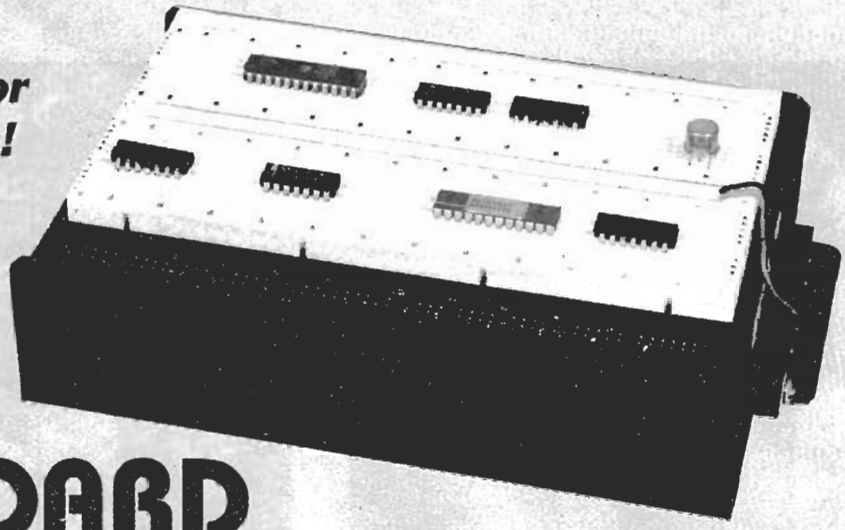
Build this addictive breadboard system for your PC—for peanuts!



BUILD THIS PC I/O BREADBOARD

DAVE DAGE

BUILD THIS SIMPLE, STRAIGHTFORward circuit that lets you breadboard circuits and control them with your personal computer. The circuit consists of three parts: a parallel-interface card, an external breadboard platform, and a cable to connect them. You can control your designs with BASIC, C, assembly language, or any other language that gives you direct access to input/output ports.

This project will be presented as three articles. Part 1 details building and testing the circuit. Part II discusses programming and provides examples in BASIC, Quick C, and assembly language (using DOS's DEBUG program). Part III provides a practical example of using the circuit for a practical application: EPROM programming.

The interface card has only four standard ICs; the decoding and buffering circuit in the breadboard box has only three ICs, plus ten octal latches—one for each for the ten input and output ports. This article presents complete construction diagrams, including the PC board artwork. In addition, complete and partial parts kits are available, as mentioned in the Parts List.

Overview

The interface card is designed to operate in any standard 8- or 16-bit IBM PC or compatible expansion slot. The primary purpose of the card is to buffer and decode the signals necessary for driving a set of 32 I/O ports accessible beginning at a jumper-configurable address (0100h, 0120h, 0130h, . . . 01C0h). A DB-25 cable connects the interface card to the breadboard.

Within a given block of 32 I/O ports, the breadboard circuit decodes only the lowest eight. In addition, the input and output ports are accessed at the same addresses. For example, 260 (decimal) is the address of input port 4; it is also the address of output port 4.

Of the 16 decoded read and write ports, ten (five inputs and five outputs) are latched; the decode strobes for the remaining six (three inputs and three outputs) are available for your projects. To power your projects, the breadboard provides ±5- and ±12-volt power, which is supplied by the host computer.

A 126-pin interface connector is positioned along the top edge of the breadboard; it provides access to all the buffered and decoded I/O port signals, as well as selected computer-bus and control signals. The five independent input ports and five independent output ports, each eight bits wide, are also available at that connector. You can access all I/O ports directly at their actual I/O addresses, rather than following some translation routine, as is done for some breadboard circuits.

The breadboard provides access to useful bus signals including reset, several interrupt lines, decoded I/O read and write signals, and buffered copies of the low-order address lines (A0–A3). With that background od the I/O breadboard in mind, consider the details of the circuit.

Interface card

Microprocessors in the Intel CPU family provide a 16-bit I/O address space, allowing for 65,536 I/O ports. However, IBM's original PC design decoded only the ten lowest address bits, thereby limiting the maximum number of ports to 2^{10}, or 1024.

Within that range, many ports are reserved for controlling such functions as disk drives, video cards, serial and parallel ports. However, a large block of ports (0100–01CFh) has been set aside for I/O interfacing. Actually, that address range is undocumented in the PC/XT but specifically set aside for I/O use in the PC/AT. Anyway, this design lets you select one of seven 32-port blocks in that range.

As shown in Fig. 1, IC2 decodes address lines A4–A9 into seven groups of 16 ports. The $\overline{Y0}$–$\overline{Y6}$ outputs of IC2 provide the base address of each block. Jumper JU1 allows you to configure the base address so that conflicts are avoided in your system. This article and the

