MTRN3500
Computing Applications in Mechatronics Systems

Week 2: General Interfacing – Hardware Descriptions

T2 - 2019

How could we send a signal outside the computer?

- We have to find the physical wires that do this.
- We have to find the connections and then make the connections.
- We have to find how to make these connections under software control.

What are the main sets of wires?

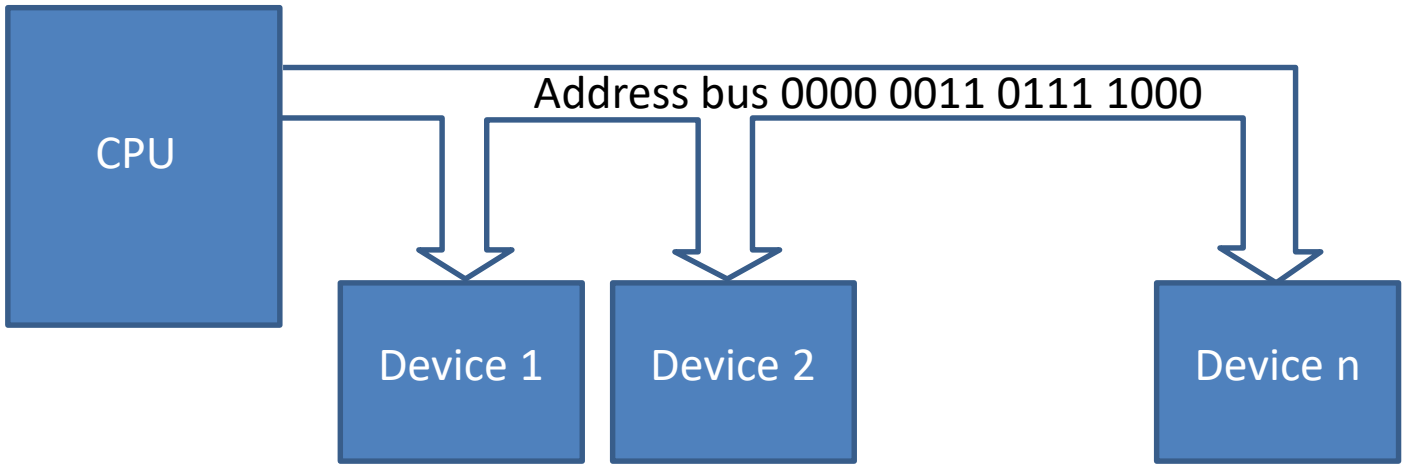
- Address bus (bus is a set of wires generally parallelly laid). Microprocessor sets a signal pattern on the address bus as required by the instructions that are being executed. (e.g. 0000 0101 1101 0011 -> 16 bit address)
- Data bus. Data bus transmits data to and from various devices including the CPU.
- Control bus. The purpose is to carry out various control actions and report status.

Where to they originate and where to they terminate?

- Address bus – originates at the micro processor or the CPU and then connects to every device on board (e.g. memory, storage, ports, other peripheral devices) (uni-directional)
- Data bus – Connects the CPU to every device on board. (bi-directional)
- Control bus – Connects the CPU to many of the devices on board. (bi- directional)

