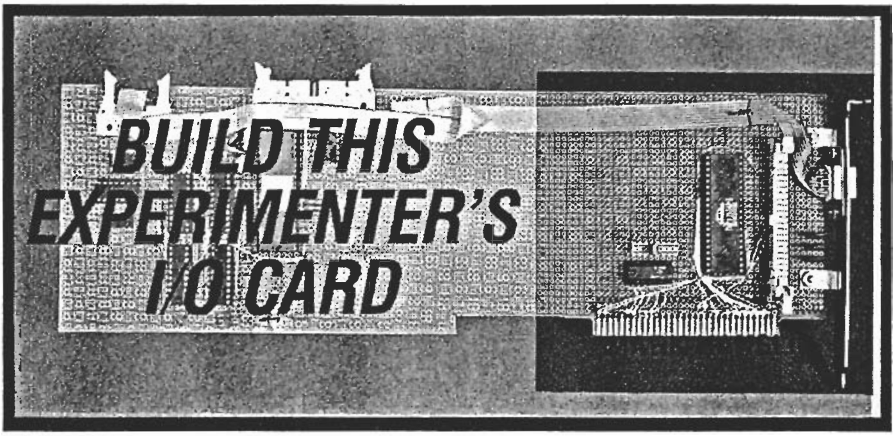
BUILD THIS EXPERIMENTER'S I/O CARD



A computer by itself can't do much; it needs some way of communicating with the outside world. It needs to be able to sense external conditions (a switch closure, for example), and it needs to be able to control circuitry (a relay, for example). The principles of interfacing those types of devices are not difficult; we'll show how easy it is by building an experimenter's card for the IBM PC expansion bus.

The card contains three eight-bit parallel ports, but is built from just a few components, thereby making construction simple and inexpensive. We'll describe several circuits for interfacing LED's, switches, and other devices to the card, as well the software required to configure and use the I/O ports. We'll also show you how easy it is to set up and use the card with simple BASIC programs.

The 8255 PPI

The heart of the design is the 8255 Programmable Peripheral Interface, or PPI. The 8255 was originally designed for use with the 8080 microprocessor, but it is also used with 8088 designs including the PC family.

The 8255 has three eight-bit TTL-compatible I/O ports (A–C), and it can operate in three different modes. Depending on the mode, the lines in each port act differently.

In Mode 0, Ports A and B can operate as either inputs or outputs, and Port C is divided into two four-bit groups, either of which can operate as inputs or outputs.

In Mode 1, Ports A and B can again act as either inputs or outputs. However, the two four-bit ports in Port C are used for handshaking and control purposes in conjunction with Ports A and B. In Mode 1, the Port C lines might be used to strobe data (supplied on either Port A or port B) into a printer, and to detect its "busy" signal.

Last, in Mode 2, Port A is used for eight-bit bidirectional bus I/O, Port C is used for control and status information, and Port B is not used at all. For further details on operating modes, consult Intel's *Microsystem Components Handbook*, Volume 2.

You select among the various modes by writing a value to a special control port; Table 1 shows the control-port values required to achieve various I/O combinations. Our examples all work in Mode 0.

The PC Interface

With Intel microprocessors, communications between the CPU and various devices is accomplished through I/O (Input/

(Continued on page 75)

I/O CARD

continued from page 73

Output) ports. Just as each house on a street has its own address, each piece of hardware connected to an Intel processor has its own port address. For example, serial port COM1 is located at address 03F8h. IBM's Technical Reference Manuals list the specific port addresses associated with specific pieces of hardware.

Our project uses 32 port addresses between 0200h and 02FFh. In order to avoid conflict with other devices, those 32 addresses can start at one of eight locations in that range; you select the desired starting address via a jumper block, as shown in Table 2. Both hex and decimal values are shown; if you're programming in BASIC, you'll probably find the decimal values useful.

As shown in Fig. 1, the address ranges are decoded by IC2, a 74LS138 demultiplexer. The 74LS138 takes three inputs and decodes the various combinations thereof into eight exclusive outputs. The IC also has one active-high (G1) and two active-low ($\overline{G2A}$ and $\overline{G2B}$) enable inputs.

