Why you need expandable multi-serial card

1. The limitation in standard multi-serial card:

Generally one standard multi-serial card can support upto 8 serial port in one card. When we need to add extra port number, we just need to add one more card. When we need to add one more card, it means that we need one more PCI slot and IRQ for this card. So we may have some problem in this condition. We may not have enough PCI slot available for this card. We may have IRQ assigned for this card to share with other card. Because the IRQ for PCI card is assigned by BIOS. Even though multi-serial card can share IRQ with other card. But we can not confirm that card can share with you. It seems that we need more simple solution to add extra serial port.

2. We need expandable card:

Due to the limitation in standard multi-serial card we need to add extra serial port in current add-on card. So we don't need one more PCI slot and one more IRQ.

We will insert our card in PCI slot to communicate with motherboard's main CPU. We will connect this card with external connector box. One box can support 8 serial ports. When we need to add more serial port, we just need to add extra connector box. In this condition there are no difference in motherboard's environment. So there are no confliction in hardware. In software issue our driver can detect such condition to add extra COM port. So it is so simple for you to add extra serial port in expandable card.

3. Do we need local processor:

When we add extra connector box to add more serial ports, we may find one condition. We just have one IRQ to service all serial port. More serial ports will generate IRQ requirement to motherboard's main CPU frequently. The main CPU need to hold current task and process your IRQ service routine. It means that the efficiency of main CPU will be degraded quickly. We may use two types of IRQ service routine structure.

One type is vector structure. We can use one vector to indicate the serial port to generate IRQ (the highest priority one will be shown). Then main CPU will service this serial port and return. If there were other serial port waiting, main CPU will enter IRQ service routine again. This process will be repeated till every one serviced (No more IRQ stand by). Because we need standard procedure to enter IRQ service routine and leave IRQ service routine (this is the minimum overhead for each IRQ service routine). It means that each serial port to ask IRQ service need to cover this overhead. When the serial port to ask IRQ service grow more and more, then the main CPU efficiency degrade quickly.

The other type is IRQ polling structure. When we enter IRQ service routine, we will ask every serial port may generate this IRQ. We check serial port one by one. Then we service the one to ask service. After every serial port are polled, we can leave IRQ service routine. So we can find many serial ports to share the overhead for IRQ service routine. But we also add more overhead to poll the serial port do not ask IRQ service.

Why you need expandable multi-serial card

Anyway we can use vector type or polling type IRQ service structure in different operating system (Generally WIN95/98/Me use polling method and WINNT use vector method). But it is not easy for us to estimate the IRQ service time for multi-serial card. So it is not easy to estimate the main CPU efficiency. When more and more serial port need to ask IRQ service, the main CPU efficiency is degraded quickly. So we must have another method to solve this condition.

Fortunately we can use one local CPU in multi-serial card.All the data transmission task are handled by local CPU. The motherboard's main CPU just need to prepare data or get data in dual port RAM. So we don't need to use IRQ in PCI slot. We just need to exchange data in dual port RAM periodically. Generally we will use system time tick to ask main CPU to exchange data. So the time to exchange data will be predictable in main CPU. But there are one problem for real-time status monitor condition. Because all the serial port data transmission are handled by local CPU. And all the serial port status are handled by local CPU. When one state change condition happened in serial port. Local CPU will sense this condition and inform main CPU in dual port RAM. When main CPU check this condition in dual port RAM and put command in dual port RAM to handle this condition. Then local CPU will get this command and do response in serial port. This procedure may be long enough for something happened. In previous IRQ structure we may not have such problem. Because main CPU will sense this condition and do action immediately. For example, we may have 100Hz system clock (or 10ms time tick period) to exchange data in dual port RAM. When the DCD signal in serial port changed, local CPU sense this condition and put status changed in dual port RAM. Generally we need wait 5ms (half-period of 10ms) later for main CPU to sense this condition and do response. It may be too late for some application.

Why you need expandable multi-serial card

4. Conclusion:

From above information we can know that the multi-serial card without local processor is suitable for real time application. And the card with local processor is suitable for high performance and CPU efficiency predictable application. If one can design a card to have local CPU respond all condition, then we can have real-time response capability and main CPU efficiecy predictable feature. But it is not easy for one standard product to have such function and feature. Because every one may have dedicated response for their application. So we have IOP3927U card to support user development feature. User can develop their special application in IOP3927U card. IOP3927U card can support upto 64 serial ports in one PCI slot. The local processor is 133MHz MIPS CPU with 16MByte local RAM. User can have upto 512KByte code area in flash ROM.