Memory-mapped IO vs Port-mapped IO

Microprocessors normally use two methods to connect external devices: **memory mapped**or **port mapped** I/O. However, as far as the peripheral is concerned, both methods are really identical.

Memory mapped I/O is mapped into the same address space as program memory and/or user memory, and is accessed in the same way.

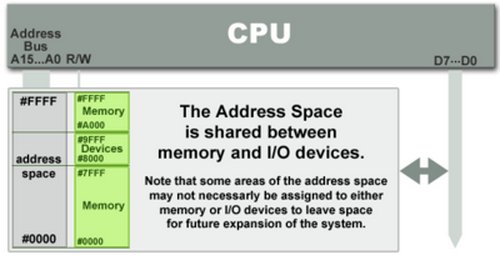
Port mapped I/O uses a separate, dedicated address space and is accessed via a dedicated set of microprocessor instructions.

The difference between the two schemes occurs within the microprocessor. Intel has, for the most part, used the port mapped scheme for their microprocessors and Motorola has used the memory mapped scheme.

As 16-bit processors have become obsolete and replaced with 32-bit and 64-bit in general use, reserving ranges of memory address space for I/O is less of a problem, as the memory address space of the processor is usually much larger than the required space for all memory and I/O devices in a system.

Therefore, it has become more frequently practical to take advantage of the benefits of memory-mapped I/O. However, even with address space being no longer a major concern, neither I/O mapping method is universally superior to the other, and there will be cases where using port-mapped I/O is still preferable.

Memory-mapped IO (MMIO)



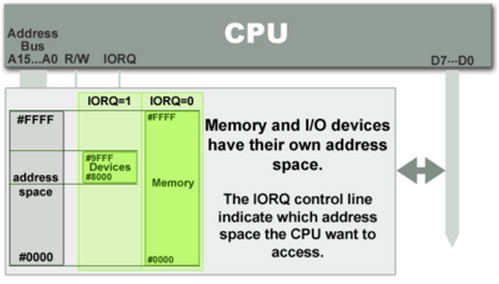
Picture source : [IO Devices](http://www.grimware.org/doku.php/documentations/devices/io.devices)

I/O devices are mapped into the system memory map along with RAM and ROM. To access a hardware device, simply read or write to those 'special' addresses using the normal memory access instructions.

The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device.

The disadvantage to this method is that the entire address bus must be fully decoded for every device. For example, a machine with a 32-bit address bus would require logic gates to resolve the state of all 32 address lines to properly decode the specific address of any device. This increases the cost of adding hardware to the machine.

Port-mapped IO (PMIO or Isolated IO)



Picture source : [IO Devices](http://www.grimware.org/doku.php/documentations/devices/io.devices)

I/O devices are mapped into a separate address space. This is usually accomplished by having a different set of signal lines to indicate a memory access versus a port access. The address lines are usually shared between the two address spaces, but less of them are used for accessing ports. An example of this is the standard PC which uses 16 bits of port address space, but 32 bits of memory address space.

The advantage to this system is that less logic is needed to decode a discrete address and therefore less cost to add hardware devices to a machine. On the older PC compatible machines, only 10 bits of address space were decoded for I/O ports and so there were only 1024 unique port locations; modern PC's decode all 16 address lines. To read or write from a hardware device, special port I/O instructions are used.

From a software perspective, this is a slight disadvantage because more instructions are required to accomplish the same task. For instance, if we wanted to test one bit on a memory mapped port, there is a single instruction to test a bit in memory, but for ports we must read the data into a register, then test the bit.

Comparison - Memory-mapped vs port-mapped

|  |  |
| --- | --- |
| Memory-mapped IO | Port-mapped IO |
| Same address bus to address memory and I/O devices | Different address spaces for memory and I/O devices |
| Access to the I/O devices using regular instructions | Uses a special class of CPU instructions to access I/O devices |
| Most widely used I/O method | x86 Intel microprocessors - IN and OUT instructions |