Nuts And Volts #6

Gene Zumchak
Buffalo, NY

In my last column, I reviewed the hardware aspects of handshaking and also described the workings of a programmable input/output port. I suggested that I would detail the transmit and receive software for using handshaking to pass a block of data between two systems. While I still plan to do that (some day), I think a more meaningful exercise would be to consider the hardware and software necessary to connect a common peripheral to a computer system. Namely, I would like to consider the connection of a parallel style, or so-called "Centronics" style printer.

As I mentioned last time, (actually a whole line was left out during the typesetting), Centronics style handshake timing has three possible flags. The one asserted by the sender is a pulse called $\overline{\text{DATA STROBE}}$ and is usually low-true. The receiver responds with a BUSY indication, usually high-true. When BUSY falls false (unbusy), the $\overline{\text{ACK}}$ pulse (low-true) is asserted. Again as mentioned, only one of the return flags, either BUSY or $\overline{\text{ACK}}$, need be used.

I just recently took delivery of a NEC Spinwriter with a parallel interface. While it didn't quite take the day I allowed to get it running with my SYM, I did encounter a few surprises. While the connector and the pin assignments are definitely Centronics style, the flags are not true Centronics.

The product description manual gives one short paragraph on the timing. Fortunately, the accompanying timing diagram accurately describes the timing. While the printer has a flag called BUSY, it is not the Centronics BUSY, and does not take part in the handshake sequence. To avoid confusion, it might better have been called READY. When this BUSY goes low, it indicates that the printer is ready to receive data. Only $\overline{\text{ACK}}$ takes place in handshaking. An initial $\overline{\text{ACK}}$ is sent to indicate that it is ready for the first character, thus, $\overline{\text{ACK}}$ is used to prompt for characters rather than to indicate that the printer is through processing the last character. Actually, for characters beyond the first, the two descriptions mean the same thing. For my Receive Only model (5530), characters are accepted with the handshaking, but no characters are printed until the buffer is full, or a carriage return is received. Busy goes true when the buffer is full, and remains high while the line is being printed. This timing is shown in Fig. 1.

The easiest way to handle Centronics timing is to poll BUSY and forget about $\overline{\text{ACK}}$. However, this is not true Centronics timing, and BUSY takes no part in the handshaking. Since $\overline{\text{ACK}}$ is a narrow pulse (2.2 microseconds), it cannot be polled, but must be used to set a flip-flop. This precludes the use of a simple input port bit to handle the flag from the printer. There are two choices. We can either use a family port chip which has edge-sensitive inputs (like the 6522), or we must provide the flip-flop, which is reset by ACK and polled as a conventional BUSY flag with an input bit. DATA STROBE can be used to set the BUSY flip-flop when a character is sent. This alternative is shown in Fig. 2.

Since my SYM has no fewer than three 6522s which are available for I/O, I did not have to provide the flip-flop, but could use one of the edge-sensitive input control bits. The connection I used is shown in Fig. 3. I used the VIA chip that responds to the base address $A800. I used the low seven bits of Port A for the printer data, and bit PA7 as an input bit to poll BUSY. The CA2 output bit was used for the DATA STROBE and input control bit CB1 was used to detect the ACK edge. As can be seen from the figure, the data lines and strobe were buffered. This was necessary since the Spinwriter inputs have a 470 ohm pullup resistor. When the input is zero, there are five volts across 470 ohms and about 10 milliamps are drawn in addition to the 1.6 mA TTL input. A family port chip can usually drive only one TTL load, but certainly not 11.6 mA. I used an octal three-state gate chip for the buffer (81LS97). Any non-inverting gate, like the 74LS241 would be suitable.

A suitable program for sending a character to the printer is shown. For those not familiar with the 6522, certain defined events cause a flag in the Interrupt Flag Register (IFR) to be set. A corresponding bit in the Interrupt Enable Register (IER) can be set with software to enable the occurrence of any particular event to generate an interrupt as well. In this case, interrupts are not desirable and we will poll the IFR to see if our flag has been set. Four control bits, CA1, CA2, CB1, and CB2 can be used as input/output flags for handshaking.