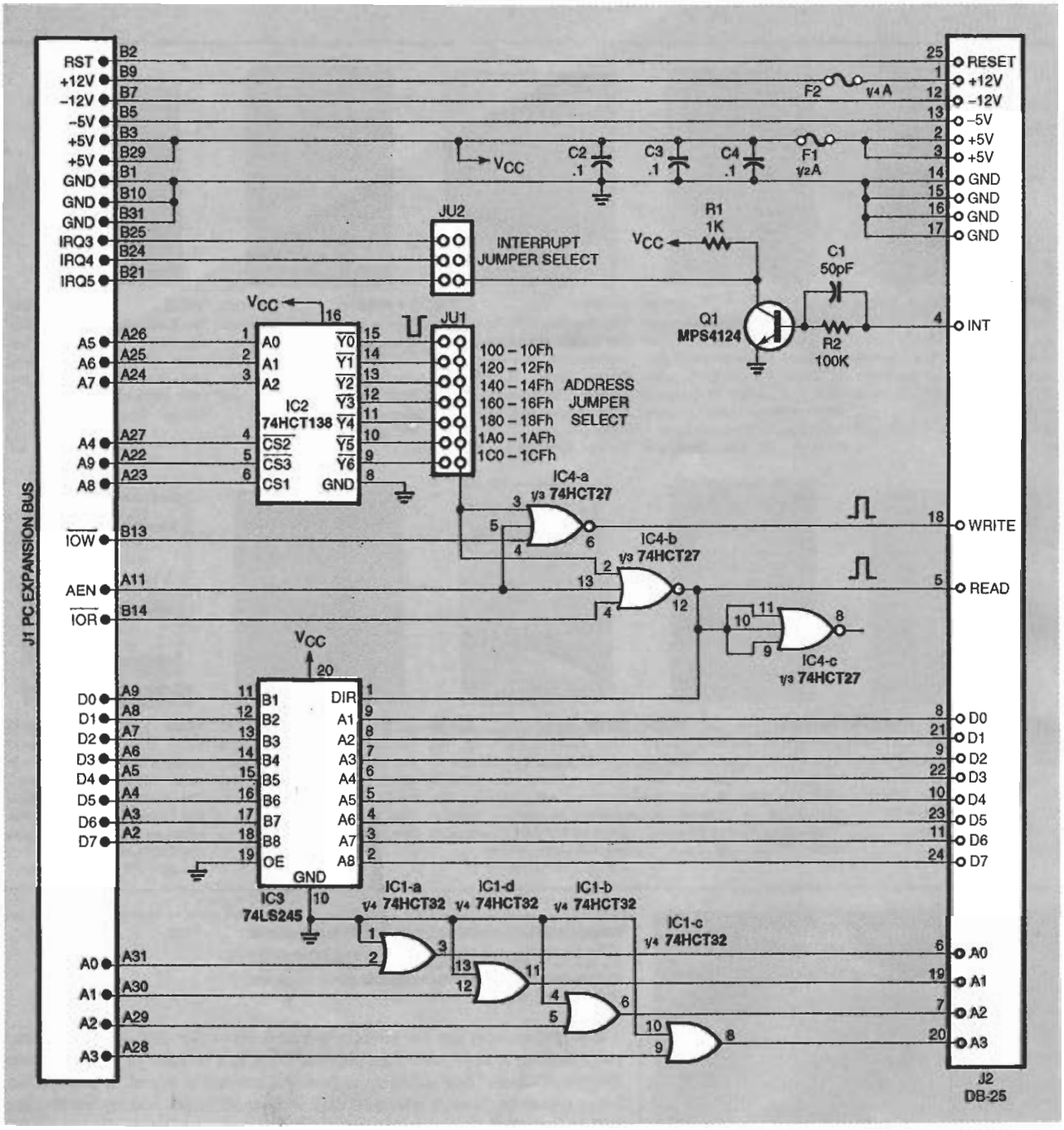


FIG. 1—THE PC INTERFACE CIRCUIT is built around four ICs that decode a 32-byte block of I/O ports, provide a buffered data bus, and buffered low-order address lines. The circuit also provides an interrupt input.

others in this series assume that you will use the lowest address block, 0100–010Fh, or 256–271 decimal.

Conflict is unlikely because all standard PC I/O ports appear at higher addresses. However, if there is conflict, you can help to resolve it by making an index card that lists all the I/O ports, memory addresses, interrupts, and direct-memory access (DMA) channels used by the cards in your computer. Keep the card in a safe place—perhaps taped to the side of the computer—and update it whenever you add a peripheral or change the system configuration in any way. Thus, if you want or need to use a different set of addresses, the circuit lets you do it.

The selected output pulse from address-decoder IC2 drives IC4-a and -b. These NOR gates function as negative three-input AND gates that provide high-going READ and WRITE pulses, respectively. The WRITE pulse occurs whenever the host CPU writes to a port in the selected range, and no DMA operation is occurring (i.e., AEN is low). Similarly, the READ pulse occurs when the host CPU performs a read operation.

The interface card provides an interrupt input. Turning on transistor Q1 with a high-going signal drives an interrupt line on the host PC low. Jumper JU2

