Int. J. Engng Ed. Vol. 10, No. 5, pp. 450–458, 1994
Printed in Great Britain.

0742-0269/91 $3.00+0.00

Centronics Port Drivers for Computer-Aided Laboratory Experiments*

KIM I. MALLALIEU
RONALD D. ARIEATAS
DERRICK SO'BRIEN

Department of Electrical and Computer Engineering, University of the West Indies, St Augustine, Trinidad, West Indies

There is an increasing use of computers in engineering education for laboratory automation. The selection of products has become a difficult process. Which hardware platforms to support? Which interfacing and software products to purchase? At the University of the West Indies, we have implemented an inexpensive computer-based data acquisition and control teaching platform. Later investigations have shown that several applications involving digital computer interfacing are adequately supported by on-board Centronics port on IBM PCs at virtually no cost. This paper is a guide for engineering laboratory educators interested in minimal-cost solutions to digital interfacing in computer-aided laboratory experiments (CALE).

INTRODUCTION

AT THE instrumentation laboratory of the University of the West Indies, we are interested in inexpensive computer-based data acquisition and control tools to support teaching electronic instrumentation.

Generally, instrumentation and control systems comprise a set of sensors, a set of actuators and some overall scheme for supervising data acquisition and control. In laboratory exercises it may be necessary to perform these data acquisition and control functions at large numbers of laboratory stations.

Although the PC is a popular choice for computer-based teaching laboratories, we have found that the cost of purchasing commercial data acquisition cards for each laboratory station is not justifiable. In an effort to find alternative means for a number of laboratory stations to communicate with external devices, we have implemented model engineering systems furnished with sensors and actuators which may be interrogated from the PC-based laboratory stations via inexpensively assembled data acquisition boards for analogue and digital input and output [1].

PCs are sold with at least one Centronics (parallel) port for communication with a printer. We have found that a range of common laboratory data acquisition and control applications can be performed using this standard port. In this manner, virtually no expense is incurred in implementing digital data acquisition and control from the PC.

The on-board Centronics port is particularly well suited to simple computer-aided laboratory experiments (CALE) which use the PC for digital interfacing. First-generation parallel ports provided 12 output lines and five input lines which could be used for direct connection to digital sensors and actuators. More recently as many as 17 lines may be configured for input [2]. Using the simple functions documented in this paper, PC-based laboratory exercises may be designed for digital data acquisition and control. We have found that the routines have been particularly helpful in final year projects for undergraduates.

CHARACTERISTICS OF LABORATORY-BASED TEACHING

Teaching by means of CALE represents a significant departure from traditional board and chalk teaching. In the former, changing emphases in course work may require changes in laboratory configurations and equipment which can be very costly. It is important, therefore, that each generation of laboratory tools be affordable, general and flexible.

The computers used in the CALEs may be used throughout the day for a variety of applications that are unrelated to laboratory exercises (laboratory administration, word processing, simulation exercises, on-line tutorials, data communications, etc.). The demand for data acquisition and control facilities is, on the other hand, often sporadic: several computers are required for data acquisition exercises at the same time for half of the day while none are required for the other half of the day. In some cases equipment may be used for one half day session per week—or worse—a few half day

* Paper accepted 2 June 1994.

sessions per term or year. It is especially important that this laboratory material be affordable.

THE CENTRONICS PORT

The Centronics port is a parallel interface utilizing a 25 pin D shell connector with pins wired for input or output. The interface signals are TTL compatible.

The Centronics port is memory mapped so that it can be accessed directly from software. The base address of the installed parallel ports on the PC are held at segment 0040 (hex) and offset 0008 (hex) + 2 × port where port = 0, 1, 2 for LPT1, LPT2 and LPT3, respectively. Three contiguous bytes, located at the base address, the base address + 1 and the base address + 2, are dedicated to each port. Within these three bytes, particular bits may be set and read to control and interrogate pins on the respective parallel port.

On the original Centronics ports, 12 bits may be set and five bits may be read to control and interrogate external logic levels. Some of these bits obey negative logic while others obey positive logic as indicated in the mapping of bits to pins shown in Table 1. Ports with bidirectional pins which offer additional input lines [2] are not needed for our laboratory experiments but applications and usage have been documented elsewhere [3].

HIGH-LEVEL LANGUAGE PARALLEL PORT DRIVER

For data acquisition and control, it is convenient to address the parallel port using high-level language calls. We use simple drivers written in Basic and 'C' to communicate with the parallel port. Our driver routines transparently handle the logic sense of each bit so that all appear to be positive logic. Example functions include the following.

