Construction project:

Universal "Real World" interface for PCs - 3

Finally, here it is! Full constructional details, showing how to get your version of the Real World Interface up and running – at least in its basic form.

by MARK CHEESEMAN

As stated in the previous articles for this project, the interface may be assembled in one of two ways, depending upon whether you are going to connect it to a serial or a parallel port on your computer. Your choice of interface will determine which components are required to build the project. The parts list has been divided into three separate sections, with the first containing those parts which are common to both versions, and the other two detailing the additional parts required for either the serial or parallel version.

There are also two separate overlay diagrams, one for each version, so that there should be no confusion as to which components need to be fitted to which version.

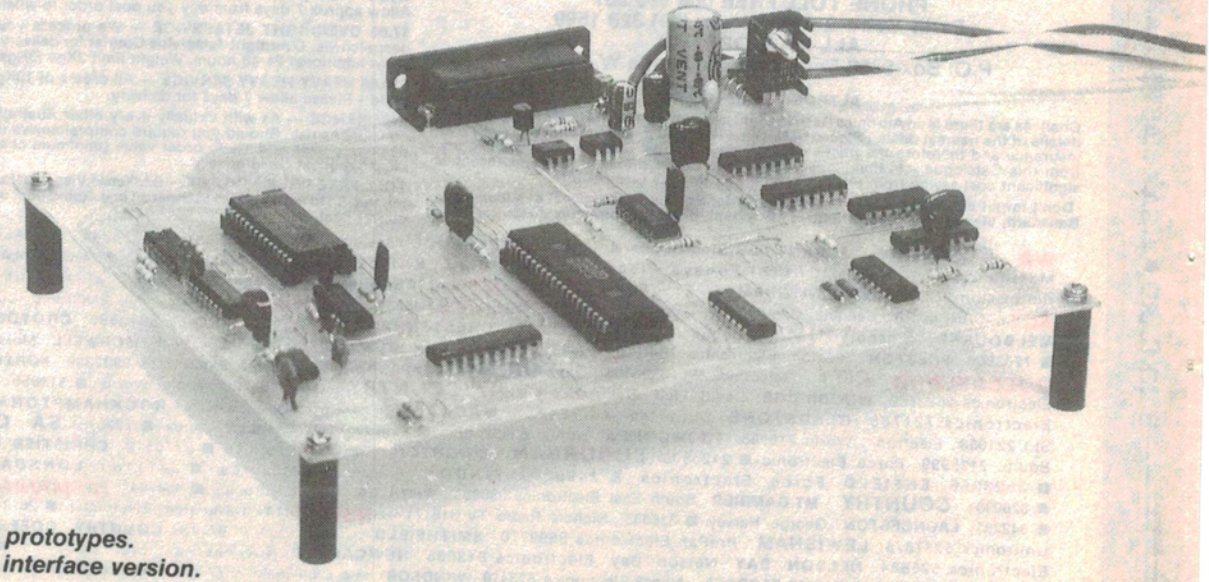
No matter which version you are building, you should first check the board carefully for bridges between tracks, and rectify any that you may find. After that, you should mount the required links on the board. There are quite a lot of these, due to the fact that we have used a single-sided board to keep costs down.

If you intend to put in sockets for the ICs, then this should be tackled next. Sockets for the large ICs (the UART and the ADC) are strongly recommended, owing to the difficulty in de-soldering such large ICs should the need arise. However sockets for the other ICs may be considered to be optional.

Next, mount the resistors, capacitors and diodes, taking note of the polarity of the polarised components. Finally, mount all the ICs, either soldering them in directly, or inserting them into their sockets if you have used them. Note that not all of the ICs face in the same direction, so watch their polarity carefully.

It is perhaps in order to discuss the choice of C5 at this point. This is the capacitor used to determine the baud rate of the serial interface and also the sampling rate of the ADC. The table of values listed in part two of this series gives values for data rates up to 9600 baud. Beyond this speed the 4N28 optocouplers start to look quite sluggish, so if you want higher speeds over a serial interface, faster optocouplers (such as the 6N138) will probably be required. However these will not fit on the existing PC board, as they have 8 pins instead of 6. C5 itself may be either a ceramic or metallised polyester capacitor, depending upon the actual value required.