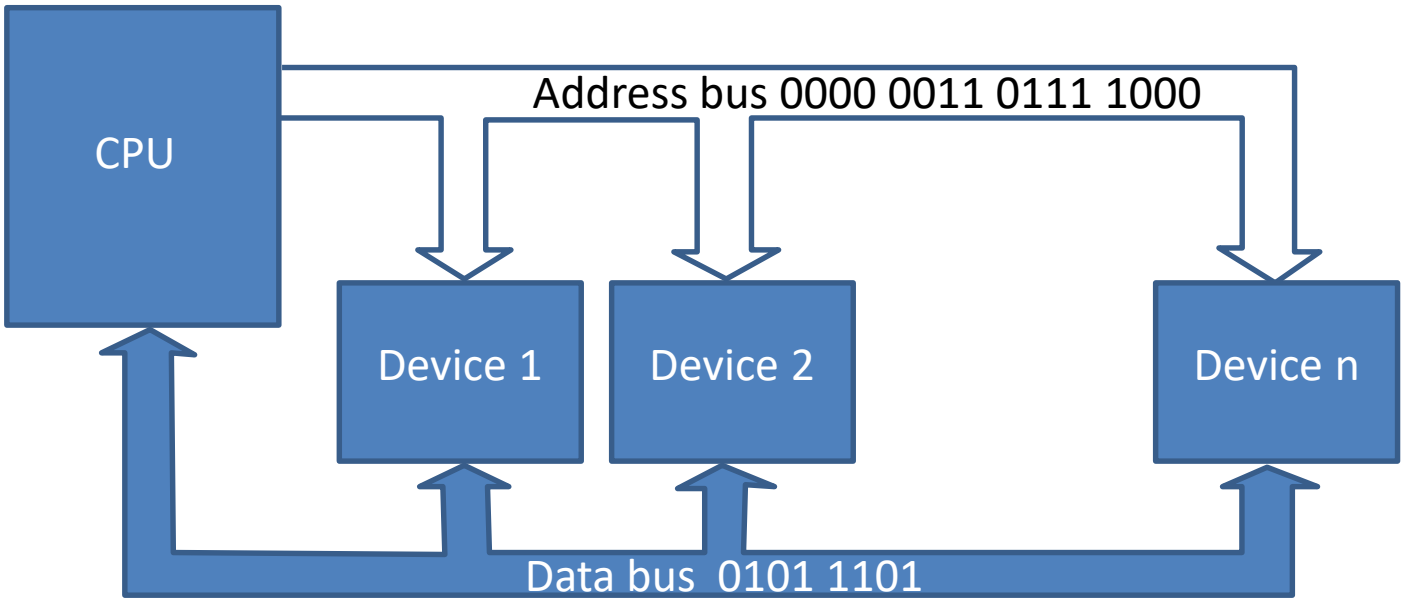
How does address bus work?

- Consider 16 – bit address bus: $A_{15}A_{14}A_{13}A_{12}…A_2A_1A_0$
- We can use certain instructions to set a bit pattern that would represent an address pattern, for example 0000 0011 0111 1000 = 0x0378
- Each address pattern refers to a unique location in the computer's memory space and in that location is a SINGLE BYTE of data.



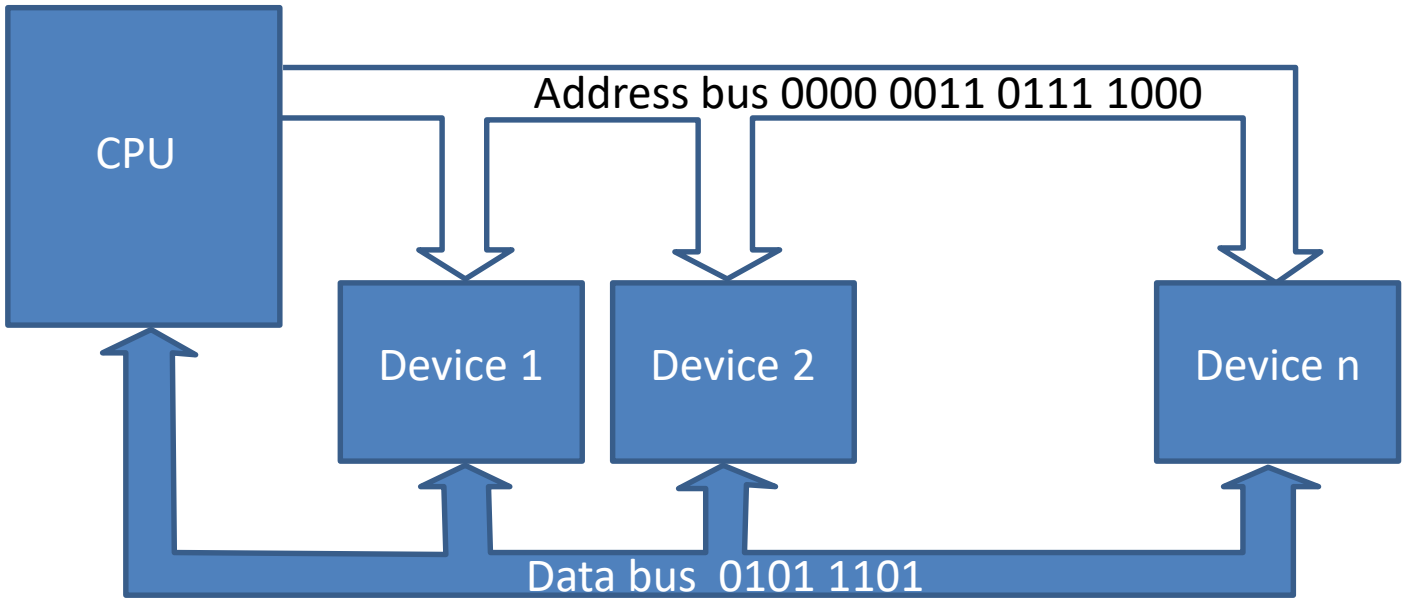
How does address bus work?

