**Basic I/O System Design**

Table of Contents

[Accessing I/O Devices 3](#_Toc100172319)

[Memory Mapped I/O 3](#_Toc100172320)

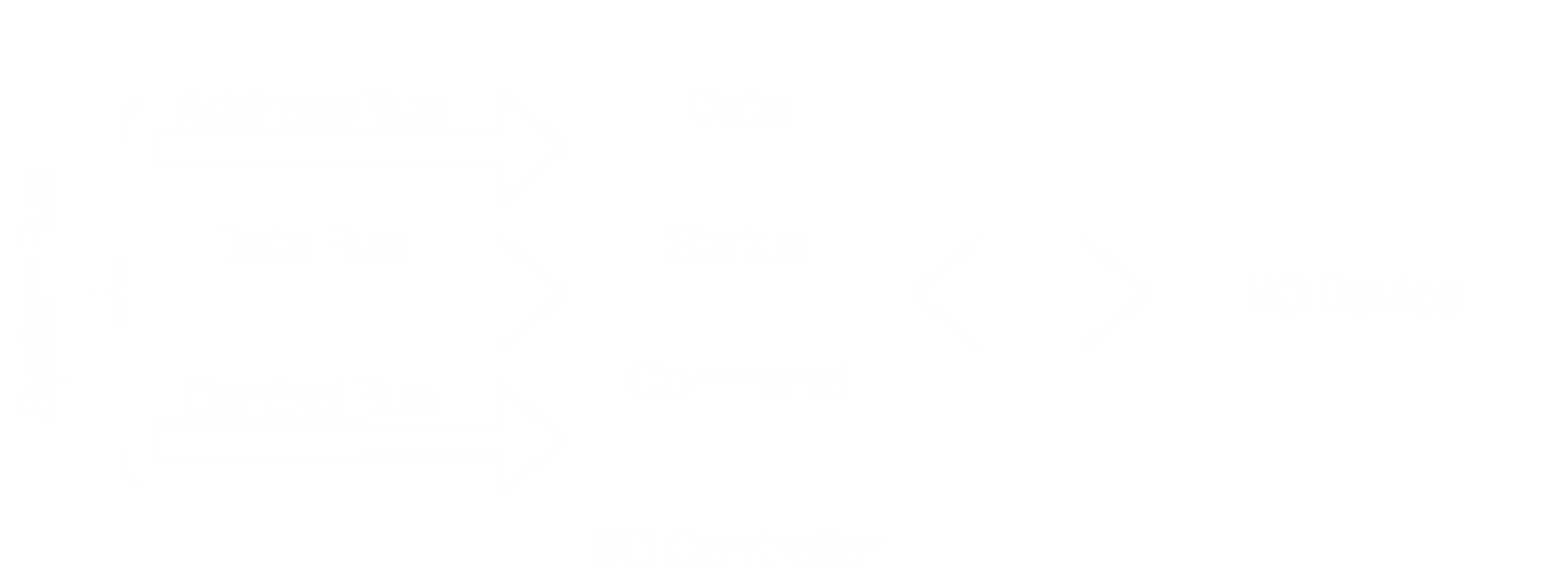
[Isolated I/O 4](#_Toc100172321)

As discussed before, there are basically three units in an I/O system, the MPU, the Interface Unit and the Peripherals Unit.

Each I/O device connects to a different **logical port**. In the Intel 8086 for example, there are logical ports. However, logical ports are just that, logical. There aren’t actually that many physical connections. In fact, there are just two, a READ pin and a WRITE pin.

This is where the **interfaces** come in. An interface connects one or more I/O device to the MPU’s address bus, data bus and control bus. For I/O devices specifically, we have the **I/O Controller Chip**. This chip has 3 parts:

1. The **data buffer** register, which stores data from the I/O device or data that must be sent to the I/O device.
2. The **status register**.
3. The **command register**, which holds the instruction from the MPU about what to do with the data in the data buffer.



## Accessing I/O Devices

So, how does the interface allow us to connect a large number of devices to a large number of logical ports without having that many physical ports.

There are two parts to accessing an I/O device:

1. Access Registers
2. Communication Protocols

The **Communication Protocols** we already know. There are three communication protocols, Programmed I/O (polling), Interrupt-Driven I/O and Direct Memory Access (DMA).

The **Access Registers** are basically memory locations. There are two ways in which memory locations can be accessed, Memory Mapped I/O and Isolated I/O. We touched on these briefly before, but will be looking into them in a bit more detail now.

### Memory Mapped I/O

In **Memory Mapped I/O**, we essentially reserve memory locations to be used by I/O devices. Thus, for logical ports, the memory locations from to are reserved.

For a specific memory location, say , the memory location is the logical port for a specific device connected to an interface. The data from that device will always be stored in that specific memory location. If the interface is connected to multiple devices, each device will have a separate memory location assigned to it.

The limitation here is not that we only have one byte per device. We can assign more if needed. However, regardless of how we handle it, we are still losing memory locations.

A huge advantage though, is that we can use the MOV operation instead of IN or OUT instructions. The MOV operation takes a memory address as a destination, so moving data to a memory location which is the logical port address for a device will cause the data to be sent to the data buffer of the interface.

Other advantages include:

* Less complication
* Less circuitry
* Less decoding
* No need to use special signals

### Isolated I/O

In **Isolated I/O**, we have a separate memory for logical port addresses. This allows the complete memory size of the original memory to be used by the MPU. This takes away the disadvantage of memory mapped I/O, but it also takes away the advantage of using MOV instructions instead of IN and OUT instructions. We also need separate signals.