Construction project:

Universal "Real World" interface for PCs – 2

This month we present a full description of the circuit for the interface board. The interface includes eight digital outputs, eight digital inputs and eight 8-bit analog inputs.

by MARK CHEESEMAN

The circuit for the controller board can be divided into three major sections: the serial interface, the parallel interface and the "core" components, which are installed irrespective of which interfacing option is used.

A complete controller board consists of the "core" components and the components for *one* interface circuit. The exception to this is when two or more boards are to be daisy-chained together to share a common interface (either serial or parallel). Under no circumstances should *both* interface circuits be installed on the same board.

This is because if both interfaces were present on the same board, the outputs of both interfaces would try to drive the data and address lines on the board simultaneously, causing unpredictable results and possible damage to the circuit.

Internal buses

Data arriving from the computer, via either the serial or parallel port, consists of eight bits, which are split up to form the internal bus control signals for the controller board.

The most significant bit of this data byte becomes the read/write signal, which determines whether the operation to be performed by the controller board is to be an input or an output.

The next three most significant bits form the address bus data for the controller. The two most significant of these (A2 and A1) allow up to four boards to be addressed via a single I/O port on the computer. In this way, each of the boards is assigned a unique 2 bit number, and the state of these two bits determines which of the boards in the system is selected for a particular operation.

A0 selects which input or output device on the selected board is going to be used. For an output operation, this bit determined which of the two four-bit latches (ICs 7 and 8) will be loaded with the data on the data output bus. If an input operation is in progress, a high on this line selects the analog inputs (via IC6), while a low will cause the digital inputs to be latched into the input latch (IC2).

The remaining four data lines from the computer form a four-bit internal data bus, which carries the data itself from the computer to the output latches during an output operation. These four bits are latched into one of the two output latches during an output operation. The lowest three of these lines also select which of the eight analog inputs will be converted to digital information when the ADC is selected. During a digital input operation, these four bits are unused.

When a byte is received via either the serial or parallel interfaces, the eight data bits are placed on the output lines of either the UART (IC1) or data buffer (IC9) respectively. These eight bits consist of the data, address and read/write information from the computer, as outlined above.

To indicate the availability of this data, the UART raises the DAV (data available) line. This triggers IC3, a 74LS123 monostable, which resets the DAV line of the UART by lowering the RDAV pin. The output from this monostable is effectively also the master strobe signal, indicating to the rest of the circuit that the data, address and read/write lines carry valid information.

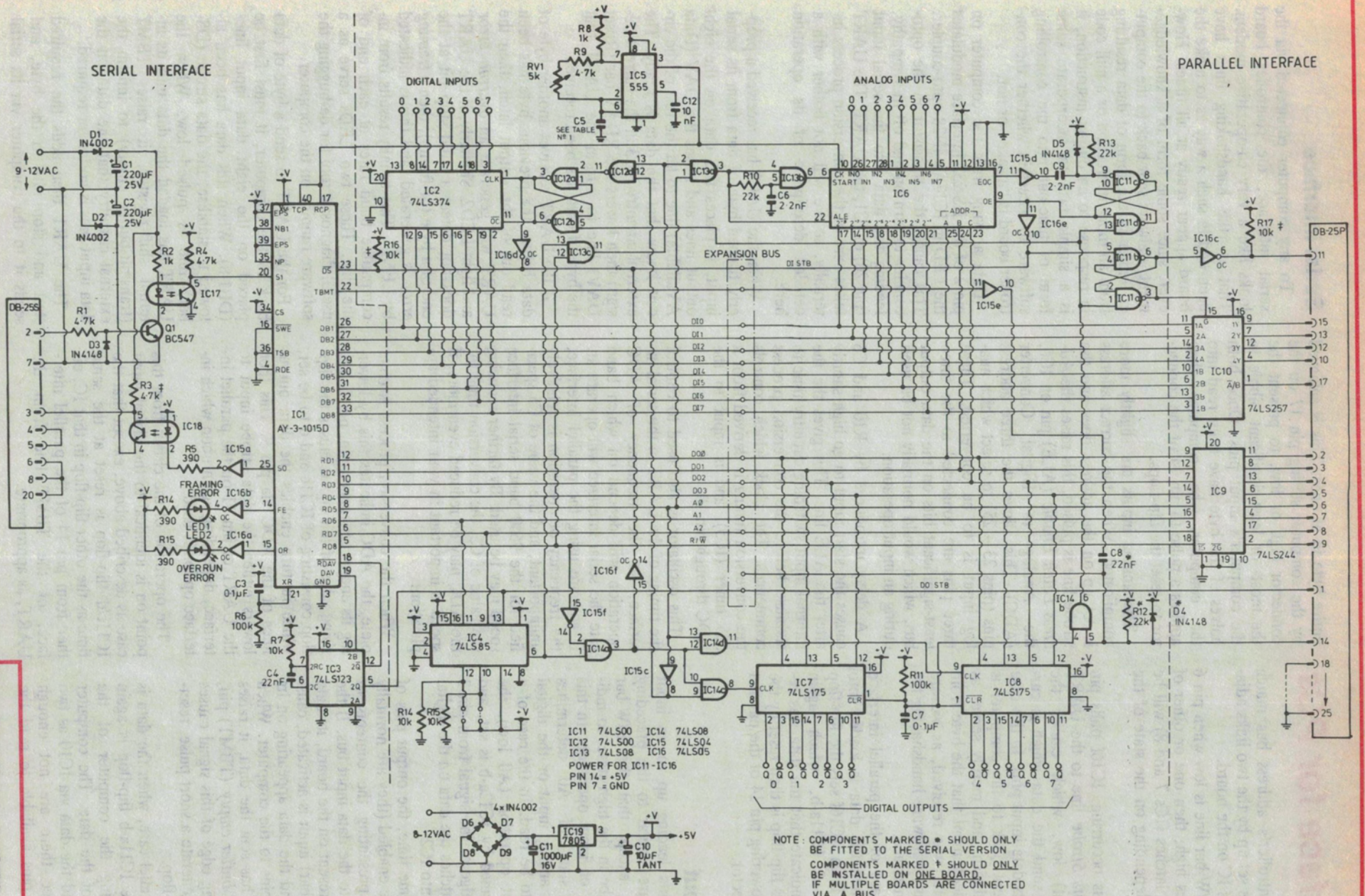
In the case of the parallel interface, this strobe signal comes directly from pin 1 of the interface connector, which is the line normally used for the printer strobe signal. In this case, the duration of the pulse is determined by the software in the computer, and IC3 is therefore not needed.