1. get_pport_addr:
   returns address of installed parallel port.
2. write_byte2pport:
   writes a byte to the parallel port.
3. write_nybble2pport:
   writes a nybble to parallel port.
4. set_pport_line:
   sets or disasserts any of parallel port lines.
5. read_5pport:
   reads five input lines of parallel port.
6. read_any_pport:
   reads any of five input lines of parallel port.

Source code in 'C' (Quick C compiler) illustrates how these functions may be implemented.

1. unsigned far get_pport_addr(int portnum)
   {
   unsigned far *p;
   FP_SEG(p) = 0×0040;
   FP_OFF(p) = 0×0008 + 2*port;
   return(*p);
   }

   This function returns the address of the installed parallel port. It expects the number of the port, portnum, whose address is sought. To use, include dos.h.

2. void write_byte2pport(unsigned base_addr, int databyte)
   {
   outp(base_addr, databyte);
   }

   To write a byte to a parallel port at the base address, a single call to the 'C' function, outp, may be used. Get_pport_addr can be used to return the base address, base_addr, of the installed port. As all bits at the base address are positive logic, no logic inversions are necessary and the data, databyte (dd hex), may be written directly to the port. To use, include conio.h.

3. write_nybble2pport(unsigned base_addr, int data_nybble)
   {
   output(base_addr+2, data_nybble ∧ 0×0B);
   }

   Four bits may be written to the parallel port, as indicated, using a single call to the 'C' function,

Table 1. Definition of pins on the Centronics port

Twelve outputs				Five inputs			
Pin	Logic	Address	Bit	Pin	Logic	Address	Bit
1	Negative	Base address + 2	1 (LSB)	10	Positive	Base address + 1	7
2	Positive	Base address	1 (LSB)	11	Negative	Base address + 1	8
3	Positive	Base address	2	12	Positive	Base address + 1	6
4	Positive	Base address	3	13	Positive	Base address + 1	5
5	Positive	Base address	4	15	Positive	Base address + 1	4
6	Positive	Base address	5				
7	Positive	Base address	6				
8	Positive	Base address	7				
9	Positive	Base address	8 (MSB)				
14	Negative	Base address + 2	2				
16	Positive	Base address + 2	3				
17	Negative	Base address + 2	4				

outp. Get_pport_addr can be used to return the base address, base_addr, of the installed port. The data nybble to be sent (data_nybble) to pins 1, 14, 16 and 17 (Table 1) is exclusive 0Red (∧0×0B) with the hex number 0B to normalize all pins to positive logic. To use, include conio.h.

4.
```
unsigned short read_5pport(int base_addr)
{
unsigned short data;
data = inp(base_addr+1);
return (((data>>3)∧0×10) & 0×1F);
}
```

The eight bits at the base address + 1 are read as an unsigned short integer. As the data appears on bits 4, 5, 6, 7 and 8, it is rotated to the right so that the least significant bit appears in bit position 1 (data>>3). The result is exclusive 0Red with 10 hex to normalize all pins to positive logic (∧0×10) and ANDed with 1F hex (& 0×1F) to force the highest three bits to 0.

As before, get_pport_addr can be used to return the base address of the Centronics port.

5.
```
void set_pport_line(int p, int base_addr)
{
switch(p){
case 1:
outp (base_addr+2, (inp(base_addr+2) & 0×FE));
break;

case 2:
outp (base_addr, (inp(base_addr) | 0×01));
break;

case 3:
outp (base_addr, (inp(base_addr) | 0×02));
break;

case 4:
outp (base_addr, (inp(base_addr) | 0×04));
break;

case 5:
outp (base_addr, (inp(base_addr) | 0×08));
break;

case 6:
outp (base_addr, (inp(base_addr) | 0×10));
break;

case 7:
outp (base_addr, (inp(base_addr) | 0×20));
break;

case 8:
outp (base_addr, (inp(base_addr) | 0×40));
break;

case 9:
outp (base_addr, (inp(base_addr) | 0×80));
break;

case 14:
outp ((base_addr+2), (inp(base_addr+2) & 0×FD));
break;

case 16:
outp ((base_addr+2), (inp(base_addr+2) | 0×04));
break;

case 17:
outp ((base_addr+2), (inp(base_addr+2) | 0×F7));
break;

default:
printf ('Pin %d is not an output \n', p);
break;
    }
}
```

This function sets (asserts) any of the 12 output pins of the Centronics port. It requires the base address of the port, base_addr and the number of the pin, p, to set. It preserves the state of all other output pins by first reading the current state of the port and rewriting all pin values except the one to be set. The pin to be set is forced high. To use, include conio.h.

