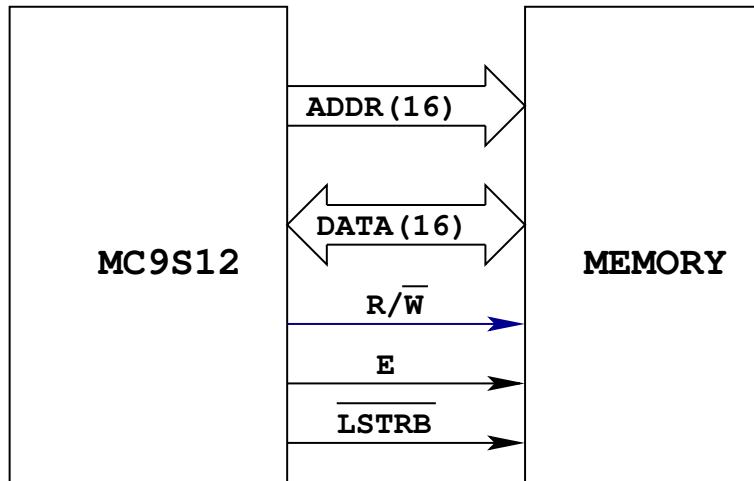
### The Real MC9S12DP256 Bus

- Up to now we have been using the MC9S12 in Single Chip Mode
  - In Single Chip Mode the MC9S12 does not have an external address/data bus
- The MC9S12 can be run in Expanded Mode
  - In Expanded Mode the MC9S12 does have an external address/data bus
- Things are a little more complicated on the real MC9S12DP256 bus than shown in the simplified diagrams above
- The MC9S12DP256 has a multiplexed address/data bus
- The MC9S12DP256 sometimes accesses a single byte on a memory cycle, and it sometimes access two bytes on a memory cycle

### The Multiplexed Address/Data Bus

- The MC9S12DP256 has a limited number of pins it can use
- To have full 16-bit address bus and a full 16-bit data bus the MC9S12DP256 would need to use 32 extra pins (in addition to several pins used for the control bus)
- To save pin count Motorola uses the same set of pins for several purposes
- When put into expanded mode, the MC9S12 uses the pins normally used for Ports A and B for its multiplexed address and data bus
  - When running in expanded mode you can no longer use Ports A and B as general purpose I/O lines
- The MC9S12 uses the same sixteen lines of Ports A and B for both address and data
- When the E-clock is low the sixteen lines AD15-0 are used for address
- When the E-clock is high the sixteen lines AD15-0 are used for data

## The Multiplexed Address/Data Bus



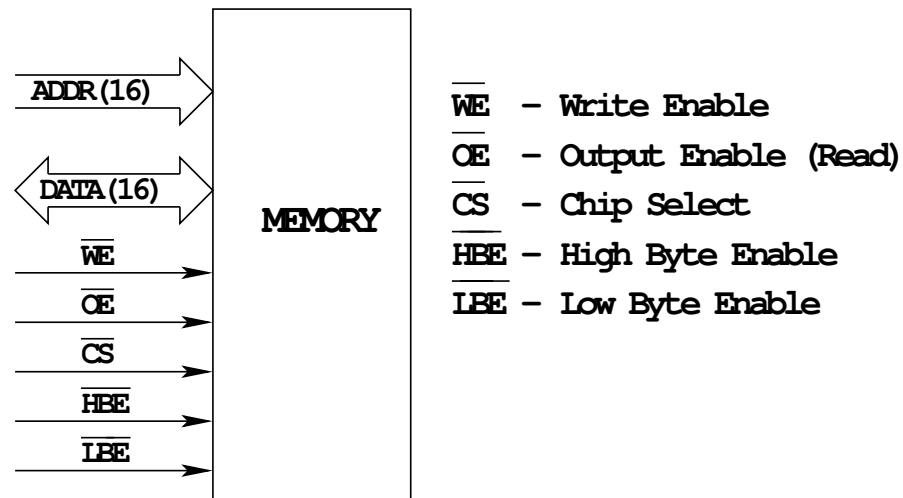
MC9S12 has 16-bit address and 16-bit data buses

Requires 35 bits

Not enough pins on MC9S12 to allocate 35 pins  
for buses and pins for all other functions

## Memory Chip Interface

- Memory chips need separate address and data bus
  - Need way to de-multiplex address and data lines from MC9S12
- Memory chips need different control lines than the MC9S12 supplies
- These control lines are:
  - Chip Select – goes low when the MC9S12 is accessing memory chip
  - Write Enable – goes low when the MC9S12 is writing to memory
  - Output Enable – goes low when the MC9S12 is reading from memory
  - High Byte Enable – goes low when the MC9S12 is accessing the High Byte (Odd Address) of memory
  - Low Byte Enable – goes low when the MC9S12 is accessing the Low Byte (Even Address) of memory

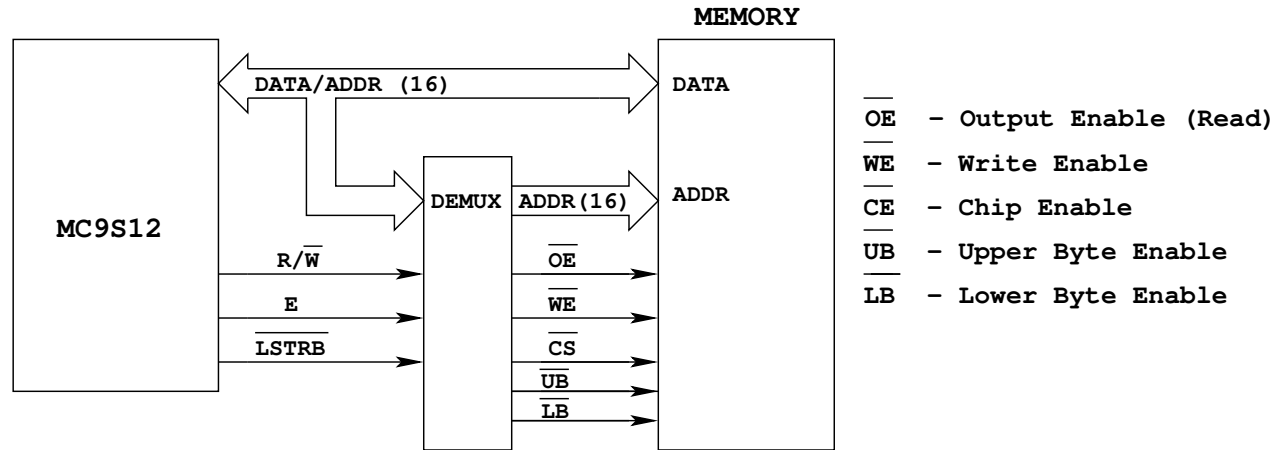


**Memory needs separate address and data busses**

**Need way to separate address and data**

## The Multiplexed Address/Data Bus

- To talk to memory chip we will need to build a demultiplexer between the MC9S12 and the memory chip



MC9S12 has 16-bit address and 16-bit data buses

Requires 35 pins

Not enough pins on MC9S12 to allocate 35 pins  
for buses and pins for all other functions

Solution: multiplex address and data buses

MC9S12 uses Ports A and B as multiplexed address/data bus

In expanded mode, you can no longer use Ports A and B for I/O

16-bit Bus: While E low, bus supplies address (from MC9S12)  
While E high, bus supplies data (from MC9S12 on write,  
from memory on read)