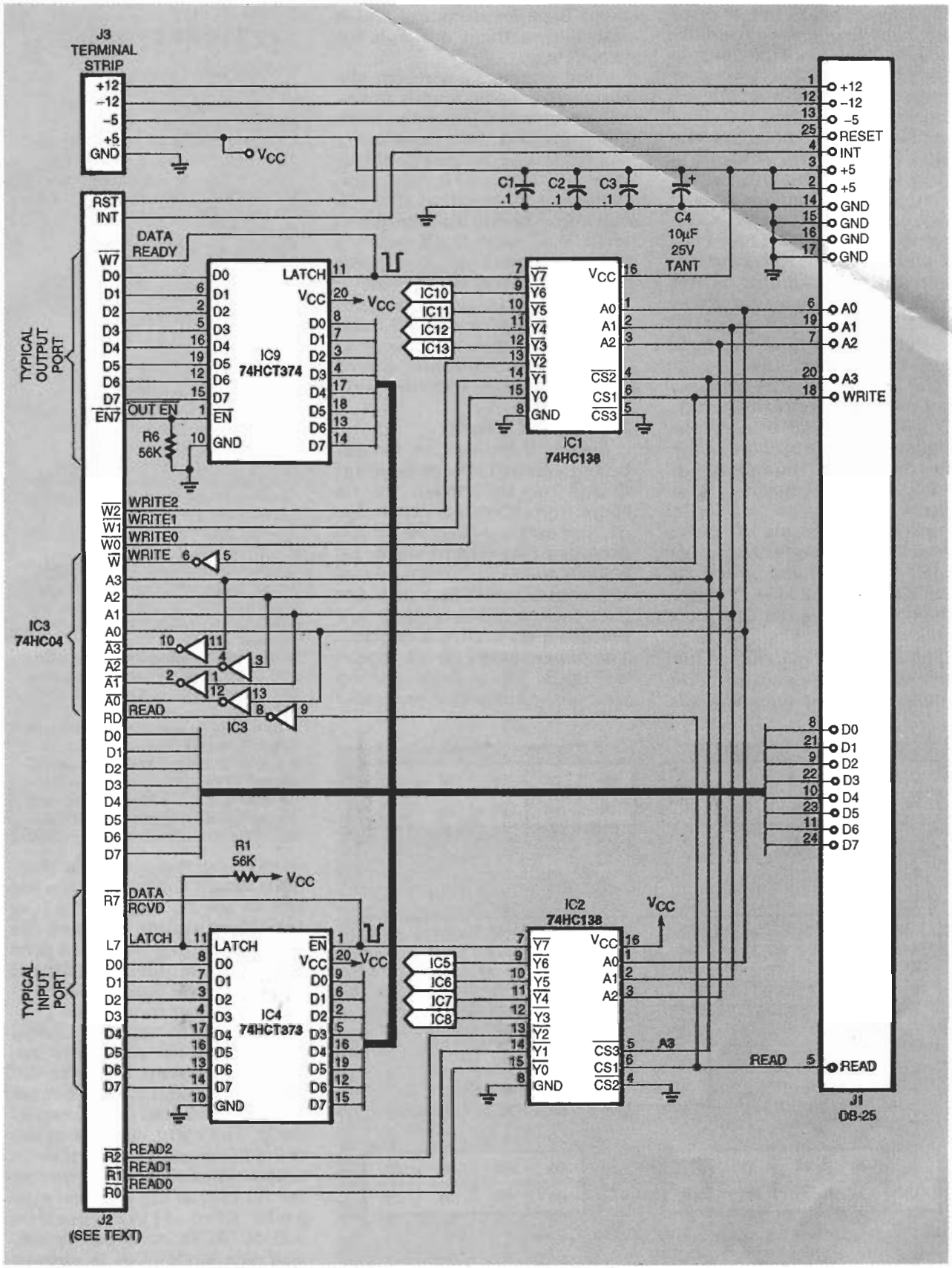


FIG. 2—BREADBOARD CIRCUIT decodes input ports and eight output ports, and latches five of each. To conserve space, only one output latch (IC9) and one input latch (IC4) are shown. The remaining latches connect directly to the data bus; the enable input of each latch connects to the corresponding output of IC1 and IC14.

allows you to select one of three interrupt lines: IRQ3 (usually used by serial port COM2, if present), IRQ4 (COM1), or IRQ7 (reserved for the printer, but seldom used).

Although the interrupt circuitry is simple, the software is complicated. The process of doing interrupts under DOS is involved and that process differs between the PC/XT and PC/AT architectures. Nevertheless, interrupt programming is not difficult, but many pages of text would be required to cover the subject adequately. At this time leave JU2 disconnected.

The circuit provides an 8-bit data-bus interface through IC3, a 74HCT245 octal transceiver. Data direction (into or out of the CPU) depends on the state of pin 1 of IC3. Pin 1 is normally low, which corresponds to a write from the CPU to an I/O port. When an I/O read occurs, pin 12 of IC4-b goes high, which in turn drives pin 1 of IC3 high, thereby reversing the direction of the data.

The gates in IC1 buffer the CPU address lines A0–A3. The breadboard card uses the address lines for decoding and it also makes them available for project use.

Nine of the 25 wires in the connecting cable supply power to the breadboard. Four lines carry ground, two lines carry +5 volts, and one line each carries −5 volts, +12 volts, and −12 volts. Notice that the +5-volt and +12-volt lines are fused with 0.5- and 0.25-ampere fuses, respectively. The fuses protect the wires in the cable. If your projects require more power, make up a separate cable assembly. Connect it directly to a spare four-conductor power connector at the computer end.

Breadboard circuit

Figure 2 shows the breadboard circuit. Overall signal flow is from right to left. Inputs come from 25-pin D connector J1, and outputs go to J2, which provides two 63-pin rows, labeled A and B, yielding a total of 126 connections. The pins are 0.025-inch square that are mounted on 0.1-inch centers. The pin numbers for J2 appear in Fig. 3, which depicts a top view of the connector as viewed from its normal operating position. Mount the pins on the foil side of the PC board, which is positioned upside down in the case. Notice that the Row A pins mount on the outermost edge of the PC board, and the Row B pins are closer to the center.

Refer again to Fig. 2, and notice that the low-order address lines feed I/O port decoders IC1 and IC2, which drive the write and read ports, respectively. The eight outputs of decoder IC1 provide inverted write pulses, all of which are available on J2. Five of the decoded outputs also drive latches IC9–IC13. To conserve space, only one latch (IC9) is shown, but the other four are similarly connected.