If you are building the parallel interface version, the frequency of the 555 only determines the sampling rate of the ADC, and may be theoretically increased up to the maximum speed of the ADC chip, which is 1280kHz. How-



One of the completed prototypes. This one is the serial interface version.

ever you may have difficulty in getting the 555 to operate this fast. RV1 is not necessary on the parallel version, as there is no need to adjust the frequency of the 555 as accurately as is required for the UART clock signal.

Finally, mount the right-angle DB-25 connector, using a male connector for the parallel version of the board, or a female for the serial. Do not be tempted to use a connector of the wrong 'sex', as the pins will be the wrong way around, and you can be virtually assured of failure.

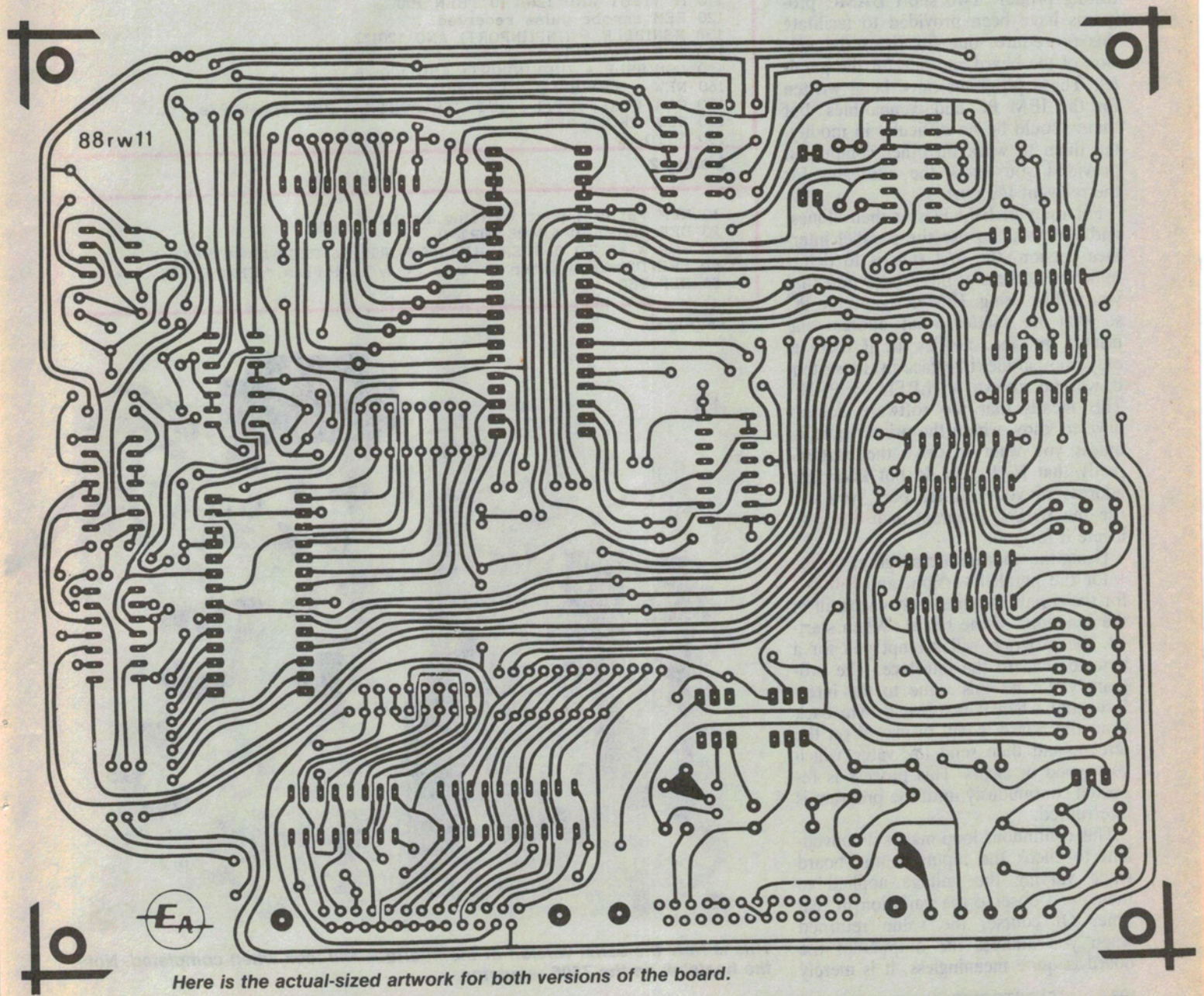
Because the serial interface is optocoupled, the power supply for the interface board should be independent from that for the computer. Power may be obtained from any transformer capable of supplying 8 – 12V AC at 150mA or so.

