Programmable Peripheral Interface

Table of Contents

[8155 PPI 3](#_Toc100240874)

[Command Register Format 4](#_Toc100240875)

[Handshaking Signals 6](#_Toc100240876)

[8255 PPI 8](#_Toc100240877)

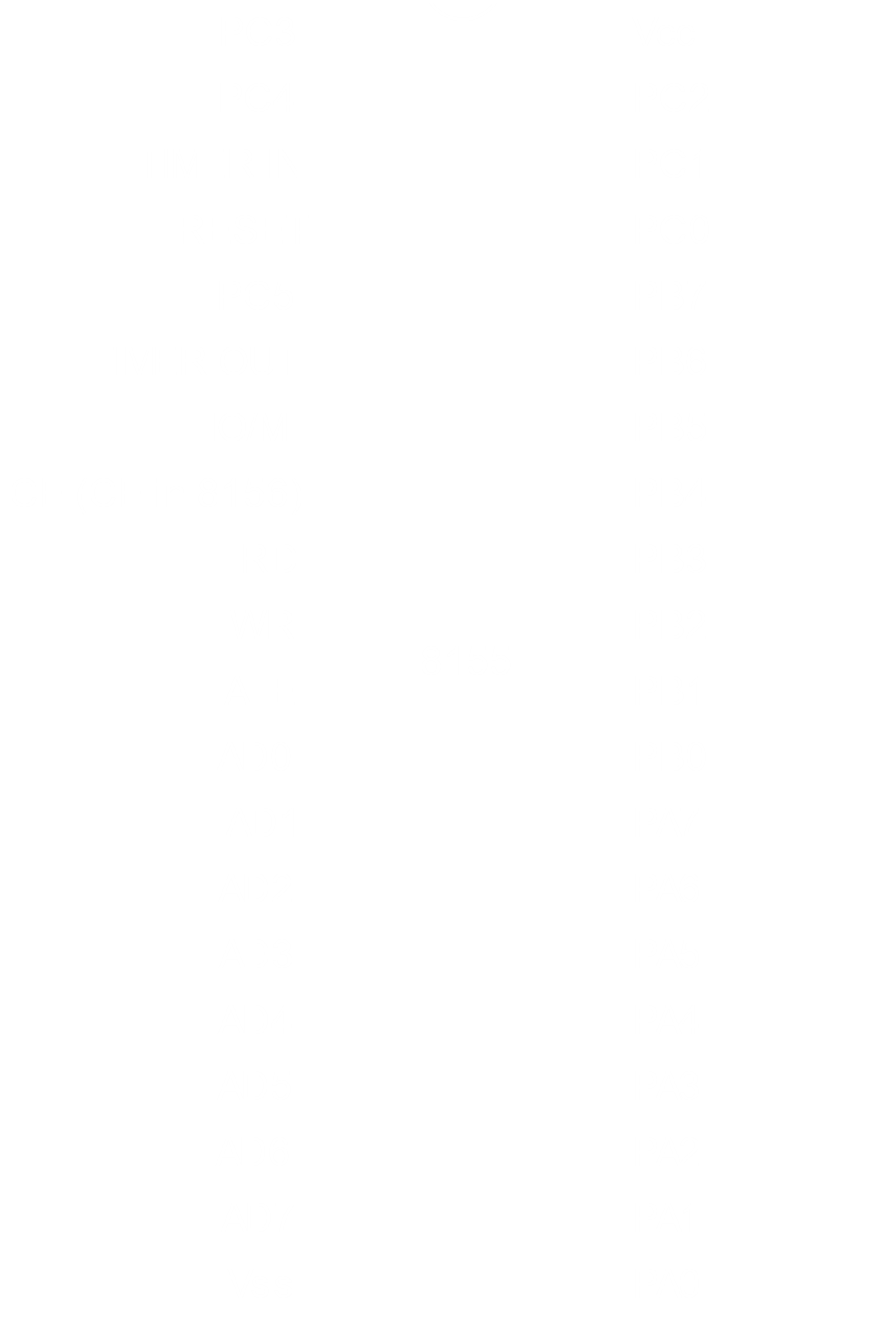
[Pins and Ports 8](#_Toc100240878)

[Programming Modes 10](#_Toc100240879)

