Teach-In 2018

Get testing! – electronic test equipment and measurement techniques

Part 8: Digital measurements

by Mike Tooley

Welcome to *Teach-In 2018: Get testing! – electronic test equipment and measurement techniques*. This *Teach-In* series will provide you with a broad-based introduction to choosing and using a wide range of test gear, how to get the best out of each item and the pitfalls to avoid. We'll provide hints and tips on using, and – just as importantly – interpreting the results that you get. We will be dealing with familiar test gear as well as equipment designed for more specialised applications.

Our previous *Teach-In* series have dealt with specific aspects of electronics, such as PICs (*Teach-In 5*), Analogue Circuit Design (*Teach-In 6*) or popular low-cost microcontrollers (*Teach-In 7* and *8*). The current series is rather different because it has been designed to have the broadest possible appeal and is applicable to all branches of electronics. It crosses the boundaries of analogue and digital electronics with applications that span the full range of electronics – from a single-stage transistor amplifier to the most sophisticated microcontroller system. There really is something for everyone in this series!

Each part includes a simple but useful practical *Test gear project* that will build into a handy gadget that will either extend the features, ranges and usability of an existing item of test equipment or that will serve as a stand-alone instrument. We've kept the cost of these projects as low as possible and most of them can be built for less than £10 (including components, enclosure and circuit board).

This month

In this penultimate part of our current *Teach-In* series, *In theory* will provide an overview of the way in which digital data is represented in a digital system and how it can be captured and analysed. *Get it right!* will help you avoid some potential pitfalls when making digital measurements and will provide useful hints and tips to help you improve the accuracy and relevance of your measurements. Finally, our *Test gear project* features a handy logic probe that can be used with a wide range of different types of digital logic.

In theory: Digital signals and logic levels

This month, we turn our attention to the tests and measurements that need to be made in a digital rather than analogue world. Since we are only dealing with two logical states – variously referred to as 'on' or 'off', 'high' or 'low', and 'logic 1' or 'logic 0' – this should be quite easy. In practice, however, this can be rather different since the logical states that we need to view are often rapidly changing and may need to be captured so that we can view and make sense of them. Even more challenging is that we often need to capture data on multiple signal lines so that we can view and analyse the relationship between signals within the time domain. This calls for some sophisticated but not necessarily expensive items of test equipment. We will start by looking at how digital signals are represented before moving on to describe ways in which they can be measured and analysed.

Representing digital signals

Logic levels

Logic levels are simply the range of voltages used to represent the logical states 0 and 1. The logic levels for CMOS differ markedly from those associated with TTL. CMOS devices are usually able to operate over a wide range of supply voltage (from 3V to 15V for most standard devices) and their logic levels are relative to the supply voltage used (with roughly one third and two thirds of the supply voltage marking the upper boundaries of logic 0 and the lower boundary of logic 1 respectively). By contrast, the logic levels associated with standard TTL devices tend to be more precise and absolute. Table 8.1 shows the normally accepted range of values for conventional 5V CMOS, TTL, and low-voltage (LV) TTL/CMOS devices.

Noise margin

Logical states need to stand out against other signals that may be present in an electronic circuit. In other words, a logic 1 should unambiguously be a logic 1. Likewise, a logic 0 should definitely be a logic 0. Of concern here is the need for sufficient separation between the two defined states. In other words, there should be a clear boundary between them. The difference between the logic 0 and logic 1 boundaries is known as the *noise margin*; it is an important parameter associated with any logic family. Put simply, noise margin is a measure of the ability of the device to reject noise; the larger the noise margin the better its ability to perform in an environment in which noise is present (and where the superimposed noise may be sufficient to cross the logic level boundaries).

Noise margin is usually defined in terms of the difference between the minimum

Table 8.1 Logic levels

Logic state	CMOS 5V	TTL 5V	LV TTL/CMOS 3.3V
Logic 1 (high)	>3.5V	>2V	>2V
Logic 0 (low)	<1.5V	< 0.8V	<0.8V
Threshold	2.5V	1.5V	1.5V
Indeterminate region	1.5V to 3.5V	0.8V to 2V	0.8V to 2V