CA2 and CB2 can be outputs which are automatically asserted when data is read or written to the corresponding port. For write handshaking, for example, CA2 can be made to go low automatically when writing to Port A. CA2 is returned high by a transition on the CA1 input. Alternatively, CA2 can be pulsed low on a write to Port A, returning high without feedback after 500 ns. Finally, CA2 can be manually made low or high with software. Both CA2 and CB2 can serve as edge-sensitive inputs.

It is assumed that the character to be printed is in the accumulator when the program is called. It is good practice for an output routine to leave the registers unchanged. Since X and Y are not used in the program, they will not be affected and need not be saved. The accumulator is pushed on the stack twice; once for later use in the program, and a second time so that the program can terminate with A unchanged. This permits one output routine to follow another.

Lines 260 and 270 cause the low seven bits of Port A to be made outputs, keeping PA7 as an input bit. The PCR register, which defines the use of the CA and CB control pins, is initialized with data "$0E", called STRBOFF, which manually forces CA2 high.

The BUSY output is polled in the first loop, WAIT1. When BUSY goes false, the program falls into the second loop, WAIT2, where the IFR is read and the bit corresponding to the CB1 input flag is tested. This flag is set by a transition on the ACK line. When that condition is found, the flag is cleared by reading port B or Input Register B (IRB). Now the character to be printed is pulled from the stack and sent to port A. The DATA STROBE is exercised, by manually forcing CA2 low, then high again. This destroys the character in A, which is restored prior to the return with the second pull from the stack.

This program is by no means the only solution. There would appear to be a large number of possible connections of the port bits to accomplish the same thing, and perhaps different programs for a particular connection. However, many combinations and programs will not work. For example, why did I manually lower and raise the CA2 strobe? Why not program CA2 for the handshake pulse mode and let it pulse automatically when data is written to the A port? I confess that I tried it. Since auto pulsing will also occur for a "read" of port A as well, when an attempt was made to poll BUSY at WAIT1, data was unintentionally strobed. This caused the same character to be printed more than once. Another possible change would be to detect ACK with the CA1 input instead of CB1, and keep all functions in Port A. Again, reading Port A at WAIT1 would cause the CA1 flag to be cleared before it was recognized. The program would then fall into the loop at WAIT2 and wait for an $\overline{\text{ACK}}$ signal that would never occur since the printer would be waiting for a DATA STROBE that will never occur.

An experienced programmer will not panic when something strange happens, or worse yet, nothing at all happens. In this case, most of the unexpected results can be predicted with a careful reading of the 6522 spec' sheet.

I was initially annoyed because a legitimate BUSY flag wasn't available and also because two return flags seemed to be required. Actually, the printer has two additional output flags that I did not choose to use, one called FAULT and another for PAPER OUT. Both of these flags, however, are reflected in the BUSY flag. That is, if a fault occurs (cover not closed) or if the PAPER OUT switch is engaged, BUSY will not return false. A PAPER indicator appears on the printer panel, as well as a READY light. Thus, nothing is lost by not using these additional flags.

Is, in fact, BUSY necessary? After I had a program running successfully, I NOPed the WAIT1 loop. I then created "paper out" and "fault" conditions. The printer stopped and the READY light went out. Printing resumed as soon as the condition cleared. Thus, it appears that the printer always affects BUSY before $\overline{\text{ACK}}$, and $\overline{\text{ACK}}$ will not be asserted if an unready state exists. It would then appear that the information contained in BUSY is in fact redundant, and that only one flag, $\overline{\text{ACK}}$, and one wait loop need be used to obtain normal print operation.

The actual incorporation of the Print Character program just considered will depend upon the particular computer system which you are using, and how it handles input and output. This is, in fact, worthy of an article by itself, and I am preparing an article on console input/output, if not for this issue, then the next.

If one is attempting to interface a Spinwriter to a PET, the above program is not necessary, if the printer is interfaced by the GPIB bus. In this case, the polling and sending of flags is performed routinely internally. However, the printer will need a hardware interface that makes it look like a legitimate GPIB instrument. While building this

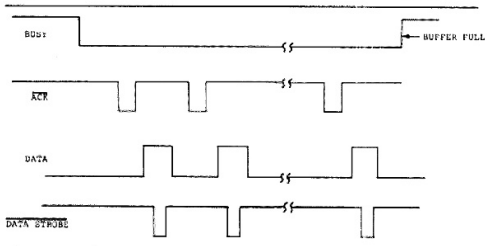

Figure 1. Timing Signals for NEC Spinwriter 6522

interface is not trivial, it would not appear to require more than a handful of gates and a flip-flop or two, perhaps $10 or less in parts (sans connectors). Since I do not have a PET, I cannot verify my gut feeling. However, I do hope to get my hands on one in the near future, and it should result in an article on building a GPIB printer interface, or perhaps building an interface for any non-standard peripheral (without using another microcomputer on the inside).