Similarly, the IC2 decoder outputs appear on J2 and drive

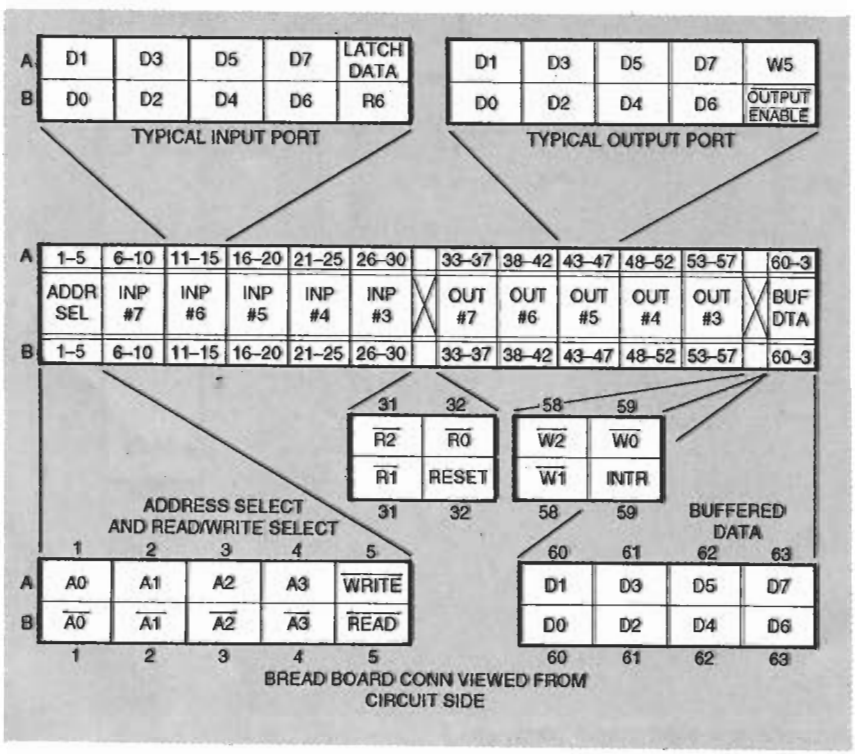


FIG. 3—BREADBOARD INTERFACE CONNECTOR J2 is shown here. There are six groups of pins: address and read/write lines, five input ports, input strobes and reset, five output ports, output strobes and the interrupt input, and the buffered data bus.

PC INTERFACE PARTS LIST

R1—1000 ohms, ¼ watt, 5%
R2—100,000 ohms, ¼ watt, 5%
C1—50 pF, ceramic
C2–C4—0.1 µF, ceramic
IC1—74HCT32 quad 2-input OR gate
IC2—74HCT138 3-to-8 line decoder
IC3—74LS245 octal bus transceiver
IC4—74HCT27 triple 3-input NOR gate
Q1—MPS4124 transistor
F1—0.5 A
F2—0.25 A
J2—25-pin female D connector, right angle, PC mount

BREADBOARD CARD PARTS LIST

R1–R10—56,000 ohms, ⅛ watt, 5%
C1–C3—0.1 µF, ceramic
C4—C10 µF, 25 volts, tantalum
IC1, IC2—74HC138 3-to-8 line decoder
IC3—74HC04 hex inverter
IC4–IC8—74HCT373 octal D latch
IC9–IC13—74HCT374 octal D flip-flop
J1—25-pin male D connector, right angle, PC mount
J2—128 wire-wrap pins, 0.025" square × 0.75" long
Other—25-conductor, 4-foot, male-to-female cable; solderless breadboards; 5-position terminal block; chassis; PC board guide; rubber feet; mounting hardware.

Note: The following items are available from Dage Scientific, P.O. Box 144, Valley Springs, CA 95252, (209) 772-2076:

- **Complete kit including manual and all parts (model ST-1)—$119**
- **Set of 2 PC boards and manual (model ST-2)—$40**

All orders add $3.95 shipping and handling. CA residents add sales tax.

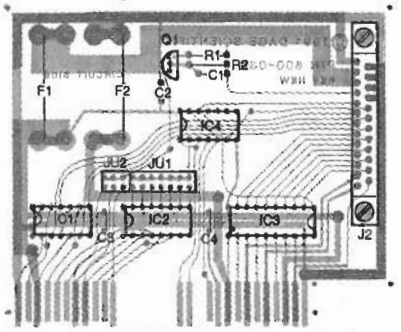


FIG. 4—INTERFACE CARD parts-placement diagram.

IC4–IC8. Only IC4 is shown. Notice that J2 provides true and inverted versions of the low-order address lines. It also provides true and inverted versions of the READ and WRITE lines from the interface card. The power lines from J3 connect to a five-position terminal strip that mounts on the opposite side of the case from J3. Use extreme care to avoid feeding any incorrect voltages to the breadboard circuits or back into J2.

Construction

The use of printed-circuit boards is recommended, particularly for the PC interface card. A breadboard circuit can be built with any accepted point-to-point wiring technique, but the PC board shown here will simplify assembly and make it easier to locate errors. The only point-to-point wiring required on this board is five wires from the board to the power strip J3.

To test this circuit and subsequently for building other circuits, you'll need several jumper wires. The connection of J2 calls for wires terminated with female pin sockets on one end and solid (or tinned stranded) wire on the other. You'll also need solid-wire jumpers for interconnecting components on the breadboard.

Referring to the parts-placement diagrams shown in Figs. 4 and 5 as guides, assemble both boards. Remember to mount the J2 pin strips on the foil side of the board. Figure 6 shows the relationship between the PC board, the J2 pins, the case, and the breadboard strips.

Check your soldering and correct any mistakes and remove any solder bridges. Install a ½-ampere fuse in the 5-volt supply line and a ¼-ampere fuse in the 12-volt supply line. Install a jumper at JU1 to select an address range that is available in your computer (assume the default address range, 0100–010Fh). Leave interrupt jumper JU2 disconnected at this time. Mount the breadboard card in a case that measures about 7.0 × 4.5 × 1.25 inches. Mount the D connector on one end of the box and the power terminal on the other. Cut a slot for accessing the J2 pins, and mount several rows of 62-pin solderless breadboard strips with double-sided tape.

