Squeeze extra outputs from a pin-limited microcontroller

Abel Raynus, Armatron International Inc, Malden, MA

Many of today's designs use low-cost microcontrollers from Freescale and Microchip, but during the last decade, device packages have resorted to ever-smaller footprints featuring as few as eight or even six pins. Although these packages minimize pc-board area, they also reduce the number of available I/O pins and pose problems for designers who need to add one more function without migrating to a device that occupies a larger package.

To overcome a shortage of inputs, a designer can increase a small microcontroller's inputs by writing a program that multiplexes and polls the input pins. However, this approach doesn't lend itself to extending outputs, because most designs require simultaneously driving multiple pins. **Figure 1** shows how to solve the problem by adding a shift register.

For example, you can add an eight-LED bar graph to a design based on IC_1, Freescale Semiconductor's 9-bit, flash-memory MC68HC908QT1 microcontroller, which has only eight pins. The device includes only four general-purpose outputs and thus by default cannot drive eight discrete LEDs. To solve the problem, you can add IC_2, a 74HC595 serial-input/serial-output/parallel-output latching shift register available from On Semiconductor and other vendors. The register's latching function allows selective drive of only those LEDs associated with specific data bits.

According to its data sheet, the 74HC595 accepts signals through the SPI protocol. Unfortunately, low-end microcontrollers, such as the MC68-HC908QT1, lack SPI hardware, but you can simulate the SPI in software by following these steps:

1. Unlatch the shift register's outputs by deasserting microprocessor IC_1's PA4 pin.
2. Starting with the MSB, copy a bit from the processor's internal data register and transfer the bit to the processor's PA0 (SD) output.
3. Generate a clock pulse at Pin PA1.
4. Repeat steps 2 and 3 for all eight data bits.
5. Assert the microprocessor's PA4 output to latch the data into IC_2, the 74HC595.

Figure 2 shows the timing diagram for transmitting data byte $F0 from IC_1 to IC_2.

Available from the online version of this Design Idea at www.edn.com/050804di1, **Listing 1** illuminates the LEDs by sending five consecutive bytes to IC_2 and the LEDs: $03, $0c, $30, $c0, and $55. The first four bytes progressively illuminate two LEDs along the bar-graph display at one step per second. The last byte illuminates and latches all odd-numbered LEDs. The **listing** contains only commonly used instructions that easily translate into other microcontrollers' assembly languages.

The SPI requires only three output pins, which frees the microcontroller's remaining I/O pins for other functions and allows remote installation of the shift register/LED driver—for example, on a separate display board with the LEDs. Also, when suitably buffered, the register's outputs can drive other loads, such as motors, relays, and incandescent lamps. EDN

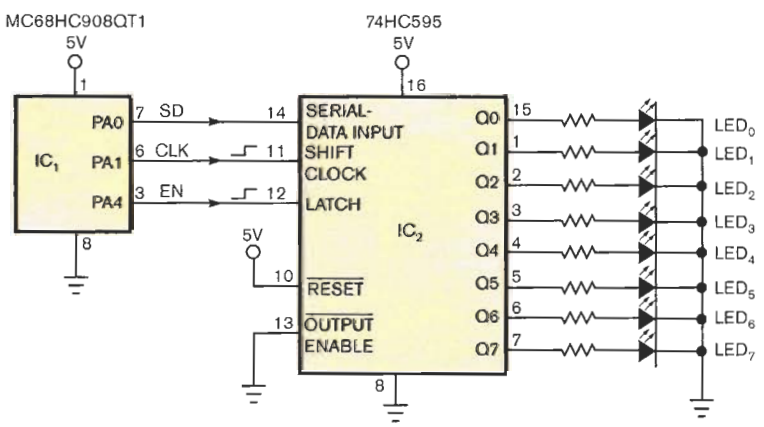

Figure 1 Do you need more outputs? You can emulate an SPI in software to add a shift register to a pin-limited microcontroller.

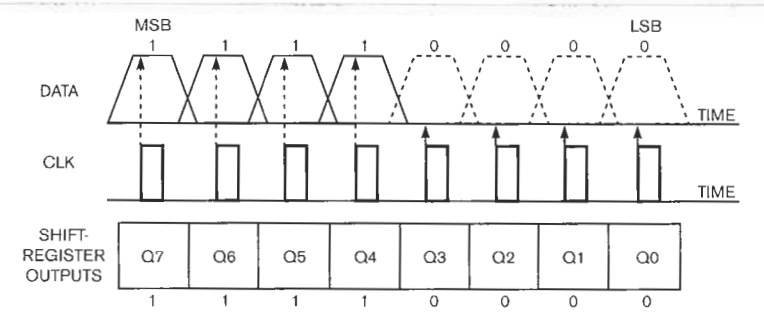

Figure 2 The sample timing diagram illustrates the loading of $F0 byte into an external shift register.