I welcome your feedback on my articles. I know that at least two people read my column.

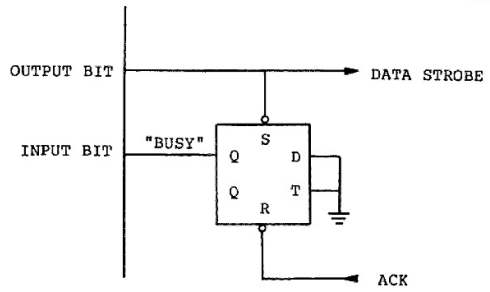

Figure 2. Flip-flop to generate "Centronics" type BUSY from ACK and DATA STROBE.

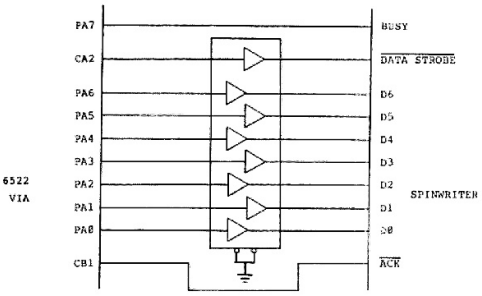

Figure 3. Connection from VIA to Spinwriter.

```
                        0010 ;BASIC HANDSHAKING WRITE ROUTINE FOR SPINWRITER
                        0020            ;
                        0030            ;2/25/80
                        0040            ;
                        0050            ; $A800 IS THE BASE ADDRESS OF 6522
                        0060            ; ON SYM USED FOR PRINTER PORT
                        0070            ;
                        0080 IRB        .DE $A800
                        0090 ORA        .DE $A801     ; LOW SEVEN BITS PORT A
                        0100            ;                IS PRINTER DATA
                        0110 IRA        .DE $A801     ; PA7 IS USED TO POLL BUSY
                        0120 DDRA       .DE $A803
                        0130 PCR        .DE $A80C     ; DETERMINES CONTROL BIT USE
                        0140            ;                CA2 USED AS OUTPUT FOR
                        0150            ;                PRINTER DATA STROBE
                        0160 IFR        .DE $A80D     ; BIT 4 (CB1) USED FOR ACK
                        0170 STRBOFF    .DE $0E       ; MAKES CA2 HIGH
                        0180 STRBON     .DE $0C       ; MAKES CA2 LOW
                        0190            ;
                        0200            .BA $300
                        0210            .MC $300
                        0220            .OS
                        0230            ;
0300- 48        0240 PRINTCHAR  PHA  ; SAVE FOR POSTERITY
0301- 48        0250            PHA
0302- A9 7F     0260            LDA #$7F      ; MAKE LO-7 BITS PORTA OUTPUTS
0304- 8D 03 A8  0270            STA DDRA
0307- A9 0E     0280            LDA #STRBOFF          ; MAKE SURE CA2 HIGH
0309- 8D 0C A8  0290            STA PCR
030C- AD 01 A8  0300 WAIT1      LDA IRA       ; WAIT FOR UNBUSY
030F- 30 FB     0310            BMI WAIT1
0311- AD 0D A8  0320 WAIT2      LDA IFR       ; WAIT FOR ACK PULSE
0314- 29 10     0330            AND #$10
0316- F0 F9     0340            BEQ WAIT2
0318- AD 00 A8  0350            LDA IRB       ; CLEAR CB1 FLAG
031B- 68        0360            PLA '         ; GET BACK PRINT DATA
031C- 8D 01 A8  0370            STA ORA       ; SEND TO PRINTER
031F- A9 0C     0380            LDA #STRBON   ; PULSE DATA STROBE
0321- 8D 0C A8  0390            STA PCR
0324- A9 0E     0400            LDA #STRBOFF
0326- 8D 0C A8  0410            STA PCR
0329- 68        0420            PLA '         ; GET BACK PRINT DATA
032A- 60        0430            RTS
                0440            .EN
```