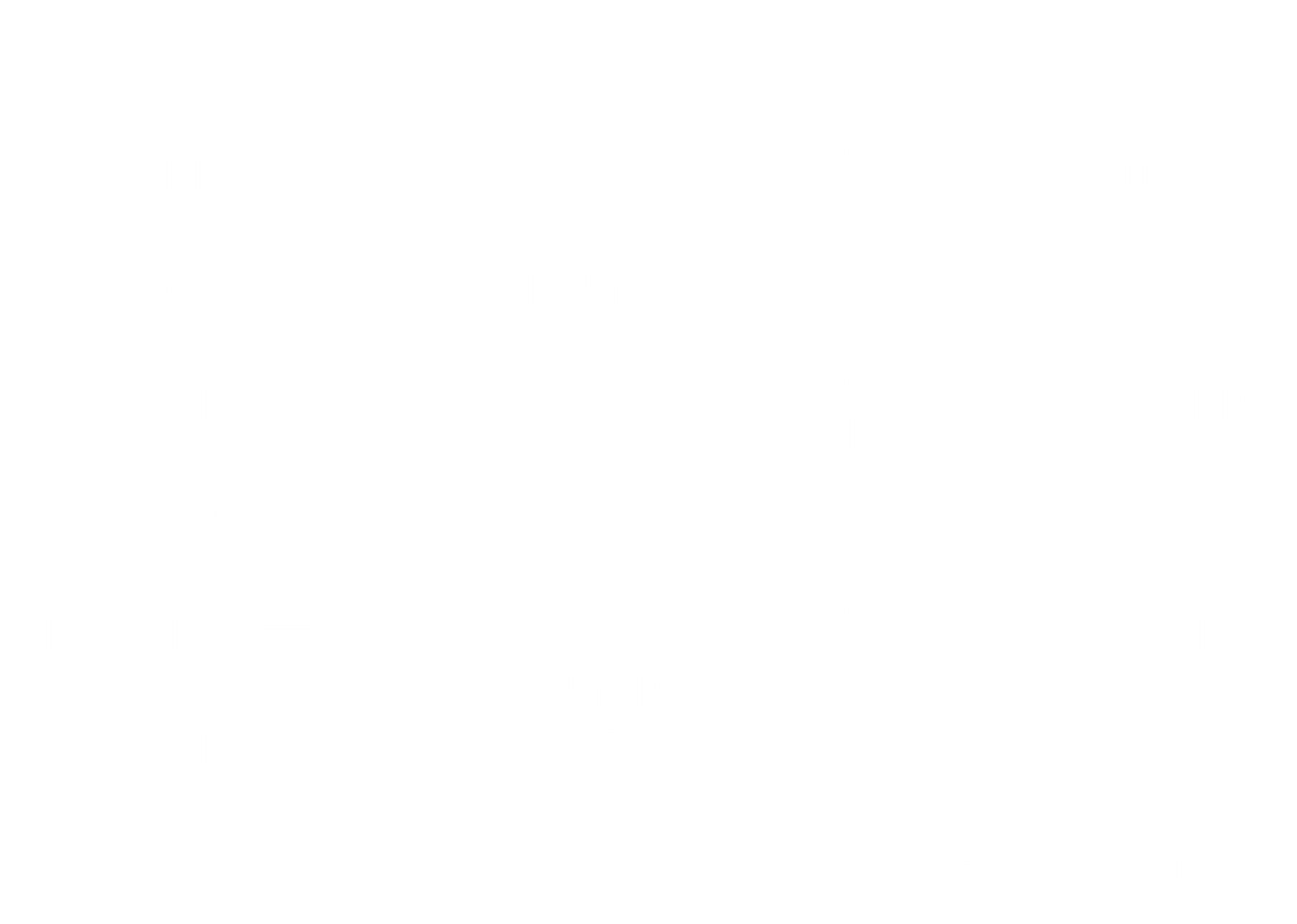
The interface between the microprocessor and an I/O device is called an **I/O controller**. I/O controllers are of two types, internal and external. **Internal I/O Controllers** are built into the motherboard and can be seen if we inspect the motherboard closely. **External I/O Controllers** use physical ports that we can see and connect peripherals to, such as the USB ports that we use.

We will be examining two different I/O controllers in this chapter, the Intel 8155 and the Intel 8255. I/O controllers can be of several types depending on their functionality. For example, normal I/O operations are handled by a **Programmable Peripheral Interface** (PPI), interrupts are handled by a Programmable Interrupt Controller (PIC), etc. Both the Intel 8155 and the Intel 8255 are PPIs.

