Technical Notes: Bidirectional Serial-to-Parallel Converter for Data Acquisition and Control

Stuart J. Birrell, University of Missouri
Scott T. Drummond, University of Missouri
Kenneth A. Sudduth, United States Department of Agriculture

Available at: https://works.bepress.com/stuart_birrell/33/
Port on a portable laptop computer. The converter consisted of both hardware conversion circuitry and control software. While it might be possible to write software which would allow for simultaneous data collection over existing serial ports, the only option for acquiring a second stream of serial data is conversion and input through the parallel port.

The objective of this work was to design a system that would allow data transmission to or from a serial device via the parallel port on the computer. The hardware was required to use only +12 VDC and to provide standard RS-232 serial I/O over a range of data formats at a minimum rate of 1200 baud. The system was to gather data in a manner that would be safe to the host computer and allow for simultaneous data collection over existing serial ports.

**HARDWARE AND SOFTWARE**

A computer's parallel port consists of eight output data pins, eight return grounds, and nine other control and status lines (table 1) (Thomas, 1993; Eckhouse, 1990). The use of the parallel data output pins as inputs on a unidirectional parallel port was deemed risky due to their limited current handling ability (Eckhouse, 1990), so other means of input needed to be found. The ideal choices for input lines on the parallel port were the status input lines, due to the fact that these lines have already been well buffered for use as inputs.

The parallel port on IBM-compatible personal computers (PCs) consists of three registers, a data register, a status register, and a control register. The addresses of the various registers are as follows:

<table>
<thead>
<tr>
<th>Data Port Address</th>
<th>= Printer Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Port Address</td>
<td>= Printer Base Address +1</td>
</tr>
<tr>
<td>Control Port Address</td>
<td>= Printer Base Address +2</td>
</tr>
</tbody>
</table>

**ABSTRACT.** A bidirectional serial-to-parallel converter was created to allow collection of serial data through the parallel port on a portable laptop computer. The converter consisted of both hardware conversion circuitry and control software. The converter supported both input and output at a variety of transmission formats and adjustable transmission rates. The control software made use of the various I/O registers of the parallel port to collect data and control the hardware circuitry. **Keywords:** Converter, Computers, Data acquisition.

**Table 1. Parallel pin/ register/bit associations, along with a description of how these bits were used in this application**

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Register</th>
<th>Bit No.</th>
<th>Function in this Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>Control</td>
<td>0</td>
<td>Connect/disconnect data lines</td>
</tr>
<tr>
<td>2-9</td>
<td>Data</td>
<td>0-7</td>
<td>Carry data to the UART data pins</td>
</tr>
<tr>
<td>10</td>
<td>Status</td>
<td>6</td>
<td>Not used</td>
</tr>
<tr>
<td>11*</td>
<td>Status</td>
<td>7</td>
<td>Input bit 3</td>
</tr>
<tr>
<td>12</td>
<td>Status</td>
<td>5</td>
<td>Input bit 2</td>
</tr>
<tr>
<td>13</td>
<td>Status</td>
<td>4</td>
<td>Input bit 1</td>
</tr>
<tr>
<td>14</td>
<td>Control</td>
<td>1</td>
<td>Register Select control of UART</td>
</tr>
<tr>
<td>15</td>
<td>Status</td>
<td>3</td>
<td>Input bit 0</td>
</tr>
<tr>
<td>16*</td>
<td>Control</td>
<td>2</td>
<td>Read/Write control of UART</td>
</tr>
<tr>
<td>17</td>
<td>Control</td>
<td>3</td>
<td>MUX control – select input nibble</td>
</tr>
<tr>
<td>18-25</td>
<td>—</td>
<td>—</td>
<td>Ground – not used</td>
</tr>
</tbody>
</table>

* Pins use reverse logic; i.e., a high value in the parallel port's register implies a low logic voltage on the corresponding pin.
The printer base address was found by searching the printer address table located at RAM location 40:08h (Eckhouse, 1990).

It was decided to use four of the input pins from the status port (pins 11, 12, 13, and 15), and a multiplexer to gather the eight-bit input data. For controlling circuitry, four pins were chosen from the control port (pins 1, 14, 16, and 17). For output of parallel data to the converter, the standard data port (pins 2 through 9) was used. With these I/O definitions, the converter circuit was designed and fabricated (fig. 1).

The design of any serial-to-parallel converter requires the use of a Universal Asynchronous Receiver/Transmitter (UART). The UART chosen for this application was a Hitachi 6850. The eight parallel data pins on the UART were tied both to the input multiplexer and to the output of the parallel port’s data register through tri-state buffers. This allowed the ability to both read from and write to those pins.

Two control lines were connected to the UART’s Register Select pin and its Read/Write pin. These controls were used to select and control operation of the four registers on the UART. The parallel port’s other two control outputs were used to control the input multiplexer and to Connect/Disconnect the parallel port’s data register from the UART’s eight data pins by controlling the tri-state buffers (fig. 1).

The design of any serial-to-parallel converter requires the use of a Universal Asynchronous Receiver/Transmitter (UART). The UART chosen for this application was a Hitachi 6850. The eight parallel data pins on the UART were tied both to the input multiplexer and to the output of the parallel port’s data register through tri-state buffers. This allowed the ability to both read from and write to those pins.

The Receive Serial Input and Transmit Serial Output pins on the UART gathered and sent data in a serial TTL format (0 VDC lo, +5 VDC hi). The RS-232 communications for this application required the data to be transmitted in a +15 VDC lo, -15 VDC hi format. Therefore, it was necessary to use a TTL to RS-232 driver chip. The Harris ICL232CPE chip was chosen to do the conversion, primarily because it could provide the necessary output using only a +5 VDC supply. In addition to the Transmit and Receive lines, all of the handshaking lines on the 6850 UART were passed through the ICL232CPE driver chips for voltage conversion.

