Iowa State University

From the SelectedWorks of Stuart J. Birrell

1995

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/

TECHNICAL NOTES:

BIDIRECTIONAL SERIAL-TO-PARALLEL CONVERTER FOR DATA ACQUISITION AND CONTROL

S. T. Drummond, S. J. Birrell, K. A. Sudduth

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.*

Portable computers are a key component in the process of "on-the-go" data collection and control. Most of these computers are limited in the number and type of output ports they can support. It is not uncommon to find laptops that supply only a single serial and a single parallel port. With recent increases in the use of serially interfaced data acquisition devices, additional serial ports could be required on many systems. For many older laptop computers, the only option for acquiring a second stream of serial data is conversion and input through the parallel port.

There are a number of products on the market that provide bidirectional serial-to-parallel output conversion. These units only provide output conversion from the computer's serial or parallel port to some output device. Serial-to-parallel input conversion for unidirectional parallel ports is not available, and even bidirectional parallel ports would require some sort of control software. While it might be possible to write software which would read input from one of these converters, we felt that the time and expense necessary to fully deduce the operation of such a unit was prohibitive.

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:

Data Port Address = Printer Base Address
Status Port Address = Printer Base Address +1
Control Port Address = Printer Base Address +2

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

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

* Pins use reverse logic; i.e., a high value in the parallel port's register implies a low logic voltage on the corresponding pin.

Article was submitted for publication in March 1995; reviewed and approved for publication by the Information and Electrical Technologies Div. of ASAE in June 1995.

Contribution of the Missouri Agricultural Experiment Station, Journal Series No. 12,296. Trade names are used solely for the purpose of providing specific information. Mention of a trade name, proprietary product, or specific equipment does not constitute a guarantee or warranty by the USDA or the University of Missouri and does not imply the approval of the named product to the exclusion of other products that may be suitable.

The authors are **Scott T. Drummond,** Scientific Programmer/Analyst, Dept. of Agricultural Engineering, **Stuart J. Birrell,** ***ASAE Member,*** Research Assistant Professor, Dept. of Agricultural Engineering, and **Kenneth A. Sudduth,** ***ASAE Member,*** Agricultural Engineer, Cropping Systems and Water Quality Research Unit, USDA-Agricultural Research Service, University of Missouri, Columbia. **Corresponding author:** Kenneth A. Sudduth, Agricultural Engineer, Cropping Systems and Water Quality Research Unit, USDA-Agricultural Research Service, Agricultural Engineering Building, University of Missouri, Columbia, MO 65211; e-mail: <Ken_Sudduth@muccmail.missouri.edu>.

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 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