For the serial version, a separate power source is also required to power the RS-232C level converting circuitry on the computer side of the optocouplers. This may be derived from a separate winding on the same transformer used to power the rest of the board, from a completely separate transformer, or from the computer itself.

The latter alternative is the most cost-effective, as you not only save on the cost of a second transformer, but can also omit D1, D2, C1 and C2 as well. All that is required is a source of +/- 9 to 15 volts DC at a few milliamps. If you intend to power the interface in this manner, the DC supply should be connected to the pads where the diodes would have gone. The positive rail should be connected to the pad marked as the cathode (striped end) of D1. The negative rail then connects where the anode of D2 would have gone.

These rails may be connected to two unused pins on the DB-25 connector in the computer, and then small jumpers added to the back of the interface board to connect the two chosen pins to the pads mentioned above.

If you cannot take the power from the computer, then you will need to either use a dual-secondary transformer to power the interface, or use a second transformer to power this part of the circuitry. While a dual-secondary transformer is the most elegant solution in this case, it may be cheaper to opt for two separate transformers, so check out the prices first.



Here is the actual-sized artwork for both versions of the board.

Real world

Testing

For this part of the procedure it would be useful to have a logic probe and also a frequency meter if you are building the serial version, although they are not absolutely vital. A multimeter would also be useful in ascertaining whether the power supply is doing what it is supposed to. First, power up the board, and check that all the ICs have power on the appropriate pins. If there is no power anywhere on the board, and the regulator seems to be getting a little hot under the collar, remove the power and check for shorts under the board.

Once any problems here have been cleared up, you can proceed with the testing proper. Two short BASIC programs have been provided to facilitate this procedure: one for the serial version of the board and one for the parallel. These programs have been written for the IBM PC and compatibles, but there should be no difficulty in modifying them to work on other computers, provided you know the addresses for the relevant I/O port.

For users of IBM PCs or their clones, and who wish to use the parallel interface version, the first step is to determine the address of the printer port that you will be using. IBM designed the PC so that no matter what address the printer interface resides at, if there is only one parallel interface in the system it will be known as LPT1: or PRN:. This means that the software doesn't have to know where the printer port is, unless you want to drive the port directly that is. If you do not know the address of your printer port, you can use the BASIC program in listing 3 to locate it for you.

Using the appropriate program (listing 1 for the parallel version, and listing 2 for the serial), progressively check all of the functions of the board. When started, the program will prompt you for a byte to send to the interface. The program will send this value to the interface, and when it receives a byte back from the board it will display it on the screen, and then send the value which you typed in again. This process is repeated continuously until the program is interrupted.

This continuous loop makes it convenient to check the inputs to the board while varying the voltage applied to them, and observe the variation in real time. Of course, the value returned when you address the outputs of the board is quite meaningless, it is merely

```
10 REM EA Real-World Interface
20 REM Serial interface test routine
30 REM Initialise COM1.
40 OPEN "COM1:9600,N,8,1,BIN" AS #1
50 INPUT "Enter byte to send to RWI"; B
60 IF B<0 OR B>255 THEN PRINT "Invalid.": GOTO 50
70 PRINT#1,CHR$(B);
80 REM wait for acknowledge
90 IF EOF(1) THEN 90
100 PRINT ASC(INPUT$(1,#1))
110 GOTO 70
```