Install the interface card in any available 8- or 16-bit slot of your computer. Be sure to seat the card firmly. Connect a 4-foot long, 25-conductor cable between the interface card and the breadboard assembly.

Testing

BASIC will be used for the test programs, and you'll also need a logic probe. The advantage of BASIC is its simplicity and its availability. You can use any version of MS-DOS or PC-DOS BASIC, including BASIC (on real IBMs), GW-BASIC (on older versions of MS-DOS), or QBASIC (on DOS 5.0 and later). Two BASIC statements provide access to the project's I/O ports. The first reads the value of the specified port that is read into variable N.

```
N = INP (PORT)
```

The second writes N to the specified port.

```
OUT PORT, N
```

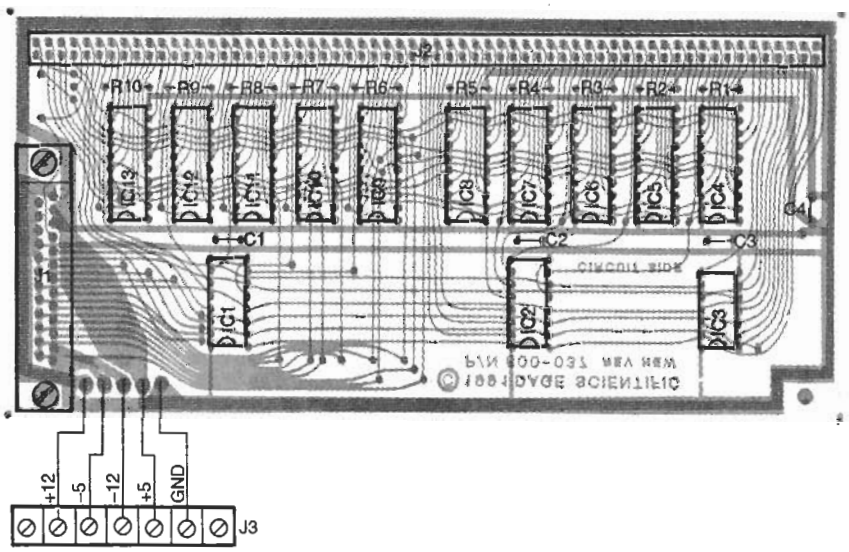


FIG. 5—BREADBOARD CARD parts-placement diagram. All 126 of the pins in J2 mount on the solder side of the board.

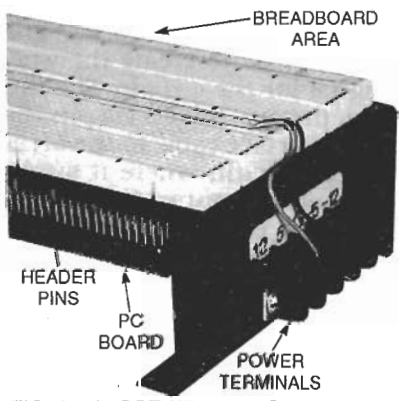


FIG. 6—CLOSE-UP VIEW OF J2, the circuit board, chassis, and breadboard strips.

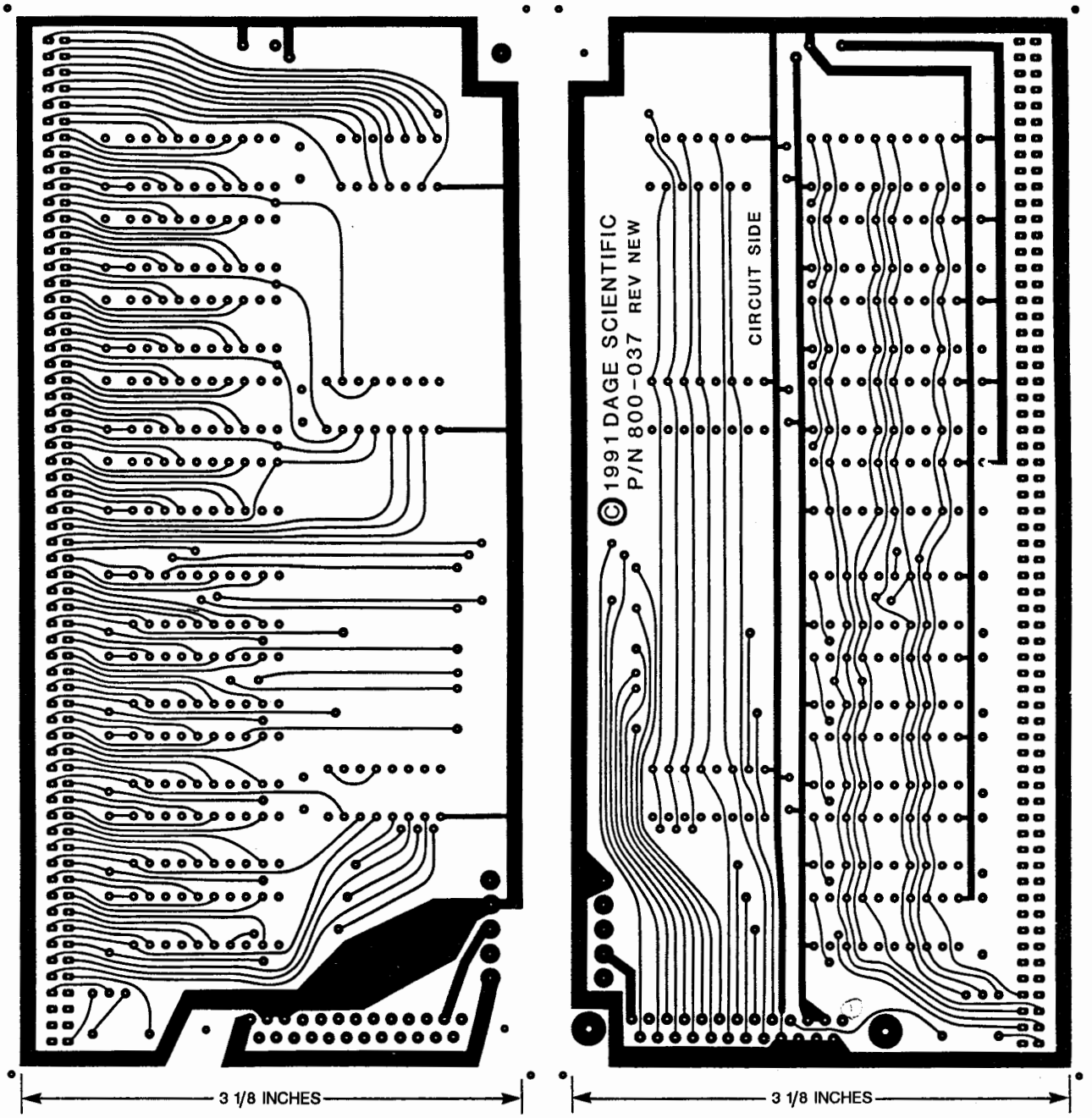
For both input and output statements, BASIC requires decimal values. For example, you could read the current value of the first port (residing at the default base address) as follows:

```
N = INP(256)
```