TABLE 1—8255 PORT CONFIGURATION

Control Word		Port		
Hex	Decimal	A	B	C
80	128	Out	Out	Out
82	130	Out	In	Out
85	133	Out	Out	In
87	135	Out	In	In
88	136	In	Out	Out
8A	138	In	In	Out
8C	140	In	Out	In
8F	143	In	In	In

TABLE 2—JUMPER POSITIONS AND PORT ADDRESSES

	Address	
Position	Hex	Decimal
1	200	512
2	220	544
3	240	576
4	260	608
5	280	640
6	2A0	672
7	2C0	704
8	2E0	736

Address lines A8 and A9 drive the control inputs, along with AEN (Address Enable), which is low when the microprocessor can access the expansion bus. When A8 and AEN are low and A9 is high, IC2 will decode address lines A5–A7, providing a single active-low output. In that way, the 256-byte page of I/O space beginning at 0200h is divided into eight 32-byte chunks. The eight outputs of IC2 are brought to the jumper block, which passes one enable signal on to the 8255.

The 8255 itself has only 4 ports. Port A is always at the base address, port B is at base + 1, port C is at base + 2, and the control port is at base + 3. Lines A0 and A1 select which port is addressed, and RD and WR determine whether data is read or written, respectively.

For example, if you short jumper position three, the base address would be 0240h, so you would access Port A at 0240h, Port B at 0241h, Port C at 0242h, and the control port at 0243h.

Construction

The circuit is built on a standard prototyping card for the 8-bit IBM PC bus. All required parts are standard items that can be obtained from most mail-order suppliers. Component placement isn't critical, but lead lengths should be minimized. (See Fig. 2). To avoid damage dur-

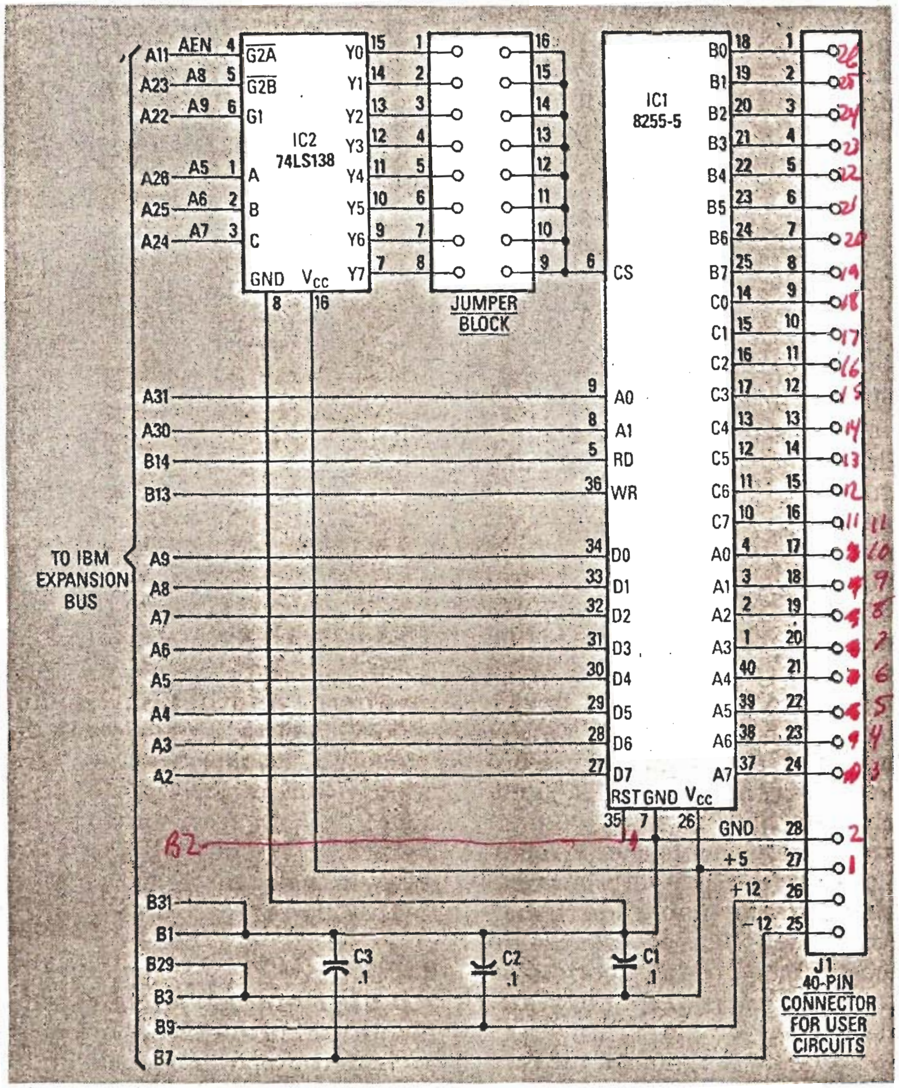


FIG. 1—ADD THREE 8-BIT PARALLEL PORTS for I/O experiments using this simple circuit. The jumper block lets you assign port addresses from 0200h to 02FFh.