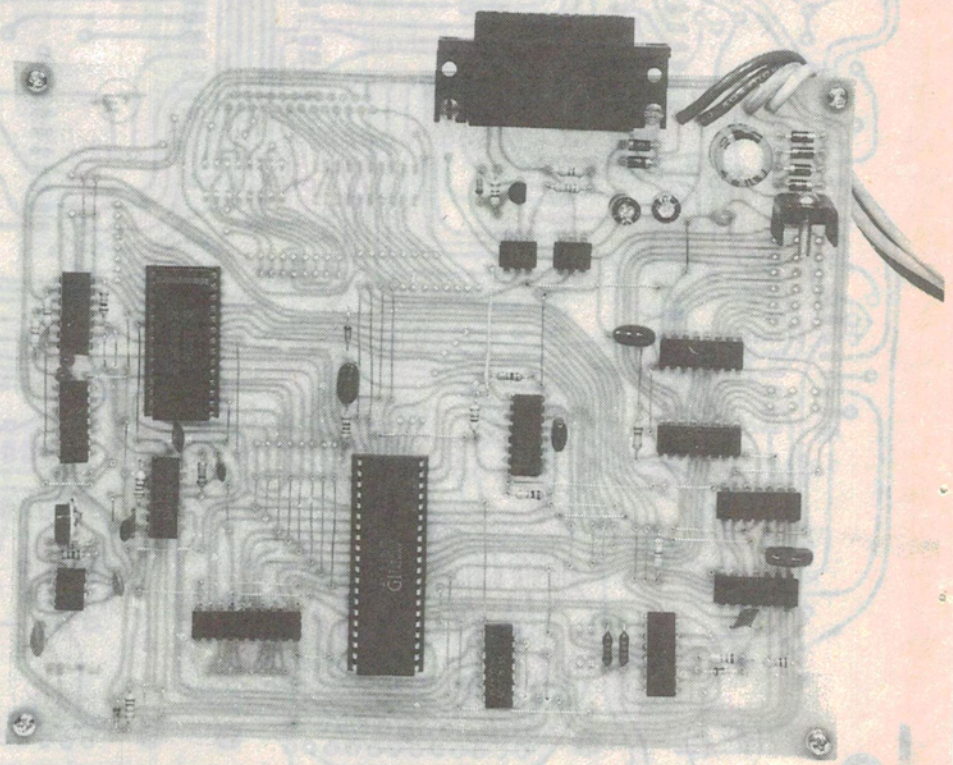
Listing 1.

```
10 REM EA Real-World Interface
20 REM Parallel interface test routine
30 BASE=&H3BC : REM Base Address for primary printer port
40 OUTPORT = BASE : INPORT = BASE + 1 : CTRLPORT = BASE + 2
50 OUT CTRLPORT,3 : OUT CTRLPORT,1 : REM Reset Flip-Flops
60 INPUT "Value to send"; VALUE
70 IF VALUE<0 OR VALUE>255 THEN PRINT"Invalid.": GOTO 60
80 OUT OUTPORT,VALUE
90 OUT CTRLPORT,0 : OUT CTRLPORT,1 : REM Generate Strobe Pulse
100 TEST = INP (INPORT)
110 IF (TEST AND 128) =0 THEN 100
120 REM strobe pulse received.
130 MSNIBBLE = (INP(INPORT) AND 120)*2
140 OUT CTRLPORT,9
150 LSNIBBLE = (INP(INPORT) AND 120)/8
160 NEW = MSNIBBLE + LSNIBBLE
170 OUT CTRLPORT,3 : OUT CTRLPORT,1 : REM reset FFs
180 CLS: PRINT NEW
190 GOTO 80
```

Listing 2.

```
10 REM Parallel port locating routine
20 DEF SEG=0:AD=&H408:PORT=0
30 FOR X =2 TO 0 STEP -1:PORT=PORT*256+PEEK(AD+X):NEXT X
40 PRINT"Your printer port is at I/O address ":HEX$(PORT):"hex"
50 DEF SEG
```

Listing 3.



This is how the serial version of the interface will look when completed. Note the heatsink on the 7805 regulator.

an acknowledgement to indicate that the board got the message.

If you have built your board with a serial interface, the first thing to do is to adjust the frequency of the 555 oscillator to the correct value. If you have a frequency counter handy, simply attach this to pin 3 of the 555, and adjust RV1 until the counter reads the correct frequency, according to table 1 in the October issue.

If you do not have access to a frequency counter, there is another way to set the operating frequency of the 555. First write a program to continuously send bytes to the serial port, without waiting for any form of acknowledgement from the interface. Then run the program, and observe the status of LED1 (the framing error indicator). If this LED is not illuminated, adjust RV1 in one direction until the LED just lights, and take note of this position. Then turn RV1 the other way until the LED just lights again. Finally set RV1 mid-way between these two points. This will result in a frequency that is close enough to the correct value.

