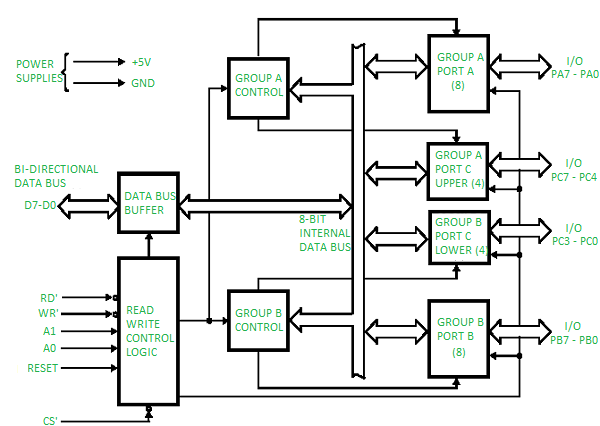
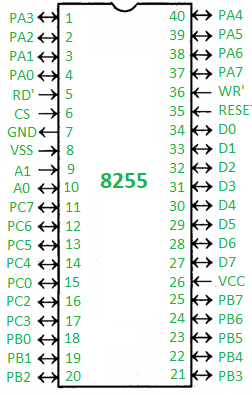
**Programmable peripheral interface 8255**

PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc. We can program it according to the given condition. It can be used with almost any microprocessor. It consists of three 8-bit bidirectional I/O ports i.e. PORT A, PORT B and PORT C. We can assign different ports as input or output functions.



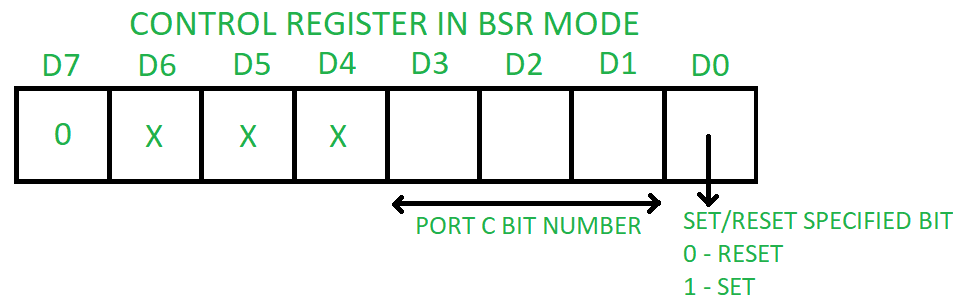
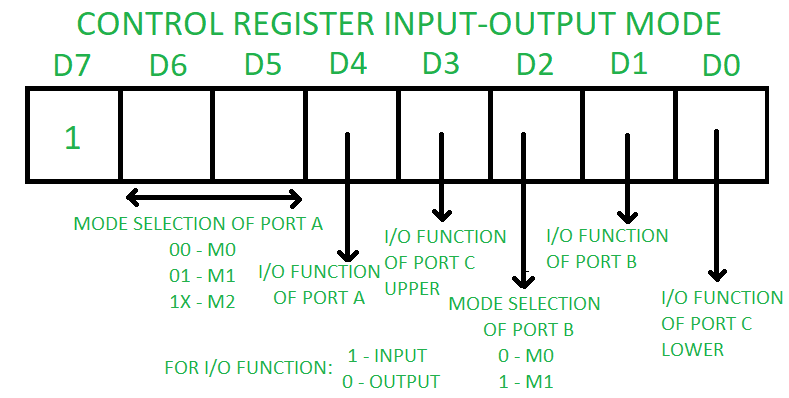
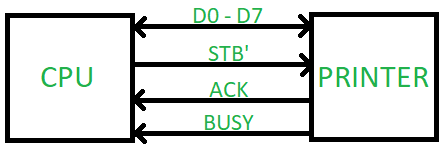
It consists of 40 pins and operates in +5V regulated power supply. Port C is further divided into two 4-bit ports i.e. port C lower and port C upper and port C can work in either BSR (bit set rest) mode or in mode 0 of input-output mode of 8255. Port B can work in either mode 0 or in mode 1 of input-output mode. Port A can work either in mode 0, mode 1 or mode 2 of input-output mode. It has two control groups, control group A and control group B. Control group A consist of port A and port C upper. Control group B consists of port C lower and port B. Depending upon the value if CS’, A1 and A0 we can select different ports in different modes as input-output function or BSR. This is done by writing a suitable word in control register (control word D0-D7).