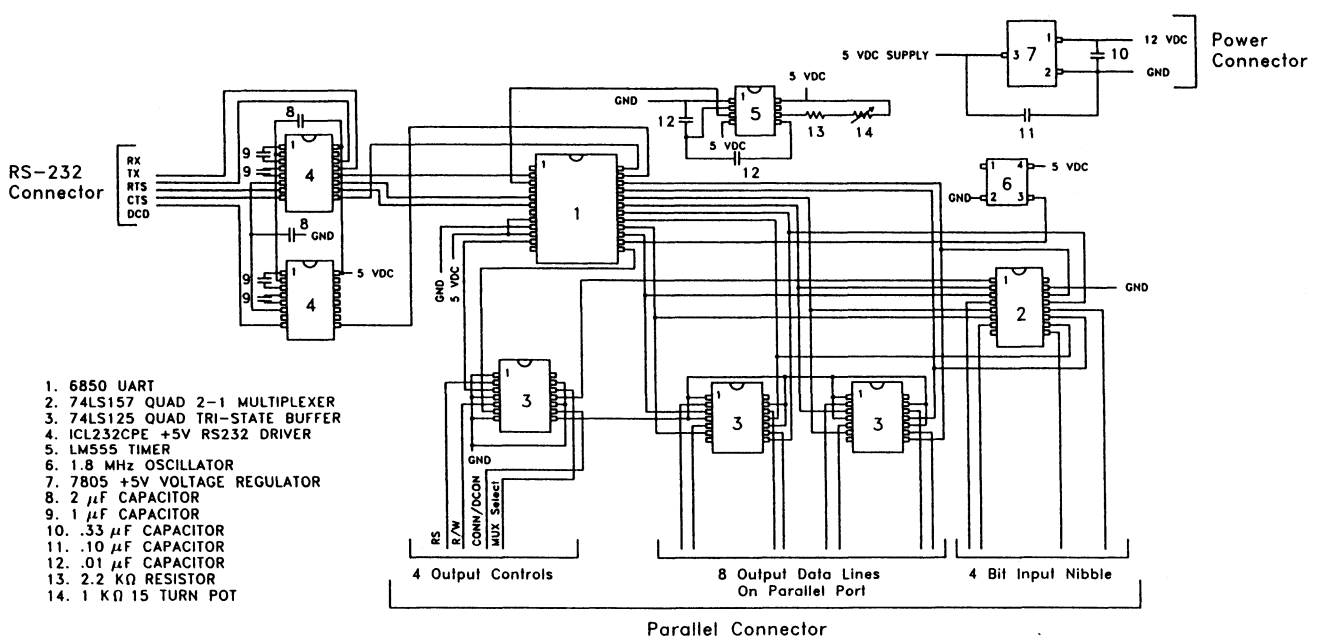

Figure 1–Circuit schematic for bidirectional serial-to-parallel converter.

```
'*********************************************************************
'    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.
' They affect control of the parallel control register.
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%)                'Find base address of LPT1.
statusport% = dataport% + 1         'Address of status register.
controlport% = dataport% + 2        '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                                 'Set UART comm. parameters.
OUT controlport%, MUXLO + RSLO + RWHI + CONN      'Select UART status register,
OUT controlport%, MUXLO + RSLO + RWHI + DCON      '& disconnect from data.

WHILE (UCASE$(LEFT$(INKEY$, 1))) <> "Q"           'Until a "Q" is pressed...
 stat% = 0: dl% = 0: dh% = 0
 '**** Loop until the UART's Receive Data Register is Full. ***
 WHILE ((stat% AND bit0%) <> bit0%)               'While RDRF bit not set,
  OUT controlport%, MUXLO + RSLO + RWHI + DCON    'select UART Status &
  stat% = INP(statusport%)                        'read UART Status Reg.
 WEND
 OUT controlport%, MUXLO + RSHI + RWHI + DCON     'Select UART Data Reg,
 a% = INP(statusport%)                            'and read low nibble.
 OUT controlport%, MUXHI + RSHI + RWHI + DCON     'Select UART Data Reg
 b% = INP(statusport%)                            'and read high nibble.
 IF (a% AND bit3%) <> bit3% THEN dl% = dl% + 8
 IF (a% AND bit2%) = bit2% THEN dl% = dl% + 4     'Convert the two nibbles
 IF (a% AND bit1%) = bit1% THEN dl% = dl% + 2     'into an ascii character
 IF (a% AND bit0%) = bit0% THEN dl% = dl% + 1     'recalling that bit3% is
 IF (b% AND bit3%) <> bit3% THEN dh% = dh% + 8    'an inverted pin.
 IF (b% AND bit2%) = bit2% THEN dh% = dh% + 4
 IF (b% AND bit1%) = bit1% THEN dh% = dh% + 2
 IF (b% AND bit0%) = bit0% THEN dh% = dh% + 1
 character$ = CHR$(dl% + dh% * 16)
 PRINT character$;
WEND
END

'This routine finds the printer base address for 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

Birrell, S. J., K. A. Sudduth and S. C. Borgelt. 1996. Comparison of sensors and techniques for crop yield mapping. *Computers and Electronics in Agriculture* (In press).

Eckhouse, R. H. 1990. Convert PC's parallel port into bidirectional instrumentation interface. *Personal Engineering and Instrumentation News* 7(6):51-56.

Thomas, B. 1993. Inexpensive parallel-port interface controls power-strip outlets. *Personal Engineering and Instrumentation News* 10(10):50-55.