TABLE 1

Value	Action Performed
0 to 15	Write 0 to 15 to first output latch (IC7)
16 to 31	Write 0 to 15 to second output latch (IC8)
128	Read digital input latch
144 to 151	Read Analog input 0 to 7

Table 1 lists the values to send to the board to address the various parts of the board. Test each of these values in turn, checking that the screen continuously updates the display, showing the last byte returned from the interface in the upper left-hand corner of the screen. If the program appears to 'hang' at any time, this is when a logic probe will come in handy to trace the signal path through the interface.

To check the outputs, connect a LED and series resistor between each output and the positive rail, as shown in Fig.1(a). When connected to the Q-bar outputs, the LED should light when a '1' is written to the appropriate bit of the relevant output latch. Similarly, the LED should light when connected to a Q output and a '0' is written to the appropriate latch.

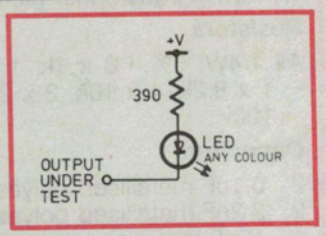
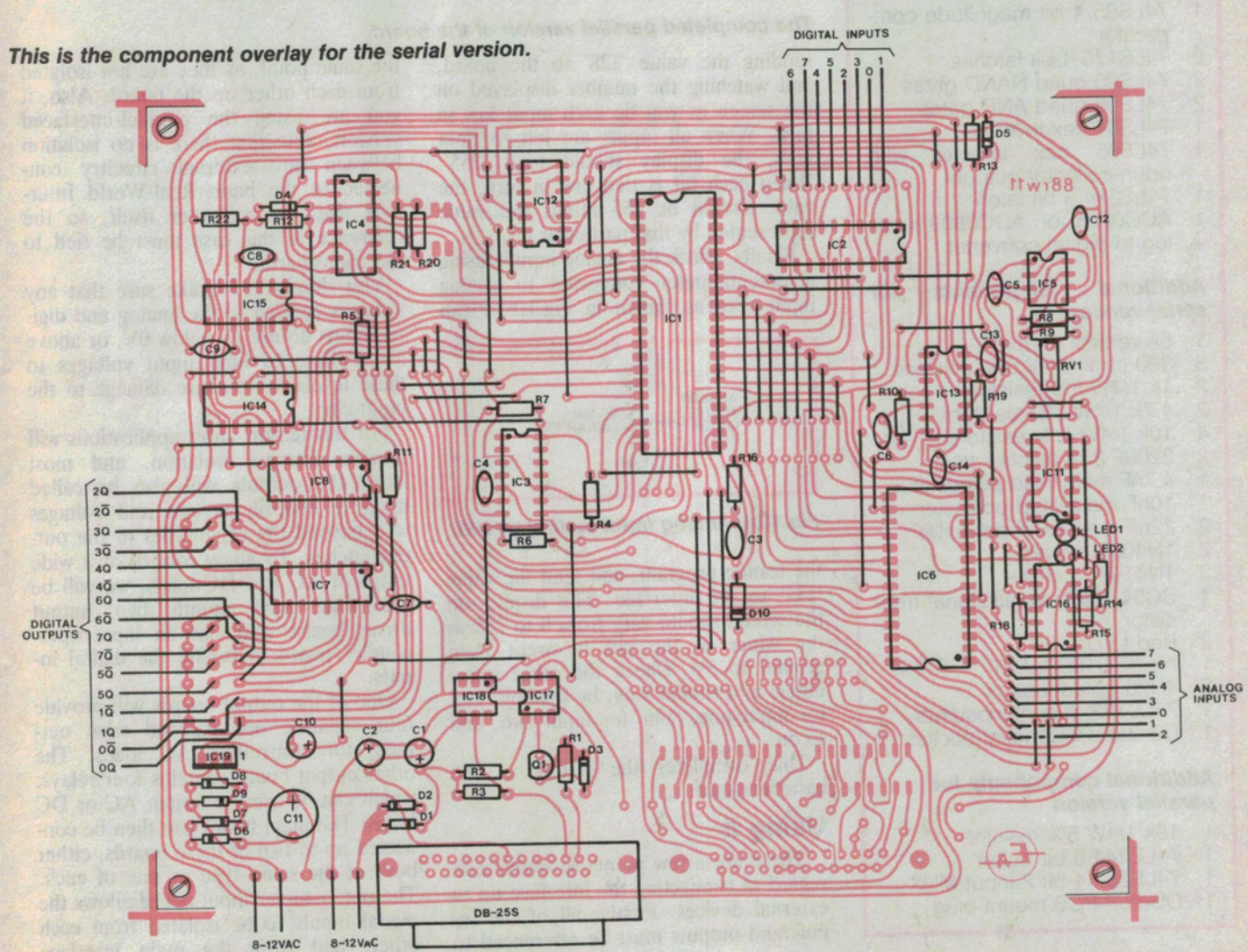