You could also write the current value of N to the same port as follows:

```
OUT 256, N
```

Table 1 provides quick reference information for port numbers and their corresponding decimal and hexadecimal addresses. The five fully decoded I/O ports are 3–7. Ports 0–2 pro-



FOIL PATTERNS FOR THE PC I/O BREADBOARD shown actual size.

vide only the strobe signals.

Listing 1, FLIPOUT.BAS, is a simple BASIC program that exercises many circuit elements. Load BASIC, then enter and run the code shown in the listing. The program works with output port 7. It successively flashes all outputs sequentially off, then on again. You will find it instructive to trace the signal all the way through the circuit.

As shown in Fig. 1, pin 15 of IC2 goes low whenever the CPU accesses any I/O port 0100–010Fh. That signal combines with the CPU's $\overline{\text{IOW}}$ and AEN signals to create the WRITE signal for the circuit. That signal, in turn, travels via pin 18 of the interface cable to the breadboard circuit, where it strobes the pin-5 input of hex inverter IC3. Pin 6 of that IC delivers a buffered WRITE signal to pin 5A of J2. Verify this operation with the logic probe to "catch" pulse activity there. If the circuit is working, the logic probe will blink. If the probe is not blinking, trace back through the circuit to find where the signal stops, and repair the error.

The WRITE signal also drives the CS1 input of decoder IC1. When the CPU addresses port 7, pin 7 of IC1 goes low, which in turn latches any signals the CPU put on the data bus into IC9's outputs. The latch signal and the eight data bits appear on pins 33A–37B and 33B–36B of J2. Verify that all nine signals are toggling with the logic probe.

Listing 2, CYCLEOUT.BAS, provides a more comprehensive test. It cycles all output ports, all write pulses, and address lines

TABLE 1—PORT ADDRESSES

Port Number	Hexadecimal Address	Decimal Address
0	100	256
1	101	257
2	102	258
3	103	159
4	104	260
5	105	261
6	106	262
7	107	263
8	108	264
9	109	265
A	10A	266
B	10B	267
C	10C	268
D	10D	269
E	10E	270
F	10F	271

LISTING 1
FLIPOUT.BAS

```
10 OUT 263,0
20 OUT 263,255
30 GOTO 10
```

A0–A3. While the program is running, check to see the activity on J2 pins 33A–59A and 33B–58B. Notice that in this and the preceding test, the output enable pins (37B, 42B, 47B, 52B, and 57B) should not cycle because they are held low by 56K resistors R6–R10. The output enable pins are made available at J2 in the event that a breadboard circuit might have to disable the outputs of a latch integrated circuit.

The signals on the data-bus pins (J2 pins 60A–63A and 60B–63B) are subject to change, as will the true and inverted address lines A0–A3 (J2 pins 1A–4A and 1B–4B). However, both the data and address lines will be cycling much faster than the output ports. This occurs because they indicate what is happening in the computer, not just at the I/O ports. The address and data lines should be cycling as long as the computer runs. By contrast, the I/O lines should cycle only when the CPU accesses an I/O port in the range specified by JU1.