6.
```
int read_any_pport(int base_addr, int p)
{
unsigned in5bits;
in5bits = ((inp(base_addr+1) ∧ 129) >> 3) & 0×1F;

switch(p){
case 15:
return (in5bits & 0×01);
break;

case 13:
return ((in5bits & 0×02)/2);
break;

case 12:
return ((in5bits & 0×04)/4);
break;

case 10:
return ((in5bits & 0×08)/8);
break;

case 11:
return ((in5bits & 0×10)/16);
break;

default:
printf ('Pin %d is not an input \n', p);
break;
    }
}
```

This function returns the value of any of the valid input pins: 10, 11, 12, 13 or 15. It requires

the pin number, p and the base address, base_addr, of the parallel port. Read_any_pport normalizes logic sense differences in the pins so that they all indicate positive logic. To use, include conio.h.

CHARACTERIZATION OF THE CENTRONICS PORT

It may be necessary to estimate the read and write speeds of the Centronics port. For our own purposes, test programs were run on a range of machines to compare the execution times of a null loop against the execution times of a loop with a read instruction and one with a write instruction. An example of a 'C' null loop is

```
for (i=0;i<x;i++)
```

An example of a read loop is:

```
for (i=0;i<x;i++)
inp(889);
```

An example of a write loop is:

```
for (i=0;i<x;i++)
outp(888,0);
```

Timing of the test loops was achieved by using the on-board time-of-day clock on the PC. This clock is accessible via interrupts [4]. The smallest increment of the clock, the 'tick', is zeroed at midnight and thereafter occurs every 1/18.2 of a second. An estimate for the maximum rate (per second) at which the port may be accessed was determined by counting the number of instructions executed in a single tick period and multiplying by 18.2.

Though execution times are expected to differ from machine to machine and in general from compiler to compiler, the results were obtained using Microsoft QuickC on the machines in our laboratory. They yield a rough idea of the input/output capabilities of the Centronics port as indicated in Table 2.

These figures were derived using ten ticks, therefore they are subject to a ±11% uncertainty. They represent example execution times for the instructions inp and outp alone. In practical applications input data is often stored in an array, displayed or used in calculations. A speed penalty for these additional instructions must be anticipated.

Table 2. Sample of Centronics I/O speeds as measured on several PCs

Machine	Processor speed (MHz)	Max input speed (approximate) (Kbytes/s)	Max output speed (approximate) (Kbytes/s)
386	33	292	300
386 SX	16	150	150
286	12	91	91
XT	12	27	29

APPLICATIONS

Data acquisition and control drivers with graphical user interfaces (GUIs) are useful in laboratory teaching applications as they provide the student with immediate access to controlling and reading external hardware devices. Little effort is therefore invested in learning to use the laboratory tools and more attention can be paid to understanding course material. We have written a Centronics port driver to suit these applications.

Where there is no need for a graphical user interface or where GUI development tools are not available, a number of useful applications of the Centronics port still exist. Example test and measurement tools include a logic probe, a bus analyser and a digital system development platform. These applications transform the PC into a universal computer-based laboratory test and measurement instrument for modest digital systems. In addition to its use in test and measurement tools, our experience has shown that parallel port drivers are popular and convenient in final year projects.

Labwindows parallel port driver

Software suites which offer data acquisition, analysis, presentation and control in an integrated environment are becoming standard laboratory tools. These packages typically have a graphical user interface and come bundled with drivers for plug-in data acquisition boards which communicate with external intruments. Simple driver routines may be written to communicate with the on-board parallel port on IBM PCs so that these GUI, analysis and presentation tools may be used without the need for purchasing data acquisition cards.

A Centronics port driver has been written for Labwindows (National Instruments), a popular instrument control and data analysis software package. The instrument driver may be used to graphically set or read any or all of the allowable pins in the Centronics port.

Figure 1 illustrates the data acquisition front panel of the Labwindows Centronics port driver. The port is selected (1 for LPT1, 2 for LPT2 and 3 for LPT3), the sample rate is selected (low, medium, high) and the data acquisition is initiated by the 'start' push button control. The graph shows five traces, one for each input pin. Data acquisition can be stopped at any time by selecting the 'stop' push button control on the front panel. For more data to be read and displayed, the 'start' button may be selected again. The screen can be printed ('print' button) and the user can quit the application ('quit' button).