Fig.1(a). Output test circuit.

The digital inputs may be checked by

This is the component overlay for the serial version.



PARTS LIST

1 PCB 191 x 159mm, coded 88rw11
1 Small TO-220 heatsink
1 Power transformer (see text)

Resistors

All 1/4W, 5%: 3 x 1k, 1 x 4.7k, 1 x 8.2k, 1 x 10k, 2 x 22k, 1 x 100k.

Capacitors

2 0.1uF metallised polyester
2 2.2nF metallised polyester
1 10nF metallised polyester
1 10uF 16V tantalum
1 1000uF 16V electrolytic
1 Capacitor according to Table 1 in October issue

Semiconductors

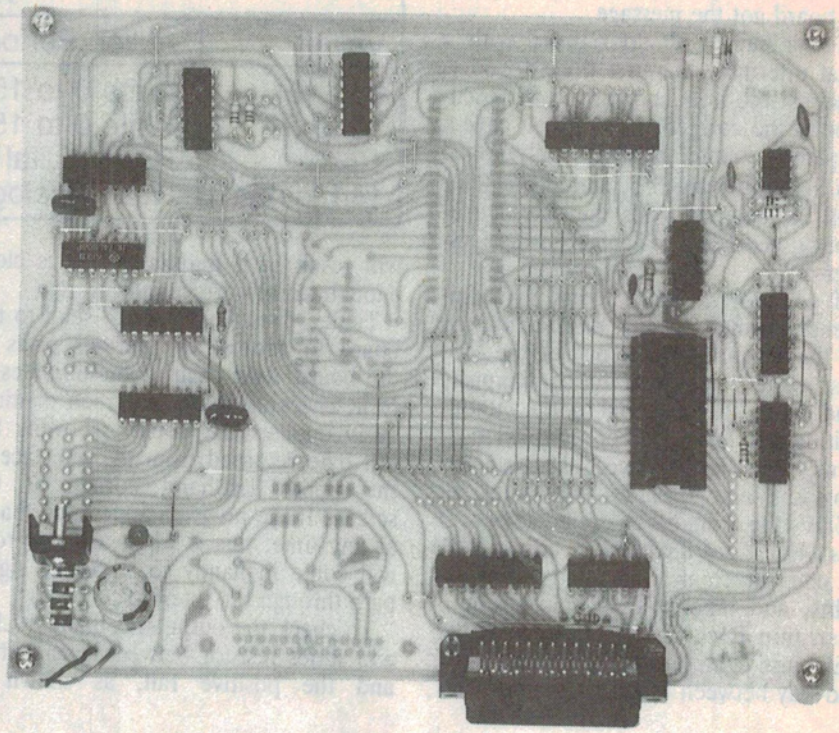
4 1N4002 diodes
1 1N4148 diode
1 7805 regulator
1 555 timer
1 74LS85 4 bit magnitude comparator
2 74LS175 4 bit latches
2 74LS00 quad NAND gates
2 74LS08 quad AND gates
1 74LS04 hex inverter
1 74LS05 hex inverter with open-collector outputs
1 74LS374 8 bit latch
1 ADC0808 or ADC0809 analog to digital converter