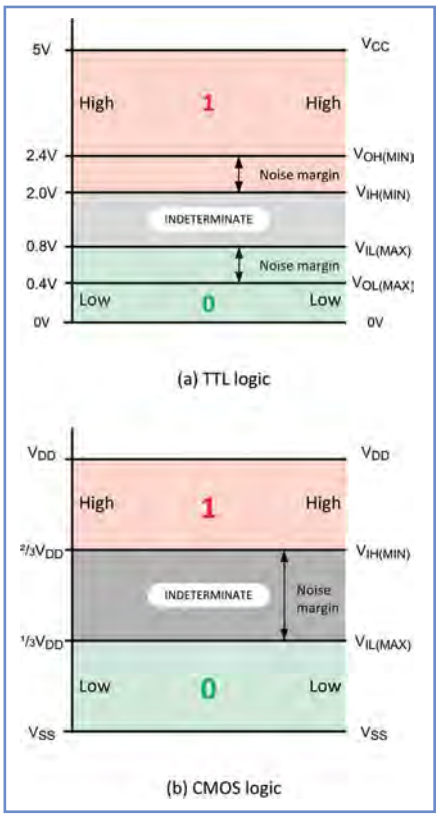

Fig.8.1. Comparison of logic levels and noise margins for standard 5V TTL and CMOS devices

values of high state output and high state input voltage and the maximum values of low state output and low state input voltage. Hence:

Noise margin = $V_{OH(MIN)} - V_{IH(MIN)}$

or

Noise margin = $V_{OL(MAX)} - V_{IL(MAX)}$

Where $V_{OH(MIN)}$ is the minimum value of high-state (logic 1) output voltage, $V_{IH(MIN)}$ is the minimum value of high-state (logic 1) input voltage, $V_{OL(MAX)}$ is the maximum value of low-state (logic 0) output voltage, and $V_{IL(MAX)}$ is the minimum value of low-state (logic 0) input voltage. The noise margin for the legacy 7400 TTL series is typically 400mV, while for 5V CMOS it is approximately 2V, as illustrated in Fig.8.1.

In practice, you are likely to encounter a variety of sub-families of the original 'standard' TTL and CMOS logic families. These include CMOS devices compatible with TTL (HCT and FCT) as well as low-voltage (LV) logic devices. The chart shown in Fig.8.2 provides a useful comparison of the threshold voltages of these different families.

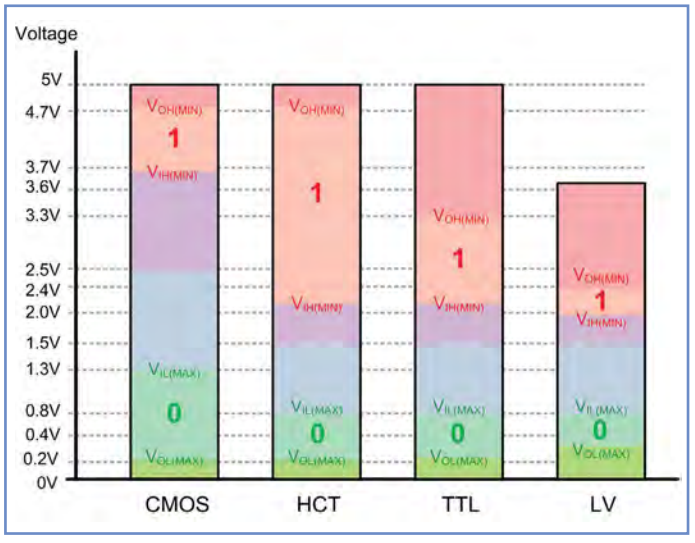

Fig.8.2. Comparison chart of common logic families

Logic gates

Basic logical operations (eg, AND, OR) are carried out by means of individual circuits known as 'gates'. The symbols for some basic logic gates are shown, together with their truth tables in Fig.8.3. The action of each of the basic logic gates is summarised below. Note that while inverters and buffers each have only one input, exclusive-OR and exclusive-NOR gates have two inputs and the other basic gates (AND, OR, NAND and NOR) are commonly available with up to eight inputs (but for these there is no theoretical limit).

Buffers

Buffers do not affect the logical state of a digital signal (ie, a logic 1 input results in a logic 1 output and a logic 0 input results in a logic 0 output). Buffers are normally used to provide extra current drive at the output but can also be used to regularise the logic levels present at an interface.

Inverters

Inverters are used to complement the logical state (ie, a logic 1 input results in a logic 0 output and vice versa). Inverters also provide extra current drive and, like buffers, are used in interfacing applications where they provide a means of regularising logic levels present at the input or output of an LSI device.

AND gates

These gates will only produce a logic 1 output when all inputs are simultaneously at logic 1. Any other input combination results in a logic 0 output.

OR gates

These gates will produce a logic 1 output whenever one or more of their inputs are at logic 1. Putting this another way, an OR gate will only produce a logic 0 output whenever all inputs are simultaneously at logic 0.

NAND gates

These gates will only produce a logic 0 output when all inputs are simultaneously at logic 1. Any other input combination will produce a logic 1 output. A NAND gate, therefore, is nothing more than an AND gate with its output inverted. The circle shown at the output denotes this inversion.