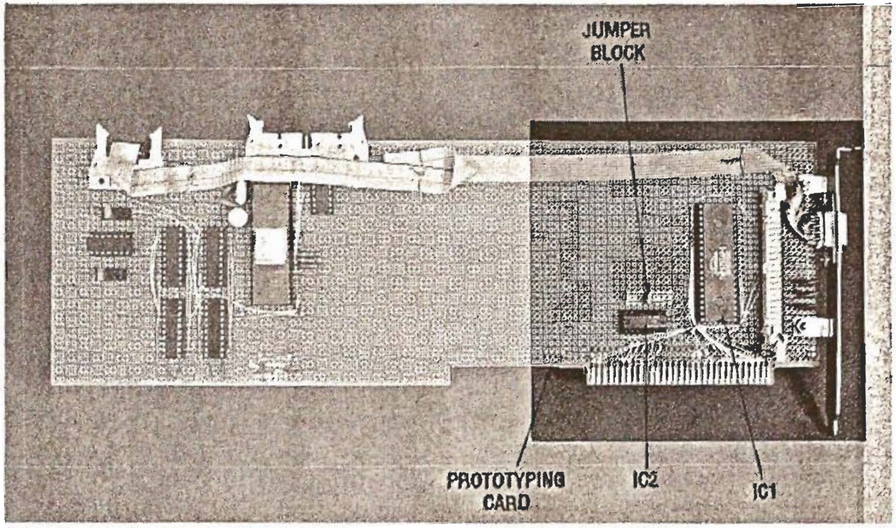


FIG. 2—THE AUTHOR'S PROTOTYPE was used to burn EPROM's, control a model railroad setup, and more.

ing construction, it's best to use sockets for all IC's. Neither IC used in this project is particularly sensitive to static damage, but you can never be too careful. The author found it convenient to use red wires for power and ground connections, white for bus connections, and blue for connections from the 8255 to the output connector.

Start with the 6 wires that run from the bus connector to IC2. (By the way, looking at the component side of your motherboard, the "B" side of each expansion slot is on the left and the "A" side on the right, and the connectors are numbered from 1 to 31 from the rear of the board to the front.) Take your time, and check each solder joint for shorts with adjacent pins.

Then connect the eight wires from IC2 to the jumper block, continue with the eight data-bus wires from the bus connector to the 8255, then the six control wires to the 8255. Then connect the 24 wires from the port outputs of IC1 to J1. The author used a 40-pin header connector for J1 in the prototype. Many projects require a source of +5 volts, so power and ground lines are also brought to J1.

PARTS LIST

IC1—8255A-5 parallel interface
IC2—74LS138 3-to-8 line decoder
C1–C3—0.1 µF
J1—40-pin header connector
Jumper block (2 rows, 8 positions)
PC bus prototyping board
Sockets, wire, etc.

Programming examples

The following examples assume that the jumper is in position three, so that the 8255 is connected to port 0240h.

When power is first applied, ports A, B, and C are all configured as inputs. To reconfigure the port, you must write the appropriate value to the correct port. For example, by connecting eight LED's to Port A as shown in Fig. 3, you could view the binary counting sequence using this program:

```
10 OUT 579,128
20 A=0
30 OUT 576,A
40 A=A+1
50 IF A 255 GOTO 20
60 GOTO 30
```

If one LED doesn't seem to light, run this program:

```
10 OUT 579,128
20 OUT 576,255
```

All of the LEDs should light. If one doesn't, check your wiring.

Reading input values is just as simple. The following program would continually read and display the contents of port B, to which various switches (Fig. 4-a, Fig. 4-b) and sensors (Fig. 4-c) might be connected:

```
10 OUT 579,130
20 A=INP 577
30 IF A
40 GOTO 20
```

That program sets up Port B for input, and then reads the value of the port. If the value is less than 255 (in other words, if at least one line is low), the value is printed.

The 8255's inputs and outputs are TTL compatible, meaning they don't have much current-carrying capacity. To drive heavier-duty devices, use a transistor, as shown in Fig. 5-*a*, or add a relay, as shown in Fig. 5-*b*.

More Ideas

Now that you understand the basics, the sky's the limit. What else could you do?

