Talk To Multiple Devices With One UART

GAURANG KAVAIYA | Microchip Technology Inc., Chandler, Ariz.
Gaurang.Kavaiya@microchip.com

The Universal Asynchronous Receive and Transmit (UART) interface is found on a variety of peripheral devices. Consider, for instance, a microcontroller-based system with four such peripherals. Ideally, in low-cost embedded applications, you would like to connect multiple peripherals to a single UART. However, a lack of chip-select signals in UARTs complicates such a task.

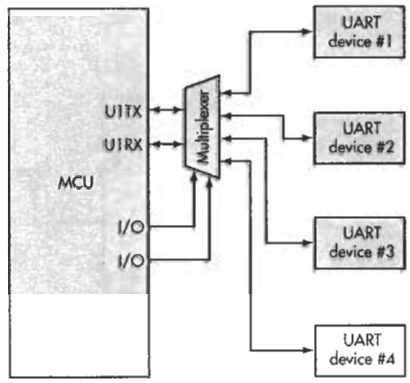
This is a common design problem, and there are a few conventional ways of solving it. The most obvious approach is to use an MCU with as many hardware UART modules as you need. Very rarely will you find an economical MCU in a compact form factor that features four hardware UART modules.

Therefore, you may need to use a high-pin-count, higher-performance MCU, even if all of the UARTs aren't used simultaneously. But employing a large MCU may be overkill for the application and may not provide a cost-effective solution.

If all UARTs aren't used simultaneously in the application, time-multiplexing a hardware UART module to four UART-enabled peripherals can be accomplished with a hardware multiplexer and a few MCU I/O lines *(Fig. 1)*. In this case, the control program will map UART hardware to the required peripheral by controlling multiplexers.

There's another innovative way to solve this problem, based on the flexible I/O pin-mapping feature offered by some MCUs. Unlike the traditional fixed I/O pinouts, these MCUs feature a set of I/O pins onto which you can map many peripheral functions. This feature, available as Peripheral Pin Select (PPS) on Microchip's MCUs, lets you define the MCU pinout as per your choice. The control program in these MCUs dynamically changes the I/O pinout after executing a certain code sequence to unlock configuration registers.

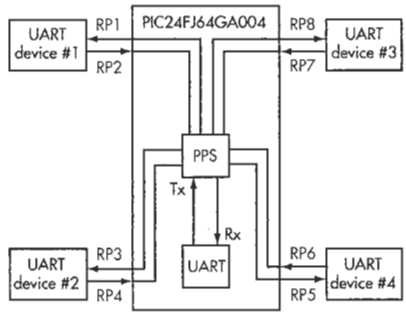
To obtain four or more UARTs on these types of devices, you start by connecting four UART-based peripherals to the MCU's remappable pins. In this example, we used the PIC24FJ64GA004 16-bit MCU *(Fig. 2)*.



1. On fixed I/O MCUs, a multiplexer lets you communicate with multiple UARTs.

The control program will then dynamically map a UART in hardware to these four pin sets, based on the application's need. For example, when the application must communicate with device 1, the program will map hardware UART to the RP1 and RP2 pins. Likewise, the process is duplicated when the MCU wants to talk to device 2.

The above approach works well if all UART channels are operating in a master/slave configuration—the peripheral device will only send data when requested by the MCU acting as the master.



2. A single MCU can interface with multiple UART devices, based on a flexible I/O pin-mapping feature. RPn denotes the pins where peripherals can be remapped.

So, what if the system needs a combination of both master and slave devices? Thanks to its versatility, the flexible I/O pin-mapping approach also works in this case. For instance, the system may need to talk to a peripheral asynchronously. In such a scenario, you can exploit the fact that many of these MCUs also feature two hardware UART modules. Simply designate one hardware UART module as an asynchronous communication channel and time-multiplex the other UART module to create multiple hardware UART modules.

In another scenario, your application may need multiple asynchronous channels with multiple slave channels. If available, take advantage of the MCU's asynchronous channel's handshaking capability with Data Terminal Ready (DTR)/Clear To Send (CTS) signals. You can use the DTR/CTS signals to hold the asynchronous channel while the UART gets mapped back to the original pin.

Where peripheral devices don't have handshaking signals, another workaround can be applied. Use edge interrupts or direct an input capture signal onto the inactive UART receive pin with the flexible I/O pin-mapping feature. If a device starts an asynchronous data transfer, the control program serves an interrupt and immediately switches the hardware UART module to the appropriate pins to receive this data.

If your application requires all four UART channels to be asynchronous channels, then the above solutions may not work and you may still need an MCU with four hardware UARTs. But for most systems, this may not be the case. As a result, the solutions mentioned here will allow you to use a single UART to talk to multiple devices.

GAURANG KAVAIYA, *engineering group manager, Applications, holds a BSEE from Gujarat University, India.*

ED ONLINE 17447