Input ports

Check the input ports in a different way. For example, check

LISTING 2—CYCLOUT.BAS

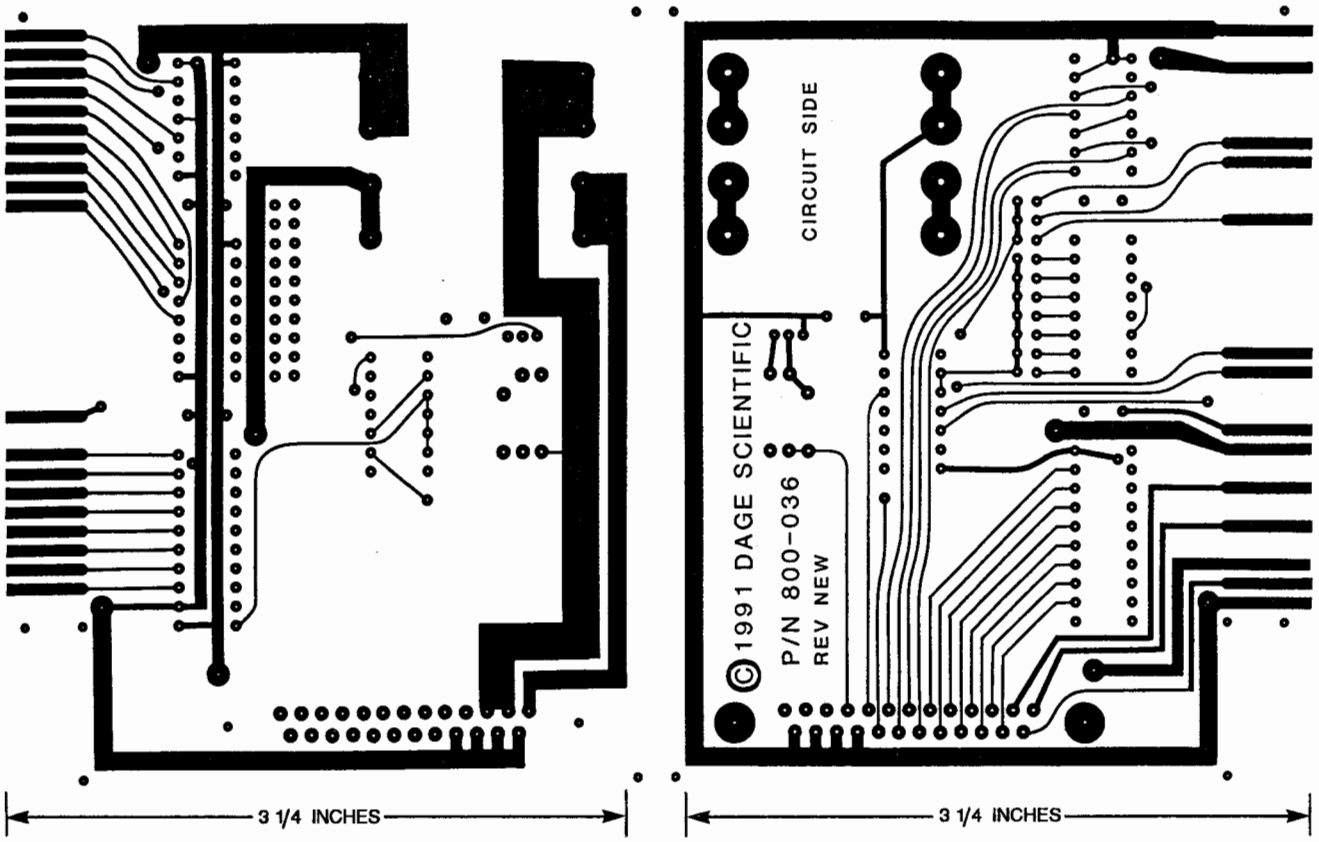
```
100 REM * for ST-1 Purpose: cycle each output data line for IC's 9 - 13, plus
110 REM * cycle all write pulses W0 - W7 and the base 'write' pulse.
120 REM * Hardware: Logic probe to check pulse activity.
130 REM ************************************************************************
140 REM * BA base address, T offset port address, C counter
150 REM ************************************************************************
160 CLS :REM * START
170 INPUT "Enter base address in decimal (for example - 256) ";BA:PRINT:PRINT
180 PRINT"WORKING":PRINT
190 PRINT"Use cntl/break to quit!"
200 FOR C=1 TO 4 : REM cycles clock        *****LOOP1*****
210 REM cycles all 8 outputs with 0's and 1's
220 FOR T=BA TO BA+7 : REM         *****LOOP2*****
230 OUT T,0 : OUT T,255
240 NEXT T : REM                   *****LOOP2*****
250 REM runs clock so that you know program is running
260 LOCATE 4,10
270 IF C=1 THEN PRINT"³"
280 IF C=2 THEN PRINT"/"
290 IF C=3 THEN PRINT"Ä"
300 IF C=4 THEN PRINT"\"
310 NEXT C : REM                           *****LOOP1*****
320 GOTO 200
```

LISTING 3—INP2CRT.BAS

```
100 REM * for ST-1. Purpose: to diplay an input port on screen.
110 REM * Hardware: Switch to set input, resistors.
120 REM ************************************************************************
130 REM * AI address input port, T counter, A$ any key, N input byte
140 REM * BP bit position during printing
150 REM ************************************************************************
160 CLS : REM *START
170 INPUT" Enter input port address in decimal  ";AI
180 PRINT:PRINT" Press Cntl/Break to exit!
190 REM * Draw 8 boxes at center screen to display input byte ***************
200 LOCATE 9,26 : PRINT"#7  #6  #5  #4  #3  #2  #1  #0"
210 LOCATE 10,25 : PRINT"Ú";
220 FOR T=1 TO 7 : PRINT"ÄÂÄÂ"; : NEXT T : PRINT"ÄÄ¿"
230 LOCATE 11,25 : FOR T=1 TO 8 : PRINT"³   "; : NEXT T : PRINT"³"
240 LOCATE 12,25 : PRINT"À";
250 FOR T=1 TO 7 : PRINT"ÄÁÄÁ"; : NEXT T : PRINT"ÄÄÙ"
260 REM * GET PORT AND DISPLAY ***********************************************
270 BP=27 : REM used as pointer for bit print       *****LOOP1*****
280 N=INP(AI)
290 FOR T=7 TO 0 STEP -1 : REM             *****LOOP2*****
300 LOCATE 11,BP
310 IF N AND 2^T THEN PRINT"1" ELSE PRINT"0"
320 BP=BP+4:NEXT T : REM                   *****LOOP2*****
330 A$=INKEY$ : IF A$="" GOTO 270 : REM              *****LOOP1*****
340 GOTO 270
```