## 8155 PPI



The Intel 8155 PPI was designed to be compatible with the **8085 Microprocessor**. If we examine the pin diagram above, we will see that it has 8 address/data line pins AD0 to AD7 along with an ALE pin, since that is how the address and data lines in the 8085 work. It also has three ports, Port A and Port B, which are both programmable 8-bit I/O ports, and port C, which is a programmable 6-bit I/O port. In addition to this, it also has a single programmable 14-bit binary counter/timer.



If we next examine the block diagram provided above, we will also see that the 8155 PPI has a RAM, which is of 256 x 8 bytes.

### Command Register Format

All I/O controllers have a **command register**, which is used by the microprocessor to specify how the I/O controller should behave. For the 8155 PPI, the command register is of 8 bits, D0 to D7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Timer  Command | | IEB | IEA | PC | | PB | PA |

* The bits D0 and D1 are used by ports A and B respectively. A 0 bit defines that the port should be used to input data while a 1 bit defines that the port should be used to output data. For example, if we have a keyboard connected to port A and a 7-segment display connected to port B, then the values of D1 and D0 would be 1 and 0 respectively.
* The bits D2 and D3 are used by port C. Port C is used for handshaking signals, which we will be examine separately a little later.
* The bits D4 and D5 are used as interrupt enable pins for ports A and B respectively. A 0 bit defines that interrupts are disabled while a 1 bit defines that interrupts are enabled.
* The bits D6 and D7 are used by the timer. We will be studying timers in a different lecture, where we will examine how these pins work.

Below, we have some program code for the example mentioned above (with a single keyboard and a single 7-segment display). Note that the **port addresses** used by the control register, port A and port B are 20h, 21h and 22h respectively.

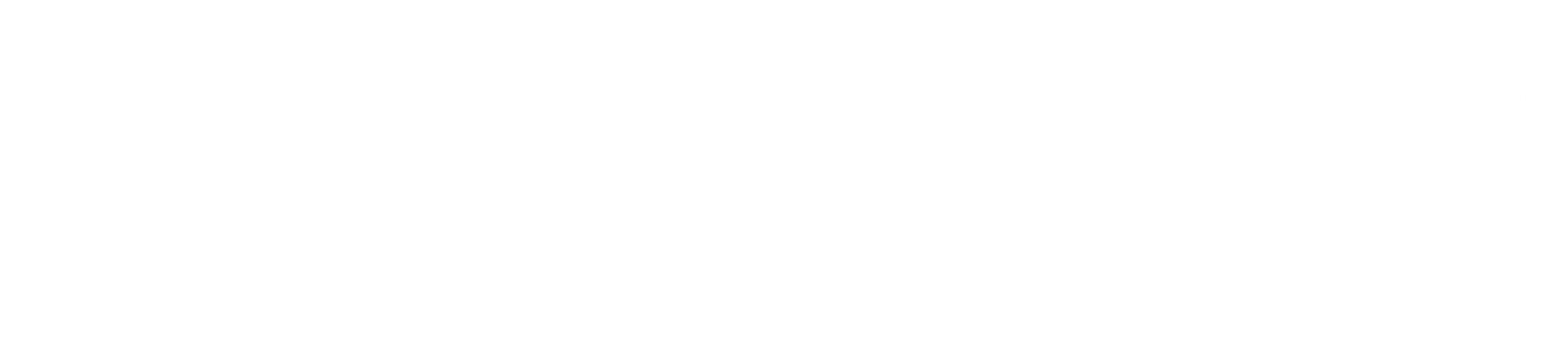
MVI A, 02 ; move 00000010 to the accumulator  
OUT 20h ; set value of control register to 00000010  
  
IN 21h ; get input from Port A  
MOV A, [0021h] ; move Port A data to accumulator  
  