For Fig. 1, the Centronics port of a PC was connected to a printer. During data acquisition, the

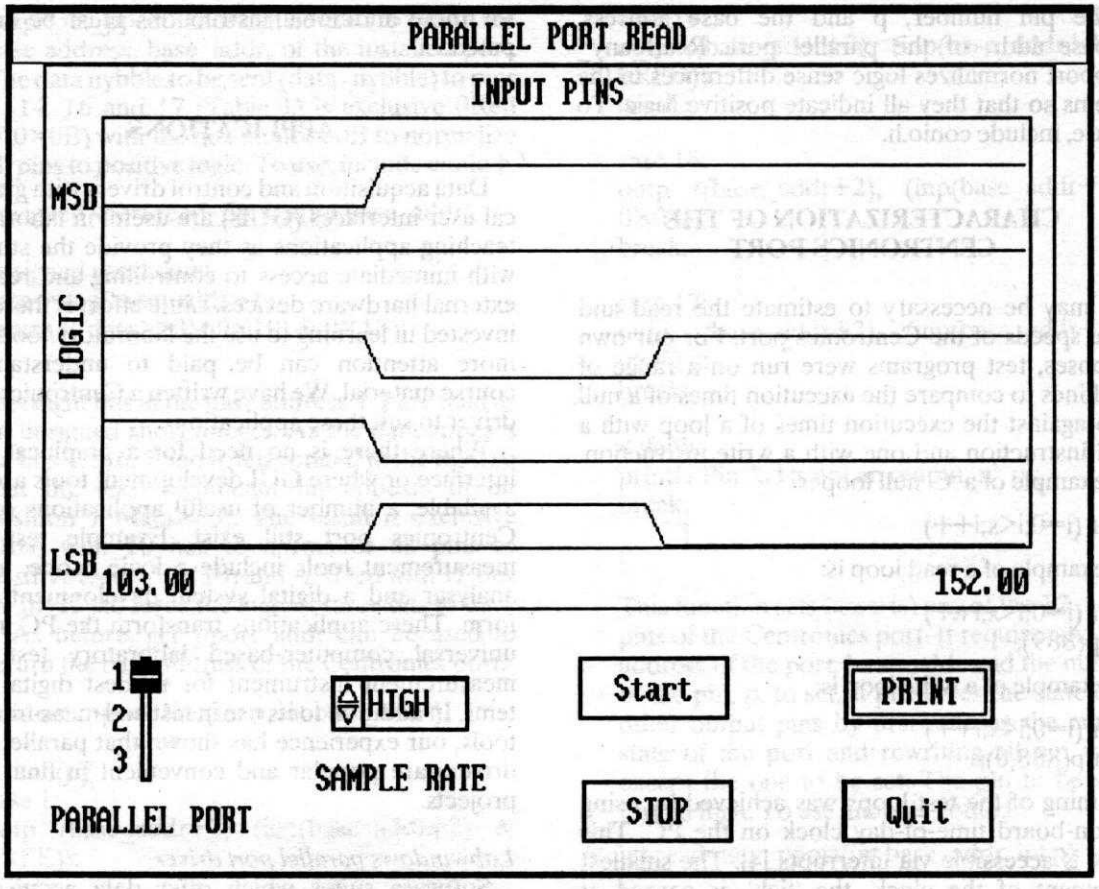

Fig. 1. Data acquisition front panel of Labwindows Centronics port driver.

printer was turned on and the trace stopped to indicate the changing levels on the input pins. The changes represent control signals sent to the PC from the printer on power up. In general, pins 10, 11, 12, 13 and 15 may be connected to a digital system under test in order to continuously monitor logic states.

Figure 2 illustrates the control front panel of the Labwindows Centronics port driver. As indicated, single shot or continuous digital control of the port is possible with the driver. For single shot applications, the operator uses a mouse to select a bit pattern for the 12 output pins of pattern 1. The 'single shot' is selected and the 'go' button pressed when ready.

For continuous output, the following are possible:

1. external trigger;
2. internal trigger.

While plug-in cards typically have their own on-board clock, timing utilities using the Centronics port are constrained to make use of either an external clock or the computer's on-board clock. The external trigger is derived from one of the five available input pins to the Centronics port. If internal triggering is selected, port writes can be controlled in software to be performed at a low, medium or high rate. The high rate is, of course, limited to the maximum write rate of the machine, as indicated in Table 2.