LISTING 4—LOOPBACK.BAS

```
100 REM * for ST-1 Purpose: to check I/O ports by feeding a test byte from
110 REM * an output to an input and comparing.  Three tests are made:
120 REM * bits stuck high or low, bits tied together, and everything else.
130 REM * Hardware: Loopback plug.
140 REM ************************************************************************
150 REM * AO address output port, AI address input port, BC bit counter
160 REM * TB test byte, SH switch halt on section error, S1 switch temp
170 REM * S1 switch one of two errors, T counter
180 REM ************************************************************************
190 CLS: SH=0 : REM * initialize soft switch
200 INPUT " Enter output port address in decimal  ",AO
210 INPUT " Enter  input port address in decimal  ",AI
220 PRINT:PRINT"Program checking for bits stuck high or low.":PRINT
230 REM * test for bits stuck high ****
240 OUT AO,0
250 TB=INP(AI)
260 FOR BC=7 TO 0 STEP -1
270 IF (TB-2^BC)>=0 THEN PRINT"bit ";BC;" is stuck high":TB=TB-2^BC:SH=1
280 NEXT BC
290 REM * test for bits stuck low ****
300 OUT AO,255
310 TB=INP(AI)
320 FOR BC=7 TO 0 STEP -1
330 TB=TB-2^BC
340 IF TB<0 THEN PRINT"bit ";BC;" is stuck low":TB=TB+2^BC:SH=1
350 NEXT BC
360 IF SH=1 THEN GOTO 560
370 LOCATE 4,50:PRINT"** PASSED **"
380 PRINT:PRINT"Program checking for bits stuck together.":PRINT
390 FOR BC=0 TO 7
400 S1=0:REM loop switch; set to one on any fault
410 OUT AO,    2^BC:TB=INP(AI): IF TB<>    2^BC THEN S1=1
420 OUT AO,255-2^BC:TB=INP(AI): IF TB<>255-2^BC THEN S1=1
430 IF S1=1 THEN PRINT" check bit";BC:SH=1
440 NEXT BC
450 IF SH=1 THEN GOTO 560
460 LOCATE 6,50:PRINT"** PASSED **"
470 PRINT:PRINT"Checking all bit combinations. Please wait!":PRINT
480 FOR T=0 TO 255
490 OUT AO,T
500 TB=INP(AI)
510 IF TB<>T THEN PRINT"Error at";T:SH=1
520 NEXT T
530 IF SH=1 THEN GOTO 560
540 LOCATE 8,50:PRINT"** PASSED **"
550 PRINT:PRINT"Test done, loopback good!":PRINT:GOTO 570
560 PRINT:PRINT"Program halted. Bit error.":BEEP
570 END
```



FOIL PATTERNS FOR THE INTERFACE CARD shown actual size.

input port 4, which is located at address 260 decimal. Start by tying all eight inputs (J2 pins 21A–24A and 21B–24B) to ground through 1-Kilohm resistors. Then run the test program INP2CRT.BAS, shown in Listing 3.

The program first asks you to enter the address of the port under test. The binary value of this port will then be displayed dynamically in the center of the screen. All bits have been tied low, so you should see eight "0's." Connect a 100-ohm resistor to V_{CC} and touch the input lines one at a time. As you touch each input, the corresponding value on the screen should change to a 1. Verify that each bit changes, and verify that the correct bit is the one changing (e.g., bit 7 is bit 7 and not bit 6). Also verify that only the bit you touch goes high.

At this time you have cycled all output ports, and you have performed a rather detailed test on one input port. However, a fully functional circuit is not yet guaranteed. For example, the output-port test cycled all the output bits on, then off. The problem was diagnosed as two or more bits shorted together, but the test could still be completed successfully.

A truly comprehensive test would examine all possible input and output combinations individually. It is not difficult to do that. The solution calls for some additional circuitry and software. The circuitry consists of a loopback plug that connects all eight bits of an output port to the corresponding eight bits of an input port. The software is shown in Listing 4, LOOPBACK.BAS.

Build an eight-wire loopback plug, or use eight separate wires. Connect all eight bits from one input port to the corresponding bits of an output port (bit 0 goes to bit 0, bit 1 goes to bit 1, and so on). Next run LOOPBACK.BAS. The program will ask you the addresses of the input and output ports before it will start testing.

The program performs three tests: 1) Bits stuck low or high, 2) Bits shorted together, and 3) Bit independence. Bit independence means the ability to read and write all binary combinations of bits (i.e., every number from 0 to 255). For the first two tests, the program halts if an error occurs. If a bit is stuck low or high, the program will display the faulty bit number and halt. The program will not proceed until the problem has been corrected.

The second part of the program senses bits that are shorted together and displays them by number. Again it will stop until the fault is corrected.

By the time the port has passed the first two tests, it should be fully functional. However, just to be sure, the third part of the test checks all possible bit combinations. In the unlikely event that a fault occurs, the program displays the incorrect value, but in this case the program keeps on running. After running LOOPBACK.BAS against all five input and output ports, you can be confident that the circuit works properly.

Build your breadboard circuit and interface card and debug it. Next month more details on programming the circuit will be published. Ω