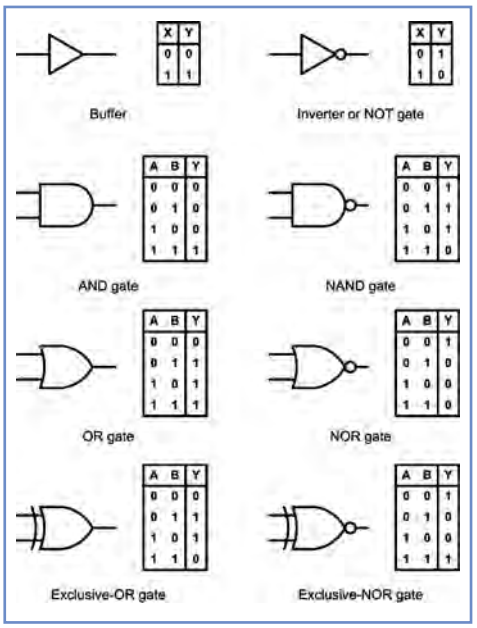

Fig.8.3. Logic gate symbols and truth tables

NOR gates

These gates will only produce a logic 1 output when all inputs are simultaneously at logic 0. Any other input combination will produce a logic 0 output. A NOR gate, therefore, is simply an OR gate with its output inverted. A circle is again used to indicate inversion.

Exclusive-OR gates

Exclusive-OR gates Sometimes written as 'XOR') will produce a logic 1 output whenever either one of the inputs is at logic 1 and the other is at logic 0. Exclusive-OR gates produce a logic 0 output whenever both inputs have the same logical state (ie, when both are at logic 0 or both are at logic 1).

Monostable

A logic device which has only one stable output state is known as a 'monostable'. The output of such a device is initially at logic 0 (low) until an appropriate level change occurs at its trigger input. This level change can be from 0 to 1 (positive-edge trigger) or 1 to 0 (negative-edge trigger) depending upon the particular monostable device or configuration. Upon receipt of a valid trigger pulse the output of the monostable changes state to logic 1. Then, after a time interval determined by external C-R timing components, the output reverts to logic 0. The device then awaits the arrival of the next trigger. A typical application for a monostable device is in stretching a pulse of very short duration.

Bistables

The output of a bistable can take one of two stables states, either logic 0 or

Table 8.2 Characteristics of logic levels

Characteristic	Logic family			
	74	74LS	74HC	40BE
Maximum supply voltage	5.25V	5.25V	5.5V	18V
Minimum supply voltage	4.75V	4.75V	4.5V	3V
Static power dissipation (mW per gate at 100kHz)	10	2	negligible	negligible
Dynamic power dissipation (mW per gate at 100kHz)	10	2	0.2	0.1
Typical propagation delay (ns)	10	10	10	105
Maximum clock frequency (MHz)	35	40	40	12
Speed-power product (pJ at 100kHz)	100	20	1.2	11
Minimum output current (mA at VOUT = 0.4V)	16	8	4	1.6
Fan-out (LS loads)	40	20	10	4
Maximum input current (mA at VIN = 0.4V)	–1.6	–0.4	0.001	–0.001

logic 1. Once *set*, the output of a bistable will remain at logic 1 for an indefinite period until the bistable is *reset*, at which time the output will revert to logic 0. A bistable thus constitutes a simple form of *memory cell* because it will remain in its latched state (either *set* or *reset*) until commanded to change its state (or until the supply is disconnected). Popular forms of bistable include R-S, D and J-K types.

R-S bistables

The simplest form of bistable is the R-S bistable. This device has two inputs, SET and RESET, and complementary outputs, Q and $\overline{Q}$. A logic 1 applied to the SET input will cause the Q output to become (or remain at) logic 1 while a logic 1 applied to the RESET input will cause the Q output to become (or remain at) logic 0. In either case, the bistable will remain in its SET or RESET state until an input is applied in such a sense as to change the state.

D-type bistables

The D-type bistable has two principal inputs; D (standing variously for data or delay) and CLOCK (CK). The data input (logic 0 or logic 1) is clocked into the bistable such that the output state only changes when the clock changes state. Operation is thus said to be *synchronous*. Additional subsidiary inputs (which are invariably active low) are provided, which can be used to directly set or reset the bistable. These are usually called PRESET (PR) and CLEAR (CLR). D-type bistables are commonly used as data latches (a simple form of memory) and as binary dividers.