; instructions to convert keyboard input to 7-segment display output  
  
OUT 22h ; output data from accumulator to port B

ASSEMBLY

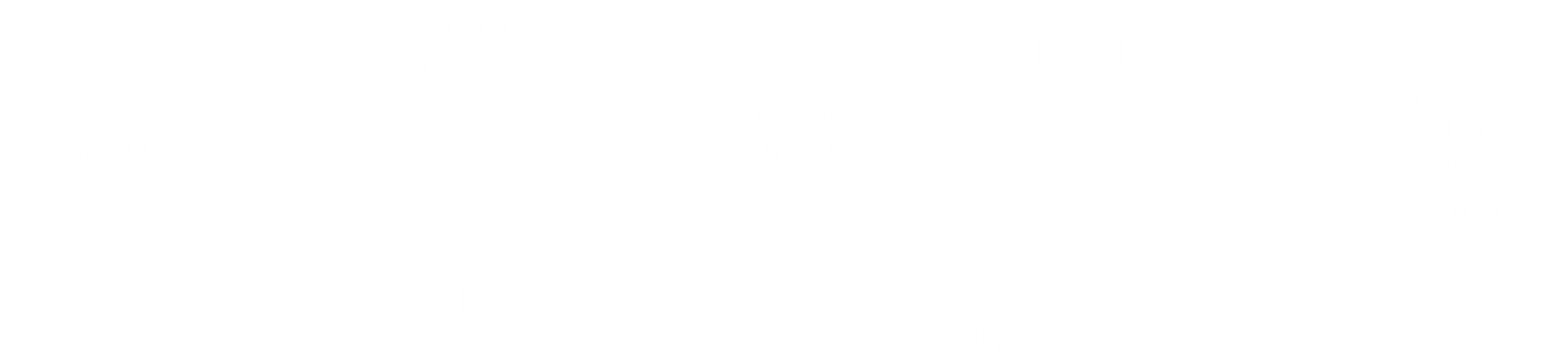
### Handshaking Signals

**Handshaking signals** can be divided into two categories, input handshaking signals and output handshaking signals.

There are two **input handshaking signals**, Strobe (STB) and Input Buffer Full (IBF). When a peripheral device has input data to send, it places the data on the data line and sends an STB signal to the 8155 PPI. If the input buffer of the 8155 PPI is full, it sends an IBF signal to the peripheral device, instructing it to not send the next byte. Next, it sends an INTR signal to the microprocessor, essentially asking the microprocessor to read the data in its input buffer.



There are also two **output handshaking signals**, Output Buffer Full (OBF) and an ACK signal. When the microprocessor sends data to the 8155 PPI to forward to a peripheral device, the data is placed in the output buffer of the 8155 PPI. Once this buffer is full, the 8155 PPI sends an OBF signal to the peripheral device, asking it to read the data from the output buffer via the data lines. Once the data has been read, the peripheral device sends back an ACK signal, at which point the 8155 PPI sends an INTR signal to the microprocessor, basically asking it to send the next byte.



As mentioned before, **Port C** is used for handshaking signals using the bits D2 and D3. The details of this are provided in the table below:

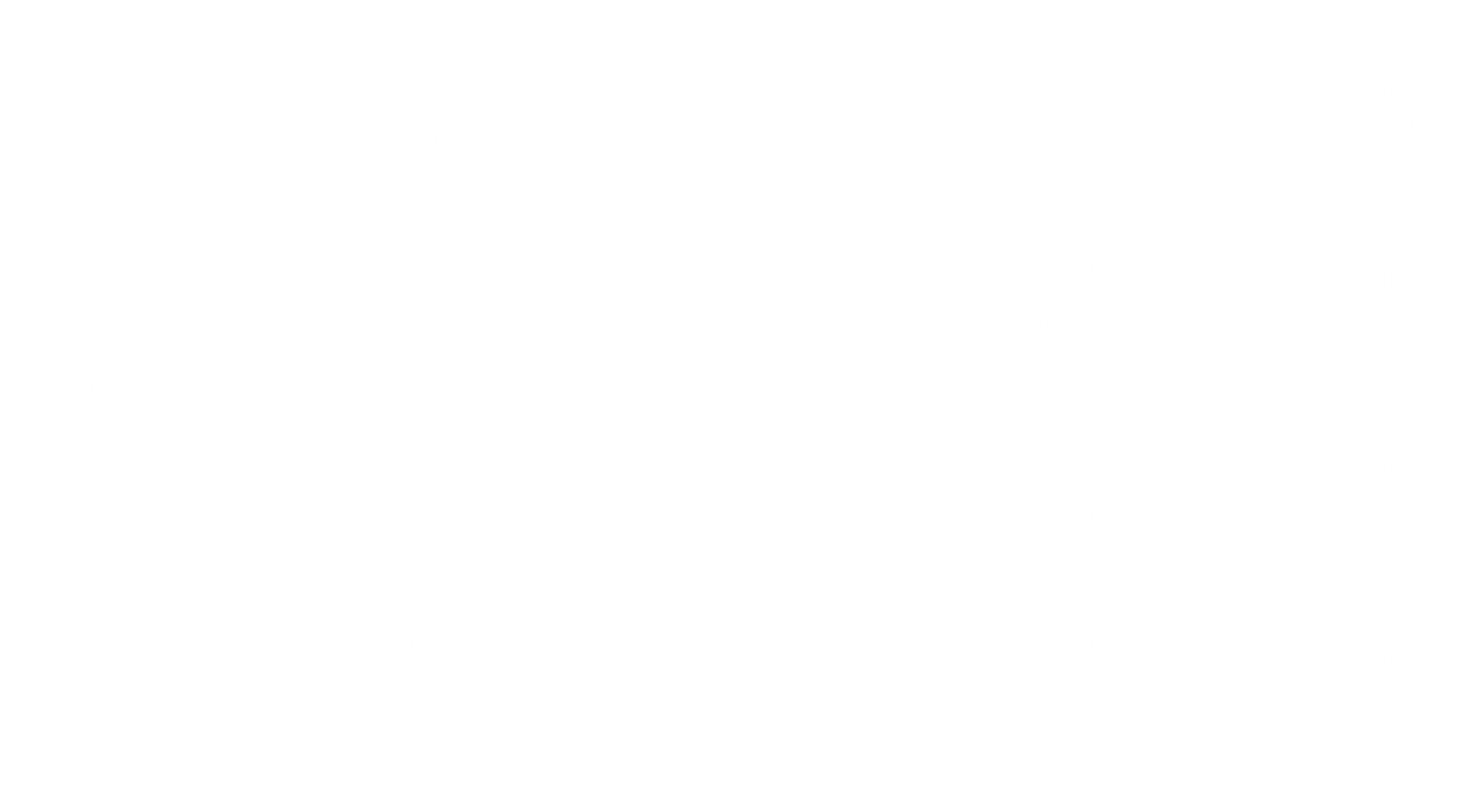
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **D3** | **D2** | **PC5** | **PC4** | **PC3** | **PC2** | **PC1** | **PC0** |
| 0 | 0 | IN | IN | IN | IN | IN | IN |
| 0 | 1 | OUT | OUT | OUT | OUT | OUT | OUT |
| 1 | 0 | OUT | OUT | OUT | STBA | BFA | INTRA |
| 1 | 1 | STBB | BFB | INTRB | STBA | BFA | INTRA |