* **PA0 – PA7 –**Pins of port A
* **PB0 – PB7 –**Pins of port B
* **PC0 – PC7 –**Pins of port C
* **D0 – D7 –**Data pins for the transfer of data
* **RESET –**Reset input
* **RD’ –**Read input
* **WR’ –**Write input
* **CS’ –**Chip select
* **A1 and A0 –**Address pins

**Operating modes –**

1. **Bit set reset (BSR) mode –** If MSB of control word (D7) is 0, PPI works in BSR mode. In this mode only port C bits are used for set or reset.
2. **Input-Output mode –** If MSB of control word (D7) is 1, PPI works in input-output mode. This is further divided into three modes:
   * **Mode 0 –**In this mode all the three ports (port A, B, C) can work as simple input function or simple output function. In this mode there is no interrupt handling capacity.
   * **Mode 1 –** Handshake I/O mode or strobed I/O mode. In this mode either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission. It has interrupt handling capacity and input and output are latched. Example: A CPU wants to transfer data to a printer. In this case since speed of processor is very fast as compared to relatively slow printer, so before actual data transfer it will send handshake signals to the printer for synchronization of the speed of the CPU and the peripherals.
   * **Mode 2 –** Bi-directional data bus mode. In this mode only port A works, and port B can work either in mode 0 or mode 1. 6 bits port C are used as handshake signals. It also has interrupt handling capacity.

**Advantages:**

**Versatility:** The PPI 8255 can be programmed to operate in a variety of modes, which makes it a versatile component in many different systems. It provides three 8-bit ports that can be configured as input or output ports, and supports multiple modes of operation for each port.

**Ease of use:**The PPI 8255 is relatively easy to use and program, even for novice programmers. The control register of the PPI can be programmed using simple commands, which makes it easy to interface with other devices.

**Compatibility:**The PPI 8255 is widely used and has been around for many years, which means that it is compatible with a wide range of devices and software.

**Low cost:**The PPI 8255 is a relatively low-cost component, which makes it an affordable option for many different applications.

**Disadvantages:**

**Limited functionality:** While the PPI 8255 is versatile, it has limited functionality compared to newer I/O interface components. It is not capable of high-speed data transfer and has limited memory capacity.

**Limited number of ports:**The PPI 8255 provides only three 8-bit ports, which may not be sufficient for some applications that require more I/O ports.

**Limited resolution:**The PPI 8255 provides only 8 bits of resolution for each port, which may not be sufficient for some applications that require higher resolution.

**Obsolete technology:**While the PPI 8255 is still used in some applications, it is considered an older technology and is being replaced by newer, more advanced I/O interface components.

**Parallel Interface**

* It is a method of conveying multiple binary digits (bits) simultaneously i.e. all bits of word are transferred at a time.
* It is faster.
* Hardware requirement is complex.
* E.g. 8255A PPI

**Modes:**

**i) Simple I/O**

* When you need to get digital data from simple switch, such as thermostat, into microprocessor, all you have to do is connect the switch to an I/O port line and read the port.
* Likewise, when you need to output data to simple display device, such as LED, all you have to do is connect the input of the LED buffer on an output port pin and output the logical level required to turn on the light.

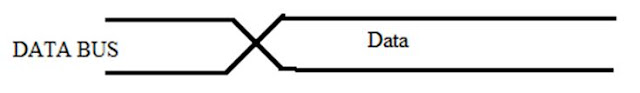
[](https://1.bp.blogspot.com/-71K08gk7w7M/VmG4gjnxdvI/AAAAAAAAAsw/NA-baiRYerI/s1600/1.JPG)

Fig: Simple I/O

* The timing waveform represents the situation.
* The crossed lines on the waveform represent the time at which a new data byte becomes valid on the output lines of the port.
* The absence of other waveforms indicates that this output operation is not directly dependant on any other signal.

**ii) Strobe I/O**

* In many applications, valid data is present on an external device only at a certain time, so it must be read in at that time.
* E.g. the ASCII-encoded keyboard. When a key is pressed, circuitry on the keyboard sends out the ASCII code for the pressed key on eight parallel data lines, and then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines.