This strobe signal from either source then goes to IC4, a 74LS85 4-bit magnitude comparator. The output of this IC (pin 6) only goes high when the four bit words applied to each set of inputs are identical. In this way, the output goes high in time with the strobe pulse, but only when the two most significant bits

Use this table to select the value of capacitor C5, to suit the baud rate required for the serial interface version. Use 390pF for the parallel interface.

Table 1: Values for C5 for various serial I/O speeds

Baud Rate	Frequency	C5
300	4800Hz	18nF
600	9600Hz	8.2nF
1200	19.2kHz	3.9nF
2400	38.4kHz	1.8nF
4800	76.2kHz	1.0nF
9600	153.6kHz	390pF



Complete circuit for the 'mother board' section of the real world interface. The section in the centre, between the two dashed lines, is the core circuit. To connect it to your computer you use either the serial interface circuit on the left, or the parallel interface on the right.

of the controller's address bus match the address set up by the two links adjacent to this IC on the board.

If the R/W-bar line is low when pin 6 of IC4 goes high, then one or other of the output latches (ICs 7 and 8) will be selected, depending on the state of the A0 line.

As this is occurring, IC16f pulls pin 23, the *Data Strobe* line to the UART (if present) low, which causes the UART to send the data which is currently on its data input pins. In this case the data sent back to the computer is random rubbish, and merely serves as an acknowledgement that the byte from the computer was received, as we are not using any hardware handshaking for the serial interface.

In the case of the parallel interface version, when the data is loaded into the output latch, IC16f sets the flip-flop consisting of IC11a-b, which again serves to indicate that the data was received. This flip-flop is then reset by the computer lowering pin 14 of the parallel input connector.

Data input

The events leading up to an input operation are similar to the preceding description, except that the R/W-bar line has to be in the high state, to indicate a *read* on input operation. In this case, the state of A0 determines whether an analog input or the digital inputs are to be read. In the case of a digital input operation (A0 low), the flip-flop formed by IC12a-b is set, and the rising edge of the signal from pin 3 of this IC latches the data on the digital input lines into IC2.

At the same time, the output pins of the latch are enabled (they are normally tri-stated), presenting the contents of the latches to the data input bus. If the UART is present on the board, its data strobe (DS-bar) input is activated, causing it to send the data appearing on its data input pins to the computer. When the UART has sent the data, it raises the *transmit buffer empty* (TBMT) pin, and the rising edge of this signal causes C8/R12 to generate a short pulse, resetting the flip-flop.

In the parallel case, when the data is available the IC11a-b flip-flop is again set, informing the computer of the availability of the data. The computer needs to read the data via IC10 as two nibbles, since there are not enough spare input pins available to read the entire byte in at once. This is achieved by the computer setting pin 17 on the connector high or low, to present the the most- or least-significant nibble to the computer's input pins. Once both halves of the byte have been read into the computer, the computer then applies a pulse to pin 14 of the connector to reset the flip-flops.

The analog inputs are slightly more complicated. When the correct address is set up on the address lines and the strobe line is pulsed, this raises the *Address Latch Enable* (ALE) pin on IC6, the Analog to Digital Converter (ADC). This allows the three address pins (pins 23 – 25) to select which analog input is to be converted. These three pins are connected to the three least-significant bits on the data output bus, which would normally not be used during an input operation.