The continuus write option requires two 12 bit patterns, pattern 1 and pattern 2. These two patterns are transmitted consecutively starting when the 'go' button is selected and halting when the 'stop' button is selected. The user can print the screen or quit the application by selecting the appropriate push buttons on the control panel.

Parallel port applications executable from the Labwindows graphical user interface may be easily developed and customized using the tools provided by the package and the parallel port driver routines documented in this paper.

Bus analyser

Graphics libraries, including the Intrinsic Microsoft graphics library, may be used to display logic signals read from the Centronics port. Figure 3 illustrates IEEE-488 signals during a communications test between a PC and a digitizing oscilloscope. The hand-shaking lines, DAC, NRFD and DAV are displayed together with the interface management lines ATN and EOI.

This inexpensive application of the Centronics port can be used for teaching simple data communications protocols and to monitor bus signals

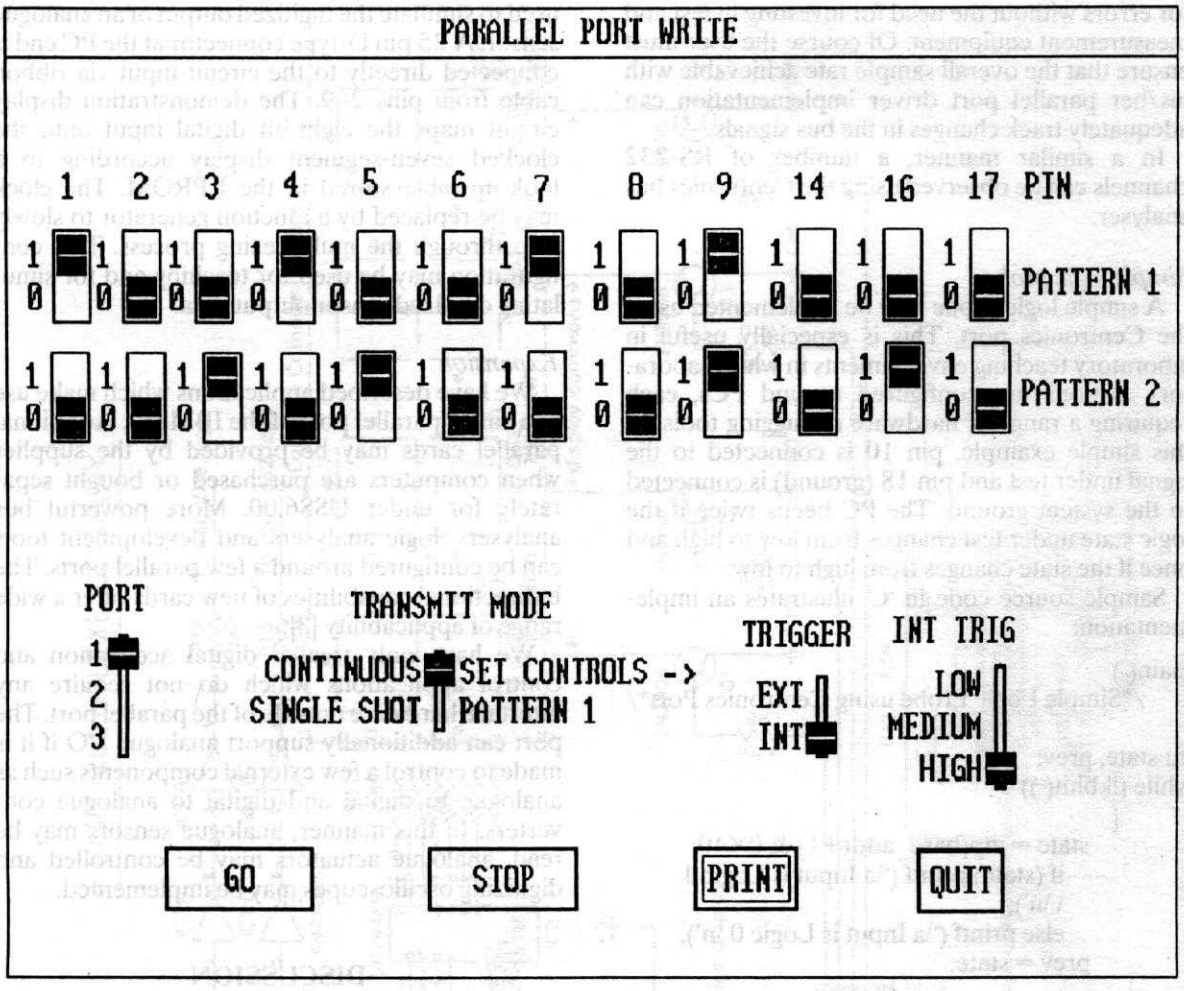

Fig. 2. Control front panel of Labwindows Centronics port driver.