- Consider 16 – bit address bus: $A_{15}A_{14}A_{13}A_{12}…A_2A_1A_0$
- We can use certain instructions to set a bit pattern that would represent an address pattern, for example 0000 0011 0111 1000 = 0x0378
- Each address pattern refers to a unique location in the computer's memory space and in that location is a SINGLE BYTE of data.



How does address bus work?

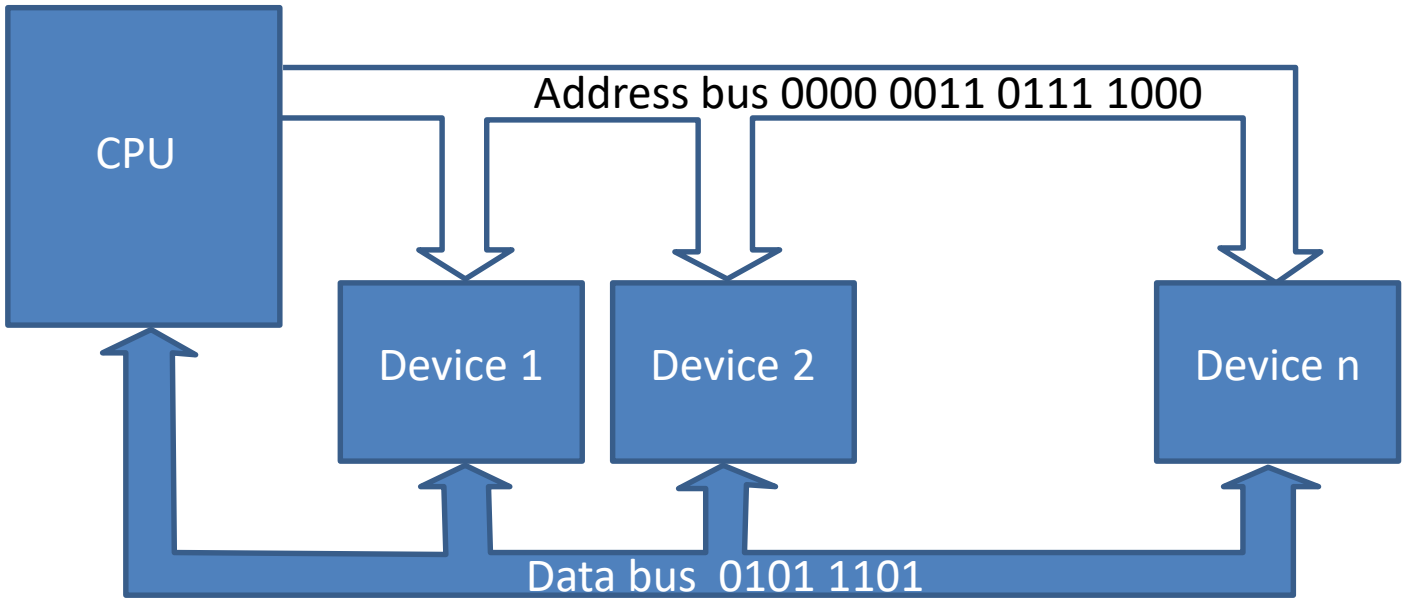
- Consider 16 – bit address bus: $A_{15}A_{14}A_{13}A_{12}…A_2A_1A_0$
- We can use certain instructions to set a bit pattern that would represent an address pattern, for example 0000 0011 0111 1000 = 0x0378
- Each address pattern refers to a unique location in the computer's memory space and in that space is a SINGLE BYTE of data.