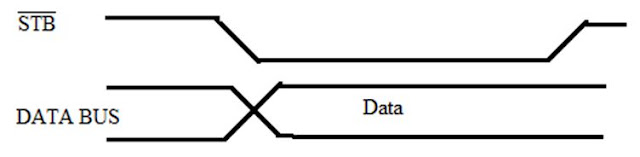
[](https://2.bp.blogspot.com/-txGav2tR_EU/VmG4yZkLoXI/AAAAAAAAAs4/K-kiij_Oo5w/s1600/2.JPG)

Fig: Strobe I/O

* This timing waveform represents strobe I/O.
* The sending device, such as a keyboard, outputs a parallel data on the data lines, and then outputs an STB signal to let you know that valid data is present.
* For low rates of data transfer, such as from a keyboard to a MP, a simple strobe transfer works well.
* However, for higher speed data transfer, this method does not work because there is no signal which tells the sending device when it is safe to send the next data byte.
* In other words, the sending system might send data bytes faster than the receiving system could read them.
* To prevent this problem, a handshake data transfer scheme is used.

**iii) Single handshake I/O**

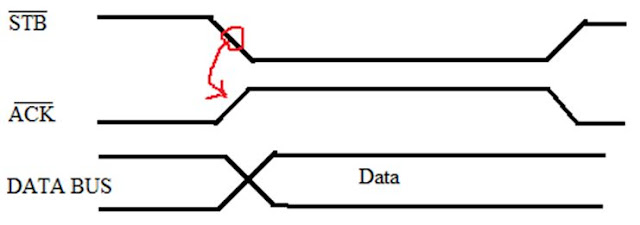
[](https://3.bp.blogspot.com/-qqHPYQG9REg/VmG48zx1NRI/AAAAAAAAAtA/awdhIJDQbFM/s1600/3.JPG)

Fig: Single Handshake I/O

* It shows the timing waveform for a handshake data transfer from a peripheral device to a MP.
* The peripheral outputs some parallel data and sends an STB signal to the MP.
* The MP detects the asserted STB signal on a polled or interrupts basis and reads in the bytes of data.
* Then, the MP sends ACK (acknowledge) signal to the peripheral to indicate that the data has been read and that the peripheral can send next byte of data.
* The point of this method is that the sending device or system is designed so that it does not send the next byte until the receiving device or system indicates with an ACK signal that it is ready to receive the next byte.

**iv) Double handshake I/O**

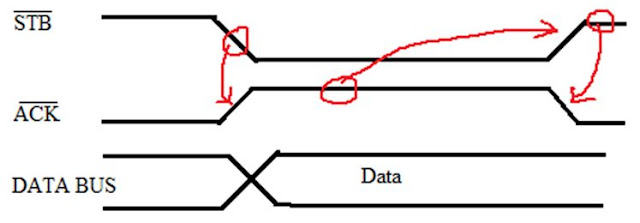
[](https://4.bp.blogspot.com/-hUYh88LwkWA/VmG5HjHMoCI/AAAAAAAAAtI/fbN-m9fXVlU/s1600/4.JPG)

Fig: Double Handshake I/O

* For data transfer where even more coordination is required between the sending system and the receiving system, a double handshake is used.
* The sending (peripheral) device asserts its STB line low to ask the receiving device whether it is ready or not for data reception.
* The receiving system raises its ACK line high to indicate that it is ready.
* The peripheral device then sends the byte of data and raises its STB line high to assure that the valid data is available for the receiving device (MP).
* When MP reads the data, it drops its ACK line low to indicate that it has received the data and requests the sending system to send next byte of data.

I/O interfacing is defined as the interfacing existing between the processing unit or CPU and I/0 devices such as – keyboard, mouse, printer, etc. Based on the kind of interfacing link between CPU and [I/O devices](https://www.electricalvolt.com/2023/03/plc-digital-input-and-digital-output-modules/), it can be subdivided into the following two categories –

1. Memory-Mapped I/O Interfacing
2. Standard I/O Mapped I/O Interfacing

**What is Memory-Mapped I/O Interfacing?**

Memory-Mapped I/O devices are treated as memory locations and not any separate I/O peripherals. In the case of memory-mapped I/O peripherals no separate memory is allocated for I/O devices, i.e., they share the same memory. A 16-bit address is assigned for the I/O devices in the memory.

Ezoic

Moreover, I/O devices and memory share the same bus to transfer data to and from the processor. As a result, this minimizes the addressing and memory capacity. The memory-mapped I/O interfacing is used in applications having small memory requirements.