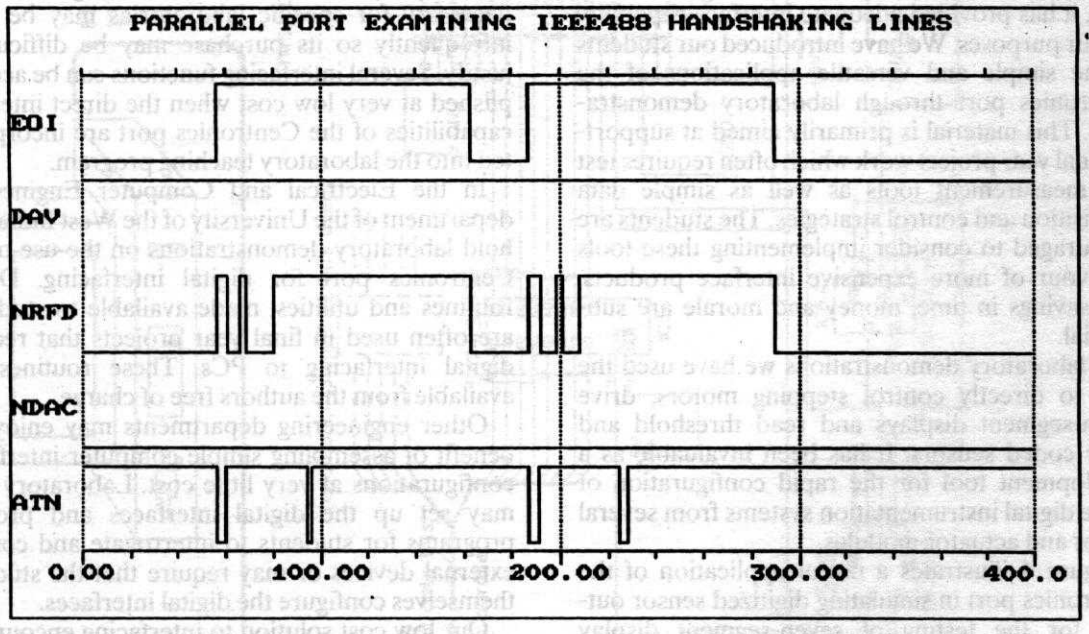

Fig. 3. IEEE-488 hand-shaking signals acquired via the Centronics port.

for errors without the need for investing in test and measurement equipment. Of course the user must ensure that the overall sample rate achievable with his/her parallel port driver implementation can adequately track changes in the bus signals.

In a similar manner, a number of RS-232 channels can be observed using the Centronics bus analyser.

Simple logic probe

A simple logic probe may be implemented using the Centronics port. This is especially useful in laboratory teaching environments in which laboratory stations are configured around PCs, each requiring a range of hardware debugging tools. In this simple example, pin 10 is connected to the signal under test and pin 18 (ground) is connected to the system ground. The PC beeps twice if the logic state under test changes from low to high and once if the state changes from high to low.

Sample source code in 'C' illustrates an implementation:

```
main( )
    /*Simple Logic Probe using Centronics Port*/
{
int state, prev;
while (!kbhit( ))
        {
            state = inp(base_addr+1) & 0×40;
              if (state) printf ('\a Input is Logic 1
              \ \n');
              else printf ('\a Input is Logic 0 \n');
            prev = state;
        }
}
```

Use of Centronics port in engineering education

Though we have only used the Centronics port in applications which read five lines and write to 12 lines, it has provided adequate interface capability for our purposes. We have introduced our students to the simple and versatile applications of the Centronics port through laboratory demonstrations. This material is primarily aimed at supporting final year project work which often requires test and measurement tools as well as simple data acquisition and control strategies. The students are encouraged to consider implementing these tools in favour of more expensive interface products. The savings in time, money and morale are substantial.

In laboratory demonstrations we have used the port to directly control stepping motors, drive seven-segment displays and read threshold and pulse coded sensors. It has been invaluable as a development tool for the rapid configuration of entire digital instrumentation systems from several sensor and actuator modules.