Q1: Which device gave us the data?

How does address bus work?

- Consider 16 – bit address bus: $A_{15}A_{14}A_{13}A_{12}…A_2A_1A_0$
- We can use certain instructions to set a bit pattern that would represent an address pattern, for example 0000 0011 0111 1000 = 0x0378
- Each address pattern refers to a unique location in the computer's memory space and in that space is a SINGLE BYTE of data.



Q1: Which device gave us the data?

Q2: Did the CPU read the data or write the data?

Address Space

- The address space is generally occupied by devices as we saw and they could be the memory or the input and output devices (an example input output device is the serial port).
- Each device must be mapped to a unique area of the address space
- There are two ways to utilize address space to connect to input output devices.
  - *Memory Mapped IO*
  - *Isolated IO*

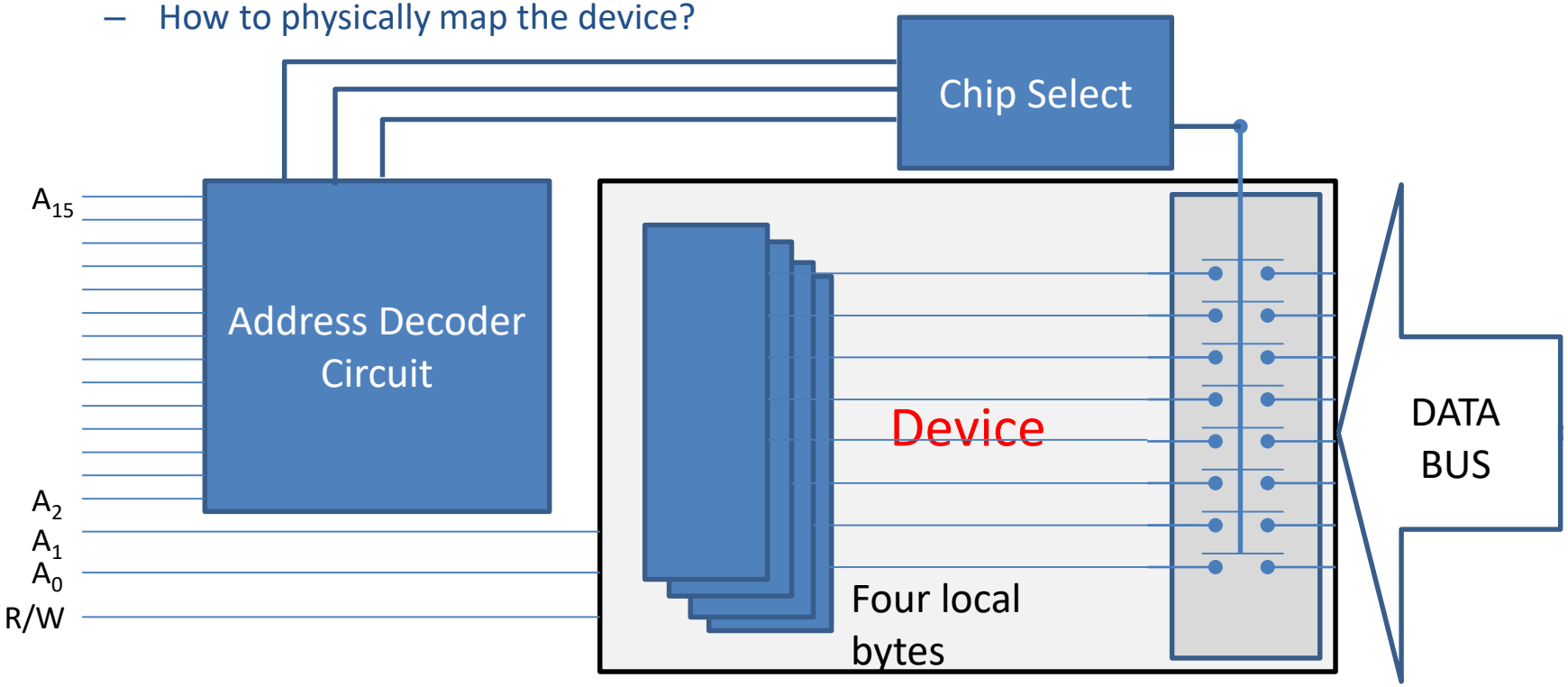
- In memory mapped IO the full address space is occupied by memory and IO devices. The address 0x378 is a single unique location in the address space.
- In isolated IO, there is one area for memory and another area for IO and these two are mutually exclusive. Therefore, the address 0x378 could be in memory or in the isolated IO. This ambiguity must be resolved in isolated IO.