* If D3 and D2 are set to 0 and 0 respectively, port C is being used as a **6-bit input port**.
* If D3 and D2 are set to 0 and 1 respectively, port C is being used as a **6-bit output port**.
* If D3 and D2 are set to 1 and 0 respectively, the higher 3 bits of port C are being used as a **3-bit output port**, while the lower 3 bits are being used for handshaking signals for port A in the order STB, IBF/OBF and INTR.
* If D3 and D2 are set to 1 and 1 respectively, the higher 3 bits of port C are being used for handshaking signals for port B in the order STB, IBF/OBF and INTR and the lower 3 bits of port C are being used for handshaking signals for port A, as before.

## 8255 PPI

The **8255 PPI** has several noticeable differences. Firstly, it does not have a **timer** built in and uses an external one if required. Secondly, it is more **structured**, as we will soon see. Finally, it has a **bi-directional port**.

### Pins and Ports



As can be seen in the **block diagram** above, the ports of the 8255 are divided into **two groups**, Group A and Group B. **Group A** consists of Port A, which uses PA0 to PA7, and the upper four bits of Port C, PC4 to PC7. **Group B** consist of Port B, which uses PB0 to PB7, and the lower four bits of Port C, PC0 to PC3. Additionally, there is a **command register**, also of 8 bits.

The ports have the following addresses:

* 60h for Port A
* 61h for Port B
* 62h for Port C
* 63h for the control register

Both ports A and B can act as 8-bit input or output ports, while port C is divided into 2 **nibbles** (4 bits), each of which can be used as a separate input or output port.

The 8255 works with the **8086 microprocessor**. It is possible to place **multiple 8255 PPIs** in the same system. A total of 8 pins are available to select a chip, from A0 to A7. The pins A2 to A7 are used to select a chip, meaning we can have up to separate 8255 PPIs in the same system. Once we have selected a chip, we can use the A0 and A1 pins to select one of the **ports** or the control register.