Additional components for serial version

1 5k vertical trimpot
3 390 ohm 1/4W 5% resistors
2 1k 1/4W 5% resistor
2 4.7k 1/4W 5% resistors
4 10k 1/4W 5% resistors
2 220uF 25V electrolytics
1 4.7nF metallised polyester
1 10nF metallised polyester
2 22nF metallised polyester
2 1N4002 diodes
3 1N4148 diodes
1 BC547 NPN small-signal transistor
2 Red LEDs
1 AY-3-1015D UART
2 4N28 optocouplers
1 74LS123 dual monostable
1 DB-25S PCB mount socket

Additional components for parallel version

1 10k 1/4W 5% resistor
1 74LS244 8 bit buffer
1 74LS257 4 bit 2 input MUX
1 DB-25P PCB mount plug



The completed parallel version of the board.

sending the value '128' to the board, and watching the number displayed on the screen as you tie each input low in turn. When all inputs are left to float high, the display should read '255'. When each bit is tied low in turn, the value should be 255 minus the value represented by that particular bit.

Finally, check the analog inputs, using a potentiometer connected to analog input zero as shown in Fig.1(b). Run

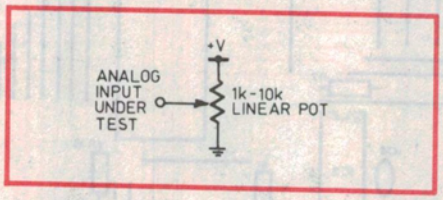


Fig.1(b). Analog input testing circuit.

the testing program, and send the value '144' to the interface. The display on the screen should vary from 0 to 255 as the wiper of the pot is swept from ground to 5V. Check the other seven inputs in a similar way, by sending '145' for input one, '146' for input two, and so on.

That completes the testing of the basic interface.

