**Priority Interrupts | (S/W Polling and Daisy Chaining)**

In [I/O Interface (Interrupt and DMA Mode)](https://www.geeksforgeeks.org/computer-organization-architecture/io-interface-interrupt-dma-mode/), we have discussed the concept behind the Interrupt-initiated I/O. To summarize, when I/O devices are ready for I/O transfer, they generate an interrupt request signal to the computer. The CPU receives this signal, suspends the current instructions it is executing, and then moves forward to service that transfer request. But what if multiple devices generate interrupts simultaneously. In that case, we have a way to decide which interrupt is to be serviced first. In other words, we have to set a priority among all the devices for systemic interrupt servicing. The concept of defining the priority among devices so as to know which one is to be serviced first in case of simultaneous requests is called a priority interrupt system. This could be done with either software or hardware methods.

SOFTWARE METHOD – POLLING

In this method, all interrupts are serviced by branching to the same service program. This program then checks with each device if it is the one generating the interrupt. The order of checking is determined by the priority that has to be set. The device having the highest priority is checked first and then devices are checked in descending order of priority. If the device is checked to be generating the interrupt, another service program is called which works specifically for that particular device. The structure will look something like this-

if (device[0].flag)  
 device[0].service();  
else if (device[1].flag)  
 device[1].service();  
.  
.  
.  
.  
.  
.  
else  
 //raise error

The major disadvantage of this method is that it is quite slow. To overcome this, we can use hardware solution, one of which involves connecting the devices in series. This is called Daisy-chaining method.

HARDWARE METHOD – DAISY CHAINING

The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner. This configuration is governed by the priority of the devices. The device with the highest priority is placed first followed by the second highest priority device and so on. The given figure depicts this arrangement.**WORKING:** There is an interrupt request line which is common to all the devices and goes into the CPU.

* When no interrupts are pending, the line is in HIGH state. But if any of the devices raises an interrupt, it places the interrupt request line in the LOW state.
* The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.
* This signal is received at the PI(Priority in) input of device 1.
* If the device has not requested the interrupt, it passes this signal to the next device through its PO(priority out) output. (PI = 1 & PO = 1)
* However, if the device had requested the interrupt, (PI =1 & PO = 0)
  + The device consumes the acknowledge signal and block its further use by placing 0 at its PO(priority out) output.
  + The device then proceeds to place its interrupt vector address(VAD) into the data bus of CPU.
  + The device puts its interrupt request signal in HIGH state to indicate its interrupt has been taken care of.
* If a device gets 0 at its PI input, it generates 0 at the PO output to tell other devices that acknowledge signal has been blocked. (PI = 0 & PO = 0)

Hence, the device having PI = 1 and PO = 0 is the highest priority device that is requesting an interrupt. Therefore, by daisy chain arrangement we have ensured that the highest priority interrupt gets serviced first and have established a hierarchy. The farther a device is from the first device, the lower its priority.

**Priority interrupts:**

**Advantages:**

1. Priority interrupts allow for the efficient handling of high-priority tasks that require immediate attention. This is especially important in real-time systems where certain tasks must be completed within strict time constraints.
2. They are more efficient than software polling as the processor does not waste time constantly checking for events that have not occurred.
3. Priority interrupts are also more deterministic, as the response time to an event can be accurately predicted based on its priority level.

**Disadvantages:**

1. One potential disadvantage of priority interrupts is the possibility of lower priority tasks being starved of resources if high-priority tasks are continuously interrupting the processor.
2. If not implemented properly, priority interrupts can lead to priority inversion, where a low-priority task holds a resource required by a higher-priority task, causing a delay in the high-priority task's execution.

**Software polling:**

**Advantages:**

1. Software polling is relatively simple to implement and does not require specialized hardware.
2. It can be used to detect events that occur at irregular intervals, as the processor can check for events whenever it is not performing other tasks.

**Disadvantages:**

1. Software polling is less efficient than priority interrupts as the processor must constantly check for events even if none have occurred.
2. In real-time systems, software polling may not be suitable as it is difficult to guarantee the response time to an event, especially if the processor is busy with other tasks.

**Daisy chaining:**

**Advantages:**

1. Daisy chaining allows multiple devices to share a single interrupt line, reducing the number of interrupt lines required.
2. It is relatively simple to implement and does not require specialized hardware.

**Disadvantages:**

1. Daisy chaining can result in increased response time as each device must wait for the previous device to complete its interrupt handling before it can start its own.
2. It can also be difficult to implement and troubleshoot, especially if there are multiple devices on the same interrupt line.