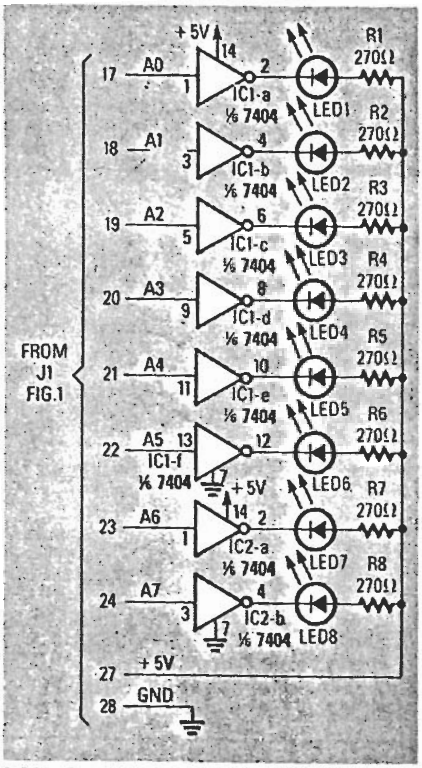


FIG. 3—FOR OUTPUT DISPLAY, add eight LED'S, eight resistors, and two 7404's.

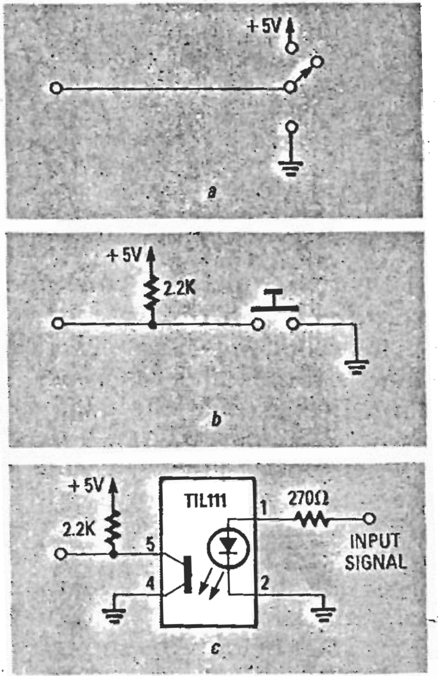


FIG. 4—FOR INPUT, add a toggle switch (a), a pushbutton switch (b), or an opto-isolator (c).

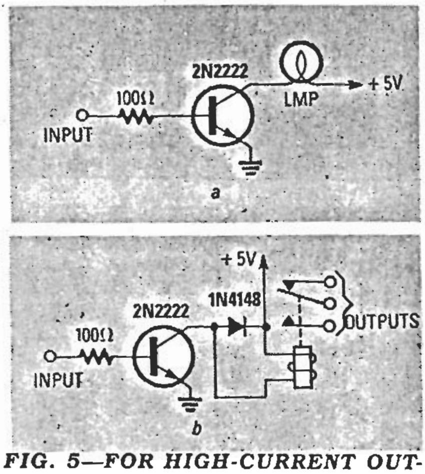


FIG. 5—FOR HIGH-CURRENT OUTPUT, use a transistor to drive a lamp (a) or a relay (b).

- How about building a robot? Output ports could be configured for motor control, voice synthesis, robotic arm control, etc. Input ports could be used to read bumper sensors, voice recognition, or keypad input.
- Or build a burglar alarm: Input ports would read data from window and door switches, and from motion detectors. Outputs would control lights, a siren, and a telephone dialer.
- Or build a home heating system. One port would be dedicated to motors that would open and close heating vents, control blower motors, etc. Input ports would read thermometers in each room and outside the house. A real-time clock would be used to turn heat on in the morning and off in the evening. You could include a wind speed gauge, controls for a solar hot-water heater, and even calculate your energy savings.
- Or build a scoreboard, a light show, or an IC tester. How about a computer-controlled popcorn popper or a dog food dispenser? The author has used his card to run a plotter, an EPROM programmer, and a model-railroad demonstration.

Another thing you could try building is an automatic home lighting system. Input ports could monitor doorways with pressure-sensitive switches or infra-red beams. The system would sense someone entering or leaving the room, and turn the lights on and off accordingly. The system would have to keep track of how many people were in the room, turning the lights off only after the last person leaves.

For some projects, three ports may not be enough. In that case, just connect a second 8255, wiring all lines except cs in parallel to IC1. Connect the cs line of the second 8255 to a different position on the jumper block—and enjoy 48 lines of digital I/O! ◗CD◖

LETTERS

Write to Letters, Radio-Electronics, 500-B Bi-County Blvd., Farmingdale, NY 11735

DESOLDERING DEVICES

I'm writing in response to Mr. Perdue's letter that appeared in the May 1990 issue of **Radio-Electronics**. (He was responding to a previously published letter that concerned IC removal.) I take extreme exception to his general statement that the use of desoldering braid is "the only one approved by government organizations." I work for a government organization that regards the use of desoldering braid as a last resort! As a matter of fact, the U.S. government approves several methods of both soldering and desoldering components on printed-circuit boards.