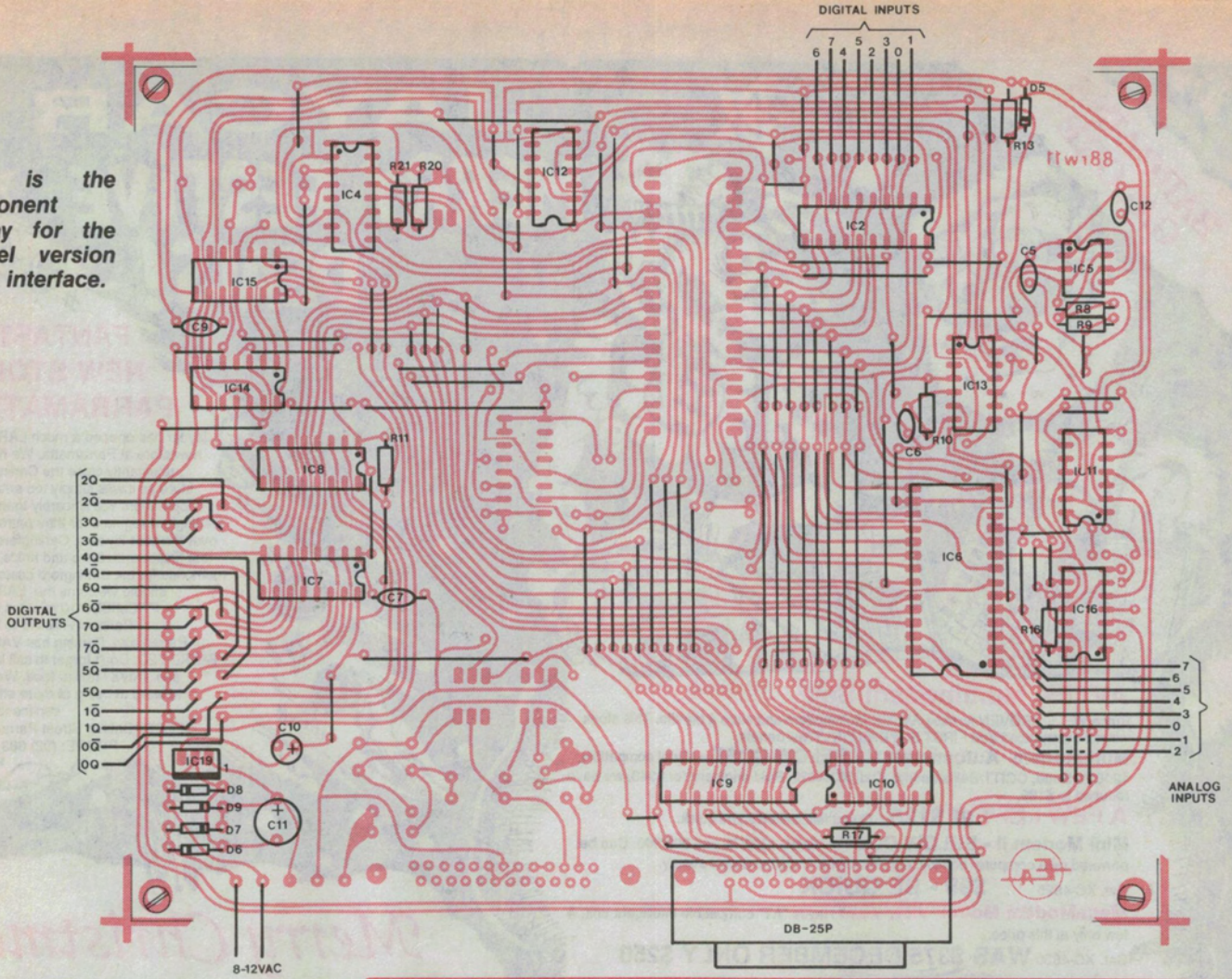
Using it

There are a few points to watch with regard to connecting the interface up to external devices. Firstly, all of the inputs and outputs must be referenced to the same point, as they are not isolated from each other on the board. Also, if you are using the parallel-interfaced version, note that there is no isolation between any external circuitry connected to the basic Real-World Interface and the computer itself, so the reference in this case must be tied to the mains earth.

You should also make sure that any voltages applied to the analog and digital inputs do not go below 0V, or above 5V. Failure to limit input voltages to these levels could cause damage to the input chip.

For this reason, most applications will require external isolation, and most likely the outputs will also be called upon to control currents and voltages well beyond the capabilities of the output latches. To allow control of a wide range of AC and DC loads, we will be presenting next month two output driver boards and also an input opto-coupler board to isolate the digital inputs.

One of the output boards will provide four separate opto-isolated triac outputs, for controlling AC loads. The other output board contains four relays, which can be used to switch AC or DC loads. The main board can then be connected up to two of these boards, either both of the same type or one of each. The opto-coupler input board allows the digital inputs to be isolated from each other, and from the main interface



Here is the component overlay for the parallel version of the interface.

board itself. This input board can accommodate up to eight opto-couplers and associated components.

We have not provided any software to drive the board beyond that used to test the board, as the intended application will have a lot of bearing on what the software has to do, and how well it must do it. For example, in many applications, BASIC routines similar to those used to test the boards may be all that is needed. However, for applications that are more dependent on speed, you may need to use languages such as Pascal, C or assembler.

For data-logging a possible, but perhaps not very obvious solution, is to use something like dBase III. This has the ability to directly manipulate the I/O ports of the computer, to control the interface, and the data can be logged directly into a database file for later processing. For applications that require the recording of large amounts of data at relatively slow rates, such as meteorological observations or say, measuring battery discharge curves, this approach will probably yield the required results for relatively little programming effort.

REAL WORLD INTERFACE PART 2 – Errata

Before commencing construction of the Real World Interface, you should first note a few changes and corrections to the circuit diagram which appeared in part 2.

First of all, the two resistors connected to IC4 should be R20 and R21, not R14 and R15 as shown. Also, they should both be 1k, not 10k as shown. Also note that R6 should be 2.2k, not 100k as shown, and R12 should be 10k, not 22k. In addition, there is another resistor connected from pin 5 of IC14b to ground which was omitted from the diagram in the October issue. This resistor should be labelled R22 and is 10k in value.

The pins of IC15f, labelled 14 and 15 should be 12 and 13 respectively. Also the inverter connected between pin 22 of IC1 and C8 should be labelled IC15b, not IC15e, as shown. The pin numbers for its input and output are 3 and 4 respectively. The pin numbers for the CLK and OC-bar pins on IC2 have been reversed. That is, the CLK line should be labelled pin 11, and the OC-bar line pin 1.

Finally, a small change has been made to the circuit published in the October issue, in order to give the data applied to the input to the UART more time to settle before being strobed in. This delay circuit is inserted between the line on the internal bus labelled "DI STB" and the DS-bar input of the UART.

The extra circuitry is shown below.