How do we map a device?

- We need to know three things:
  - How many unique address locations the device require?
  - Which locations are free to map this device?
  - How to physically map the device?

How do we map a device?

- We need to know three things:
  - How many unique address locations the device require? 4 locations
  - Which locations are free to map this device? 0x210-0x21F, 0x300 – 0x30F
  - How to physically map the device?

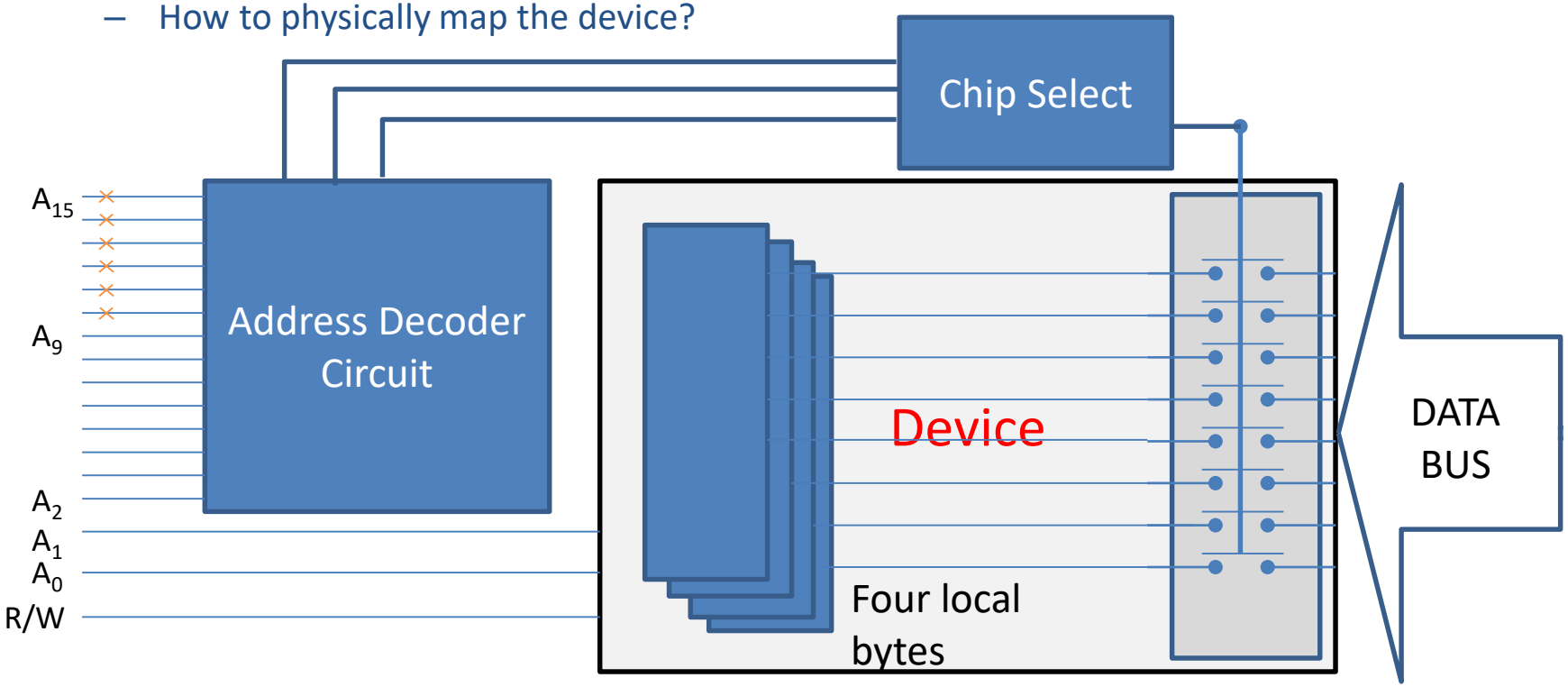


How do we map a device?

- We need to know three things:
  - How many unique address locations the device require? 4 locations
  - Which locations are free to map this device? 0x210-0x21F, 0x300 – 0x30F
  - How to physically map the device?
- Let us say we need to map this devices to addresses 0x300, 0x301, 0x302, 0x303
- What happens when we write to each of these address locations can be found in the documentation of the device.
- Let us arrange the addresses:
  - 0000 0011 0000 00 | 11 – BASE+3
  - 0000 0011 0000 00 | 10 – BASE+2
  - 0000 0011 0000 00 | 01 – BASE+1
  - 0000 0011 0000 00 | 00 – BASE ADDRESS
- You can see here that the higher 14 bits are constant and the address decoder circuit should be such that they should generate the chip select signals.

How do we map a device?

- We need to know three things:
  - How many unique address locations the device require? 4 locations
  - Which locations are free to map this device? 0x210-0x21F, 0x300 – 0x30F
  - How to physically map the device?