Software routines to control the circuit were written in Quickbasic. The example routine in figure 2 is illustrative of circuit control for reception of serial-to-parallel input data. Modifications would be required to adapt it to specific applications.

**CONVERTER PERFORMANCE**

The converter was tested for both transmission and reception errors at a data rate of 1200 baud. Operation of the converter was satisfactory, particularly for the parallel-to-serial output conversion. Our tests detected no errors in this type of transmission. The serial-to-parallel input conversion was not as reliable. The software used a polling loop to continually check for the arrival of a new character in the UART’s receive register. This made the conversion process susceptible to problems involving interrupt requests by the system, since these interrupt requests might cause the polling loop to cycle too slowly and skip over one or more characters. The error rates for this test ranged from 2 to 75 errors/10,000 records. The only type of error detected was that of missing one or more characters from a record.

The converter was used to collect position and time data from an Ashtech M-XII Global Positioning System (GPS) receiver, as part of a system for combine-based yield data collection (Birrell et al., 1996). The host computer acquired GPS records through the converter (at 1200 baud), and yield data records through the serial port. Each yield data record was tagged with the GPS time to allow for recombination of post-processed differential GPS positions at a later date. Since the GPS records were of constant length and sent on second intervals, transmission errors were easily trapped and any problem data was flagged in the data file. The error rates experienced in field tests were

---

**Figure 1—Circuit schematic for bidirectional serial-to-parallel converter.**

---

**List of Parts**

1. 6850 UART
2. 74L5157 QUAD 2-1 MULTIPLEXER
3. 74L5155 QUAD TRI-STATE BUFFER
4. ICL232CPE +5V RS232 DRIVER
5. LM555 TIMER
6. 1.8 MHz OSCILLATOR
7. 7805 +5V VOLTAGE REGULATOR
8. 2 μF CAPACITOR
9. 1 μF CAPACITOR
10. .33 μF CAPACITOR
11. .10 μF CAPACITOR
12. .01 μF CAPACITOR
13. 2.2 kΩ RESISTOR
14. 1 kΩ 15 TURN POT
This routine captures incoming characters and prints them to the screen.

DECLARE SUB findlpt1 (dataport%)

• These bits are the 4 input bits from the parallel status register.
  bit3% = 128: bit2% = 32: bit1% = 16: bit0% = 8

• These constants are used for direct control of the circuit.
  CONN = 1: DCON = 0 'Connect/Disconnect data to UART.
  RSHI = 0: RSLO = 2 'Register select on UART.
  RWHI = 4: RWLO = 0 'Read/Write select on UART.
  MUXHI = 0: MUXLO = 8 'MUX control for selecting input nibble.

findlpt1 (dataport%)
statusport% = dataport% + 1 'Address of status register.
controlport% = dataport% + 2 'Address of control register.

'Find base address of LPT1.
'Address of status register.
'Address of control register.

•Select UART Control Register & connect data port to UART's data port.
OUT controlport%, MUXLO + RSLO + RWLO + CONN
OUT dataport%, 3 'Master Reset of UART.
OUT dataport%, 21
OUT controlport%, MUXLO + RSLO + RWHI + CONN
OUT controlport%, MUXLO + RSLO + RWHI + DCON

WHILE (UCASE$(LEFT$(INKEY$, 1))) <> "Q"
stat% = 0: dl% = 0: dh% = 0
**** Loop until the UART's Receive Data Register is Full. ***

WHILE ((stat% AND bit0%) <> bit0%)
  'While RDRF bit not set,
  'select UART Status &
  'read UART Status Reg.
  stat% = INP(statusport%)
WEND

OUT controlport%, MUXLO + RSHI + RWHI + CONN
a% = INP(statusport%)
OUT controlport%, MUXHI + RSHI + RWHI + DCON
b% = INP(statusport%)

IF (a% AND bit3%) <> bit3% THEN dl% = dl% + 8
IF (a% AND bit2%) = bit2% THEN dl% = dl% + 4
IF (a% AND bit1%) = bit1% THEN dl% = dl% + 2
IF (a% AND bit0%) = bit0% THEN dl% = dl% + 1
IF (b% AND bit3%) = bit3% THEN dh% = dh% + 8
IF (b% AND bit2%) = bit2% THEN dh% = dh% + 4
IF (b% AND bit1%) = bit1% THEN dh% = dh% + 2
IF (b% AND bit0%) = bit0% THEN dh% = dh% + 1

character$ = CHR$(dl% •»• dh% * 16)
PRINT characters;
WEND

'Find base address of LPT1.
SUB findlpt1 (dataport%)
DEF SEG = (&H40)
dataport% = PEEK(&H8) + (PEEK(&H9) * 256)
DEF SEG
END SUB

Figure 2–Sample program for reception of characters through the conversion circuit.

approximately 50 errors/10,000 records transmitted, which was acceptable for our application and comparable with our bench results.

SUMMARY

A bidirectional serial-to-parallel converter was designed and fabricated for flexible data acquisition and control, as a part of a system for combine-based yield data collection. The converter was tested at 1200 baud and performed well in parallel-to-serial output mode. In serial-to-parallel input mode, a significant number of skipped characters occurred. The type and rate of errors suggested that error checking or handshaking could be effective ways to improve performance without substantially changing circuit design.

REFERENCES