Figure 4 illustrates a useful application of the Centronics port in simulating digitized sensor outputs for the testing of seven-segment display systems. In a laboratory demonstration, a PC is used to simulate the digitized output of an analogue sensor. A 25 pin D-type connector at the PC end is connected directly to the circuit input via ribbon cable from pins 2–9. The demonstration display circuit maps the eight-bit digital input onto the clocked seven-segment display according to a look-up table stored in the EPROM. The clock may be replaced by a function generator to slowly step through the multiplexing process. This configuration may be used for teaching and for simulating digitized sensor output data.

Expansion

We have described applications which make use of a single parallel port of the IBM PC. Additional parallel cards may be provided by the supplier when computers are purchased or bought separately for under US$6.00. More powerful bus analysers, logic analysers and development tools can be configured around a few parallel ports. The bidirectional capabilities of new cards offer a wide range of applicability [3].

We have only treated digital acquisition and control applications which do not require any interface hardware outside of the parallel port. The port can additionally support analogue I/O if it is made to control a few external components such as analogue to digital and digital to analogue converters. In this manner, analogue sensors may be read, analogue actuators may be controlled and digitizing oscilloscopes may be implemented.

DISCUSSION

PCs have become popular in laboratory teaching environments [4–6] as data acquired from external devices may be processed, analysed and displayed with ease. Output devices may be manipulated under program control. Interfacing hardware necessary for specific laboratories may be used infrequently so its purchase may be difficult to justify. Several interfacing functions can be accomplished at very low cost when the direct interface capabilities of the Centronics port are incorporated into the laboratory teaching program.

In the Electrical and Computer Engineering department of the University of the West Indies, we hold laboratory demonstrations on the use of the Centronics port for digital interfacing. Driver routines and utilities, made available to students, are often used in final year projects that require digital interfacing to PCs. These routines are available from the authors free of charge.

Other engineering departments may enjoy the benefit of assembling simple computer-interfaced configurations at very little cost. Laboratory staff may set up the digital interfaces and provide programs for students to interrogate and control external devices or may require that the students themselves configure the digital interfaces.

Our low cost solution to interfacing encourages the development of courseware design for