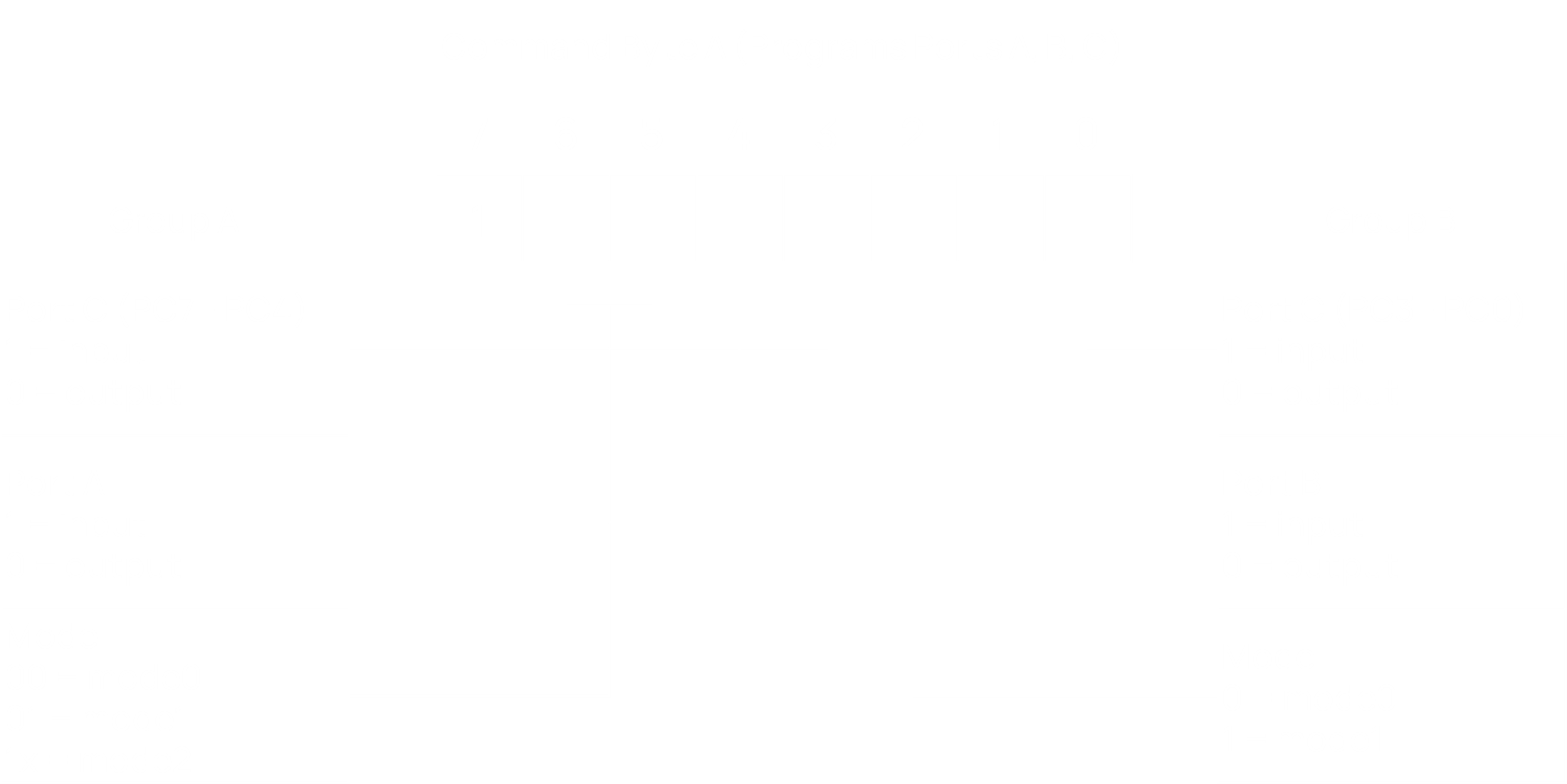
Notice that we can directly use these pins to select a chip and a port, without specifying an address. Thus, we are not using memory-mapped I/O, but rather **isolated I/O**. We have the choice of using either isolated I/O or memory mapped I/O using the IO/M pin.

### Programming Modes

The 8255 can work in one of three programming modes:

* **Mode 0** – In this mode, all three ports are being used for basic I/O operations.
* **Mode 1** – In this mode, ports A and B are being used for basic I/O operations, while port C is being used for handshaking signals. The pins of port C work groupwise, with pins PC4 to PC7 providing the handshaking signals for Port A and pins PC0 to PC3 providing the handshaking signals for Port B.
* **Mode 2** – In this mode, port A is being used for bi-directional I/O operations while port B is being used for uni-directional I/O operations. Port C is divided with pins PC3 to PC7 providing handshaking signals for Port A and PC0 to PC2 providing handshaking signals for port B. Notice that in this specific case only, PC3 does not maintain its group.