In this scenario, the I/O ports are considered memory locations used for performing the writing and reading operations. The same set of instructions is used for the transfer of data to and from the I/O or the memory. A control signal is activated whenever an address is sent over the address line. In a memory-mapped I/O peripheral, the data from the I/O device is sent to the **ALU (arithmetic and logical unit).**

EzoicThe assemble language instructions namely LOAD and STORE are utilized to execute the read and write operations from both the I/O peripherals and the memory.

**What is I/O Mapped I/O Interfacing?**

I/O Mapped I/O devices are also called isolated I/O mapping as the I/O devices are allotted separate addressing space from the memory. These distinct spaces specially allotted for the I/O peripherals are called as ‘Ports’. Therefore, separate instructions for reading and writing are present for both I/O and memory.

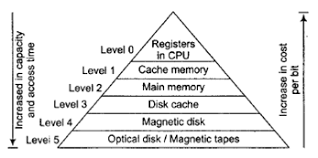
The I/O devices in this case have been assigned distinct read and write control lines. The data transfer in the case of I/O mapped I/O devices occurs such that when a piece of data or information has to be transmitted from the I/O to the processes it is firstly placed on the common address bus, followed by the activation of the I/O read and I/O write control lines in order to enable this data transfer.

This kind of system is applicable in scenarios where the memory requirements are large. In contrast to the memory-mapped I/O system, the I/O mapped I/O system operates vis the I/O read and I/O write cycles. However, it interfaces comparatively lesser ports i.e. approximately 256. Moreover, the arithmetic and logic unit (ALU) operations cannot be performed using the information accessed from the I/O devices of the I/O mapped I/O system approach.

**Difference Between Memory-mapped I/O and I/O-mapped I/O**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Memory-mapped I/O** | **I/O-mapped I/O** |
| Functionality | I/O peripherals share the same instructions and address space. | I/O peripherals are allotted separate or isolated, special address spaces called ports. |
| Width of Assigned address space | The size of the address space is about 16 bits. | The size of the address space is about 8 bits. |
| Instruction types | The type of instructions involved in memory writing and memory reading. | The type of instructions involved in I/O writing and I/O reading. |
| Instructions for data transfer | Both memory and I/O devices share the same type of instructions for the transfer of data. | Memory and I/O devices have separate and distinct types of instructions for the transfer of data. |
| Ports for I/O interfacing | The interfacing size for memory-mapped I/O devices is quite large i.e., 64K bytes. | The interfacing size for I/O-mapped I/O devices is comparatively less i.e., 256 bytes. |
| Control signals | No distinct control signals are involved in the transfer of data. | Special and distinct control signals are involved in the transfer of data, namely – I/O read, and I/O write control signals. |
| Efficacy | The efficiency of memory-mapped I/O systems is relatively low. | The efficiency of I/O-mapped I/O systems is relatively high. |
| Hardware for decoder | Comparatively more hardware is needed to design a decoder. | Comparatively lesser hardware is needed to design a decoder. |
| Status of pin IO/M | The value of the IO/M pin of the system is set to 0 (i.e., low) when the memory read and memory write cycles are executed. | The value of the IO/M pin of the system is set to 1 (i.e., high) when the I/O read and I/O write cycles are executed. |
| Movement of data | Data is transmitted between ports and registers in the case of a memory-mapped I/O approach. | Data is transmitted between ports and accumulators in the case of the I/O-mapped I/O approach. |
| System complexity | The system is relatively simple due to the absence of separate control signals and requires a lesser number of pins. | The system is relatively complex due to the presence of separate control signals and requires a greater number of pins. |
| Speed | The speed of operation is relatively low. | The speed of operation is relatively high. |
| Application | The memory-mapped I/O approach is used in the case of smaller systems requiring less memory space. | The I/O mapped I/O approach is used in the case of larger systems requiring greater memory space. |

**Memory Device Hierarchy**

[](https://ioesolutions.esign.com.np/storage/uploads/memhier_1598015568.png)

- Memory hierarchy is the pyramid of different memory technology that provides comparative analysis of memory technologies based on cost per bit, capacity, access time and so on.  
- The main goal is to use the memory so as to obtain highest possible access speed minimizing the total cost.  
- The memory hierarchy is shown below:  
- As we go down, cost per bit decrease, capacity increase and access time increase.