A delay introduced by R10 and C6 causes the start input to go high shortly after the ALE line. This gives the the internal input switching circuitry time to stabilise before the conversion process commences. The clock which controls the conversion timing is derived from a 555 timer (IC5), and is input to the ADC through pin 10.

This oscillator is also used to control the timing of the UART in the serial interface, so the frequency of ADC operation will depend on which baud rate the serial interface will operate at. If you are using the parallel interface, the frequency of operation is relatively unimportant, and the value of C5 specified for the 9600 baud serial interface speed may be used. By further reducing the value of C5, the conversion time for the ADC may be reduced even more, if speed is important in your intended application.

When the conversion process is complete, the ADC indicates this by lowering its end-of-conversion pin, pin 7. The rising edge of this signal causes the flip-flop consisting of IC11c and d to be set, which in turn enables the data output lines of the ADC via pin 9. This then lowers either the data strobe input to the UART or pin 11 on the parallel interface, depending again upon which interface option is used.

The operation of the circuit from this point on is identical to the digital input case as described above, except that the IC11c/d flip-flop is reset at the same time as the other flip-flop in that IC, either from pin 14 of the parallel interface, or the TBMT line from the UART, as appropriate.

Serial interface

To any computer connected via the serial interface, the controller board looks like a very co-operative modem. That is, the handshaking lines have been tied in such a way as to make the board appear ready at all times. However, since some form of acknowledgement is always sent back to the computer, either in the form of data resulting from an input instruction or a null code in response to an output instruction, it is a simple task to incorporate some form of time-out into the controlling software to ascertain whether the controller board is "awake" or not.

Data arriving from the computer on pin 2 of the RS-232 interface modulates the LED in IC17, a 4N28 opto-coupler. This causes the transistor in the opto-coupler to turn on and off in sympathy, feeding the data into the serial input (pin 20) of IC1, the UART. The UART takes care of the conversion process internally, so we will not bother with a detailed description of its operation here.

When the UART has received a properly framed byte of data from its serial input, it places this data on the eight output lines and raises the DAV (Data AVailable) line to let the rest of the circuit know that the data is ready. The external circuitry (IC3 in the current design) then lowers the RDAV-bar (Reset DAV) line to inform the UART that the byte of data has been received.

If the UART receives another byte of data before the previous byte has been read out of the output latch, then the UART will generate an *overrun error* and light LED2. Similarly, if the UART does not detect a valid stop bit at the end of a byte (or frame), then a *framing error* is generated, and this is indicated by LED1 . This usually results from incompatible baud rates at each end of the line. These two LEDs serve as a useful diagnostic tool for de-bugging the software running on the computer.

For the UART to send a byte of data back to the computer, it must first be placed on the eight data input lines (DB1-8). When this data has been allowed to stabilise, the data strobe (DS-bar) line is pulsed low. When the UART has loaded the data into its internal shift-register, it raises TBMT (transmit-buffer empty) to inform the external circuitry that the data on the data input pins is no longer required.

The UART then adds the required start and stop bits to the byte, and sends it to the computer via its serial

output (pin 25). IC18 provides electrical isolation of the serial output line from the computer, so that any computer connected via the serial port of the interface is protected from the ravages of whatever may be connected to the Real-World Interface.

Timing for the serial interface is derived from a 555 (IC5) running as an astable oscillator. The frequency of oscillation of this oscillator is set to 16 times the baud rate of the link between the interface and the computer. The broad frequency range is set by the appropriate choice of C5, according to Table 1, and fine adjustments are made using RV1. Once RV1 has been set initially, no further adjustment should be necessary, as this form of oscillator is more than stable enough for byte-orientated serial communications.

The power supply for the interface is quite conventional. Low voltage AC from an external transformer is rectified by the bridge made up of diodes D6 to D9. It is then filtered by a 1000uF electrolytic (C11), and regulated down to 5V by IC19.

The serial-interface version has an additional power supply to power the interface circuitry, and yet enable it to be electrically isolated from the rest of the interface circuit. A separate transformer supplies the power to this portion of the circuit, as we found that two single-secondary transformers could be obtained for less than a dual-secondary unit.

This power-supply is a dual half-wave set-up generating positive and negative 12-15 volt rails from a 9-12 volt AC input. These rails ensure that the serial data coming from the controller meets true RS-232C specifications. While simpler power-supply configurations are possible, there is a significant chance of the interface not working with computers that are fussy about the voltage levels of the received data.

Some computers may make suitable voltages available at the RS-232C port, which may be used to power the serial interface circuitry. If this is the case, then the relevant power supply components (D1, D2, C1, C2 and the second transformer) may be omitted, and the power derived from the computer itself. This has the additional advantage that power for the serial interface circuitry will automatically be applied whenever the computer is turned on.

Next month we will present the constructional details and PCB artwork for the interface.