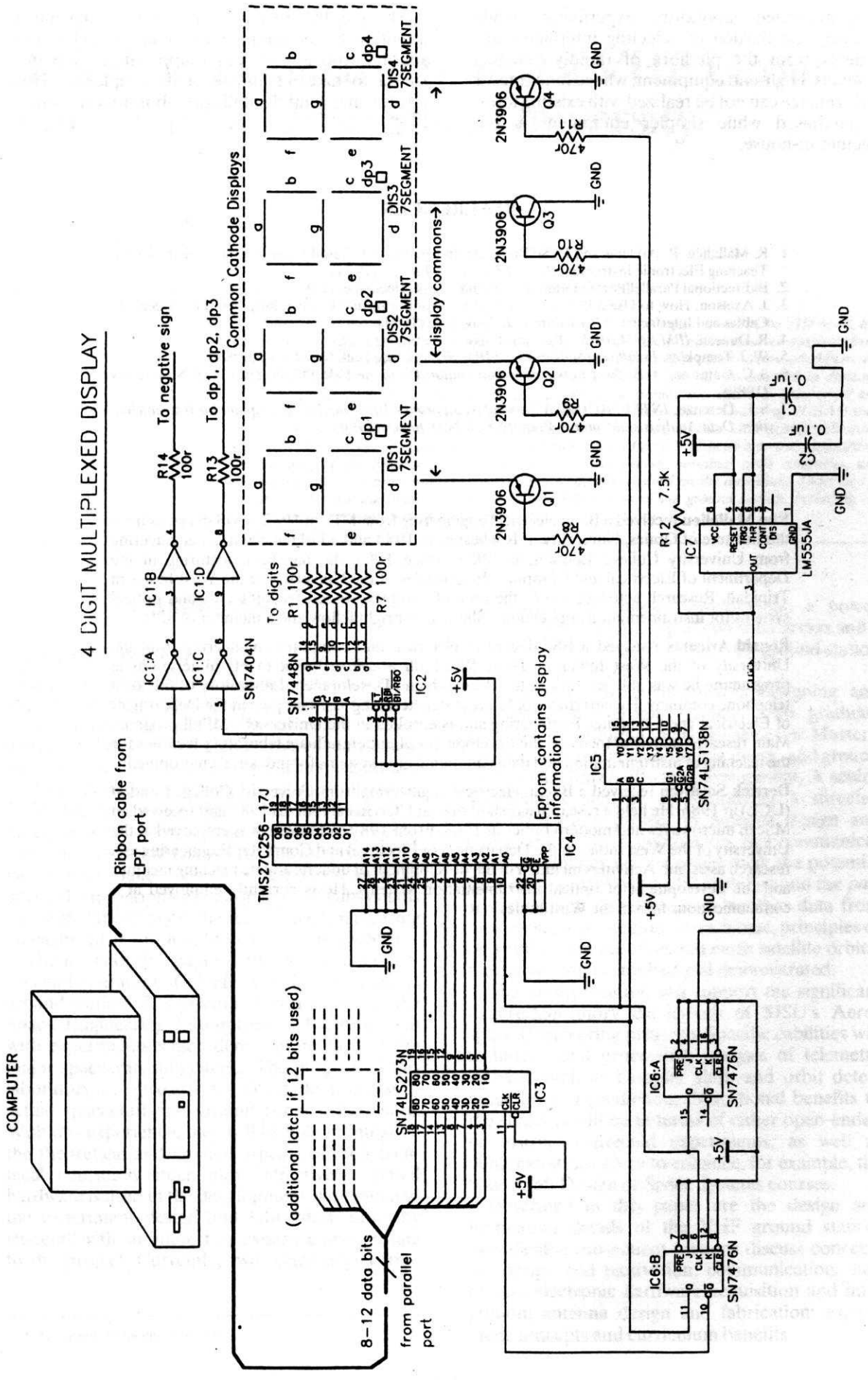

Fig. 4. PC communicating with digital system through the Centronics port.

computer-aided laboratory experiments while relieving the burden of selecting interface components from the plethora of rapidly-changing products. High-end equipment, whose functions or performance can not be realized with existing tools, is purchased while simpler equipment is put together in-house.

The parallel port is currently receiving much attention for communication, control and acquisition purposes. No other application is better suited to take advantage of its simplicity, cost-effectiveness and flexibility as laboratory training.

REFERENCES

1. K. Mallalieu, R. Arieatas and D. So'Brien, An Inexpensive PC-based Laboratory Configuration for Teaching Electronic Instrumentation, *IEEE Trans. Ed.*, **37** (1994).
2. Bidirectional Parallel Port Operation, *Computer Craft*, December (1992).
3. J. Axelson, How to Use a PC's Parallel Port for Monitoring and Control Purposes, Part 1: Signals, Cables and Interfacing', *Microcomput. J.*, May/June (1994).
4. R. Duncan, *IBM ROM BIOS*, Microsoft Press, Washington (1988).
5. W. J. Tompkins, *Interfacing Sensors to the IBM PC*, Prentice Hall, New Jersey (1988).
6. S. C. Gates and J. Backer, *Laboratory Automation Using the IBM PC*, Prentice Hall, New Jersey (1989).
7. S. E. Derenzo, *INTERFACING A Laboratory Approach Using the Microcomputer for Instrumentation, Data Analysis and Control*, Prentice Hall, New Jersey (1990).

Kim Mallalieu received a BS in electrical engineering from MIT in 1982, a MS in optics from the Institute of Optics, University of Rochester, in 1983 and a PhD in electrical engineering from University College London, in 1987. Since 1987 she has been lecturing in the Department of Electrical and Computer Engineering at the University of the West Indies in Trinidad. Research activities are in the area of computer-based teaching aids and optical systems for instrumentation applications. She is a Fulbright fellow and a member of SPIE.

Ronald Arieatas received a BSc degree in electrical and computer engineering from the University of the West Indies in 1989. While on a course break in his undergraduate programme he was able to work in the Research and Development Laboratory of the local telephone company. He currently holds a research/teaching assistant post in the Department of Electrical and Computer Engineering and is enrolled in the University's MPhil program. Main research activities focus on the development of undergraduate laboratory facilities for the teaching of instrumentation and data communication systems for industrial environments.

Derrick So'Brien received a BSc in electrical engineering from University College London (UCL) in 1986. He held a research assistant post at UCL from 1986 to 1987 and received an MSc in microwaves and modern optics in 1988. From 1989 to 1992 he was employed at the University of the West Indies in the Department of Electrical and Computer Engineering as a research assistant. Activities included in the development of undergraduate teaching facilities and the development of optical instrumentation systems. He is currently employed at a communications firm in the West Indies.