One of the approved methods of desoldering IC's is to use a motorized vacuum device, such as a "Pace Kit." Of course, that method requires specialized training, and the cost of such a device would be prohibitive to the average home technician. Next on the list is the mechanical vacuum device, commonly known as the "solder sucker." That item is available at most electronics parts stores. To use it, simply cock it and then heat the joint to be desoldered with a low-wattage iron. As soon as the solder begins to flow, place the tip of the solder-sucker vacuum over, and in physical contact with, the joint, and press the trigger button. When the solder sucker has tripped, remove both the iron and the vacuum immediately. Check the joint to make sure that all the solder was removed. If not, simply repeat the procedure.

A few words of caution: First, use a low-wattage (10-25 watts) soldering iron at all times, to prevent overheating the IC and causing internal damage. A temperature-controlled iron is even better. Next, when working with IC's—particularly CMOS types—always use a ground strap (a metal wrist strap with a detachable ground wire) and connect its wire to ground. That will prevent the dreaded static discharge from destroying your IC or other components on the circuit board. If you don't have a ground strap, discharge yourself on a cold-water pipe or some other type of ground beforé starting work. Finally, use a small-diameter pointed or wedge tip on your soldering iron. That helps to heat only the area intended to be heated, and will prevent circuit-board runs from being lifted.

As a last resort, Mr. Perdue's desoldering braid method, as described in his letter, will work. Using either the solder-sucker or the solder-wick method will take some practice, and I, too, would recommend that the novice practice on a junk circuit board to get a feel for either method. That lessens the chance of accidentally destroying a good circuit board or its components.

STEVEN E. SWENTON
Glen Burnie, MD

I/O CARD INPUT

I was intrigued by Mark Hanslip's article, "Build This Experimenter's I/O Card" (**Radio-Electronics**, June 1990). I find it amazing that the 8255 PPI, an LSI IC introduced about a decade and a half ago for 8080 systems, is still being used in new designs.

I disagree with the author's statement about Port B when Port A is initialized for mode 2 operation: "Port B is not used at all." Although Port B cannot be initialized for mode 2 operation, it is far from useless. Port B, independent of Port A's mode of operation, can still be used in either mode 0 or 1.

One last thing: The pins of Port C that are not commandeered for use by Ports A and B (when operating in modes 1 and/or 2 for handshaking) are available for use as input or output lines.

JAMES KOVAR
Lincoln, NE

Fig. 1 goes here

Mr. Kovar has a point. The chart that he provided (Fig. 1) leaves a blank where Port B would be located in relation to mode 2. As I have never needed to use mode 2, the situation has never come up. Thanks, Mr. Kovar, for clearing up that anomoly.

Mode Definition Summary

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA_0	IN	OUT	IN	OUT	↔
PA_1	IN	OUT	IN	OUT	↔
PA_2	IN	OUT	IN	OUT	↔
PA_3	IN	OUT	IN	OUT	↔
PA_4	IN	OUT	IN	OUT	↔
PA_5	IN	OUT	IN	OUT	↔
PA_6	IN	OUT	IN	OUT	↔
PA_7	IN	OUT	IN	OUT	↔
					↔
PB_0	IN	OUT	IN	OUT	—
PB_1	IN	OUT	IN	OUT	
PB_2	IN	OUT	IN	OUT	MODE 0 OR MODE 1 ONLY
PB_3	IN	OUT	IN	OUT	
PB_4	IN	OUT	IN	OUT	
PB_5	IN	OUT	IN	OUT	
PB_6	IN	OUT	IN	OUT	—
PB_7	IN	OUT	IN	OUT	—
PC_0	IN	OUT	INTR_B	INTR_B	I/O
PC_1	IN	OUT	IBF_B	$\overline{\text{OBF}}_B$	I/O
PC_2	IN	OUT	$\overline{\text{STB}}_B$	$\overline{\text{ACK}}$	I/O
PC_3	IN	OUT	INTR_A	INTR^{B}_{A}	INTR_A
PC_4	IN	OUT	$\overline{\text{STB}}_A$	I/O	$\overline{\text{STB}}_A$
PC_5	IN	OUT	IBF_A	I/O	IBF_A
PC_6	IN	OUT	I/O	$\overline{\text{ACK}}_A$	$\overline{\text{ACK}}_A$
PC_7	IN	OUT	I/O	$\overline{\text{OBF}}_A$	$\overline{\text{OBF}}_A$

CAN ALSO BE USED WHEN PA IS MODE 2

OR