How to do this in software?

- The system you will use is a PC and it has isolated IO. Therefore, there are different ways to reach memory and IO.
- First you need to ask permission to read or write to a certain memory area. This is done using the following function:
  - `ioperm(lowest address, number of bytes, 1);// 0 = no error`
  - `ioperm(0x300, 4, 1);`
- To write a data byte of 0x55 to location 0x300, provided that the hardware allows to write to that location, you must use the following instruction.
  - `outb(0x55, 0x300);`
  - `// note that the way you write in the lab classes are slightly different.`
- To read location 0x303 into an unsigned char type data, you can use,
  - `unsigned char Data;`
  - `Data = inb(0x303); // note that the way you read in the lab classes are slightly different`

Ghost Addresses?

- If the address lines A15-A10 are ignored, we can write anything to those bits and it should not matter.
- To write a data byte of 0x55 to location 0x300, provided that the hardware allows to write to that location, we could now write any of these.
  - `outb(0x55, 0x300);`
  - `outb(0x55, 0x700): //0x700 is a ghost address of 0x300`
  - `outb(0x55, 0xF00); //0xF00 is a ghost address of 0x300`
- To read location 0x303 into an unsigned char type data, you can use any of these.
  - `unsigned char Data;`
  - `Data = inb(0x303);`
  - `Data = inb(0x703);`
  - `Data = inb(0xF03);`
- **Hence the addresses 0x703, 0xF03 are ghost addresses of 0x300.**