We specify which mode each port will be working in using a **command byte**, as shown below:



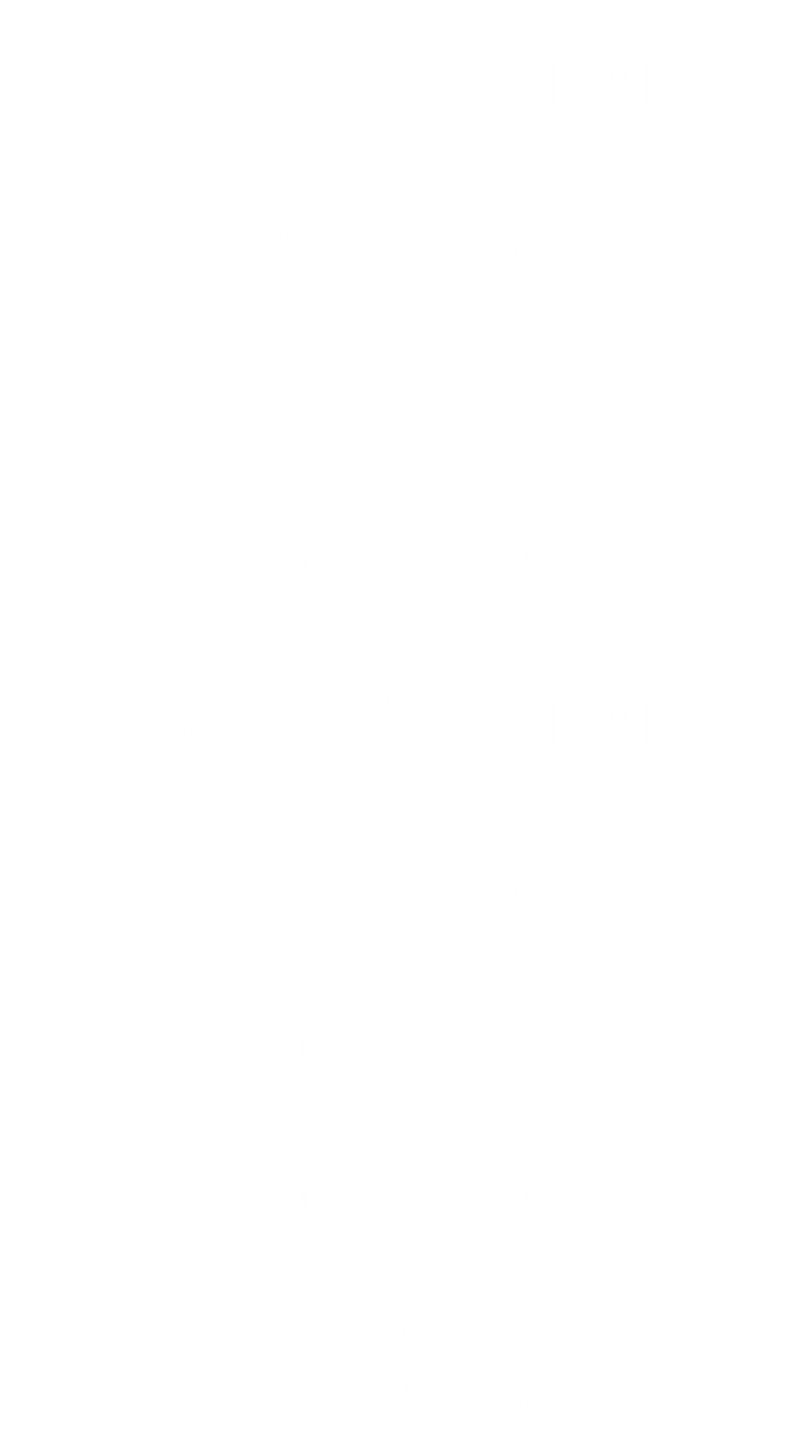
A few examples should make this easier to understand.

Example 1:

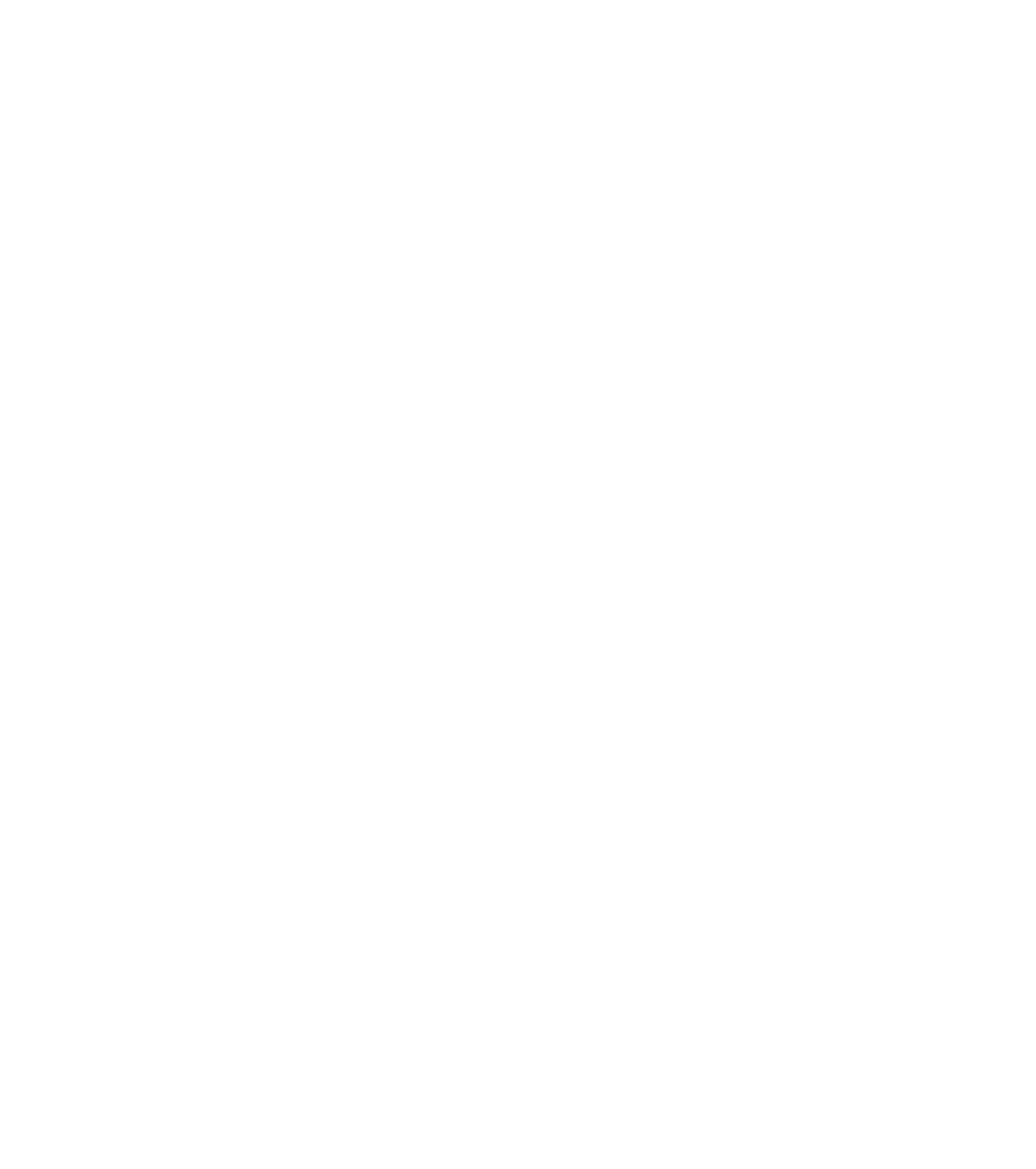
Suppose we want to work in **mode 1**, and both ports A and B are working as input ports. The command byte will be:

1 0 1 1 X 1 1 X

The first byte from the left is always 1. The second and third bytes are set to 0 and 1 respectively, since Port A will work in mode 1. The fourth byte is set to 1 since Port A is acting as in input port. Since we are in mode 1, port C is working with handshaking signals. Thus, the fifth and eight bits don’t actually matter. The sixth and seventh bits are both set to 1 since port B is working in mode 1 and as an input port.



A **timing diagram** for this example is provided below. This is a generic timing diagram that can be assumed to be for either port A or port B in this example.



As can be see, an STB signal is sent by the port to start with. At the same time, data is placed on the data buffer. Once the data has been read, an IBF signal is sent out, which causes the port to stop sending data. An interrupt is sent to the microprocessor, which comes and reads the data.

A similar case can be drawn up for output signals instead.

Example 2:

If we are working in **mode 2**, Port A is simultaneously acting as an input and an output device. Because of this, it requires both the IBF and the OBF signals at the same time, which results in a total of 5 different handshaking signals. This is why we had to borrow one from group B.

Other than this, all the details remain the same. We can still use the remaining three port C pins for either basic I/O operations or for handshaking signals for port B.