Fig.8.4. A typical logic probe

J-K bistables

J-K bistables are the most sophisticated and flexible of the bistable types, and they can be configured in various ways including binary dividers, shift registers, and latches. J-K bistables have two clocked inputs (J and K), two direct inputs (PRESET and CLEAR), a CLOCK (CK) input, and outputs (Q and $\overline{Q}$). As with R-S bistables, the two outputs are complementary (ie, when one is 0 the other is 1, and vice versa). Similarly, the PRESET and CLEAR inputs are invariably both active low (ie, a 0 on the $\overline{\text{PRESET}}$ input will set the Q output to 1 whereas a 0 on the $\overline{\text{CLEAR}}$ input will set the Q output to 0).

Logic gate characteristics

Table 8.2 summarises the key characteristics of the original members of the TTL family with the equivalent CMOS logic. There are some important points worth noting:

- CMOS devices are static sensitive and require appropriate anti-static handling techniques
- CMOS logic operates over a much larger range of supply voltage than conventional TTL
- CMOS devices tend to be much slower than their TTL counterparts
- TTL devices consume significantly more power than their CMOS counterparts
- TTL devices are capable of driving more loads than CMOS devices
- CMOS devices require negligible input current and impose minimal load on an input.

Logic probes

The simplest and most convenient method of examining logic states involves the use of a logic probe. When making measurements on digital circuits, this handy gadget is much easier to use than a digital multimeter or an analogue oscilloscope. It comprises a hand-held probe fitted with LEDs that indicate the logical state of its probe tip.

Unlike a digital multimeter, a logic probe can usually distinguish between lines which are actively pulsing, and those that are in a permanently *tri-state* (effectively disconnected) condition. In the case of a line which is being pulsed, the logic 0 and logic 1 indicators will both be illuminated (though not necessarily with the same brightness) whereas, in the case of a tri-state line neither indicator should be constantly illuminated.

Logic probes generally also provide a means of displaying pulses having a very short duration, which may otherwise go undetected. A *pulse stretching* circuit is usually incorporated within the probe circuitry so that an input pulse of very short duration is elongated sufficiently to produce a visible indication on a separate pulse LED.

Logic probes invariably derive their power supply from the circuit under test and are connected by means of a short length of twin flex fitted with insulated crocodile clips (see Fig.8.4). Note that it is essential to ensure that the supply voltage is the same as that used to supply the logic devices on test.

A typical logic probe circuit suitable for home construction is shown in Fig.8.5. This circuit uses a dual comparator to sense the logic 0 and logic 1 levels and a timer, which acts as a monostable pulse stretcher to indicate the presence of a pulse input rather than a continuous logic 0 or logic 1 condition. Typical logic probe indications and waveforms are shown in Fig.8.6.

Fig.8.7 shows how a logic probe can be used to check a simple arrangement of logic gates. The probe is moved from node to node and the logic level is displayed and compared with the expected level. Fig.8.8 shows how a logic probe can be used to test a much more complex circuit in the shape of a modern Mini-ITX computer system.

Logic pulsers

It is sometimes necessary to simulate the logic levels generated by a peripheral

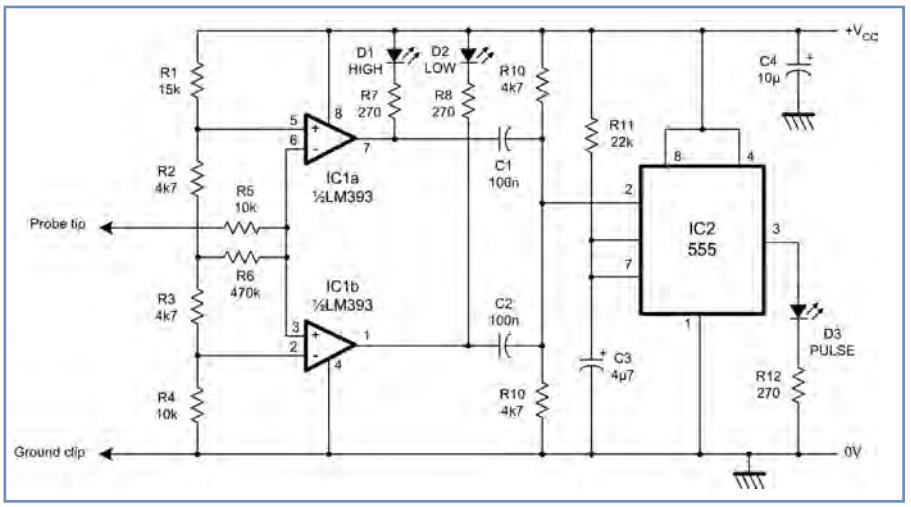

Fig.8.5. A logic probe circuit suitable for home construction

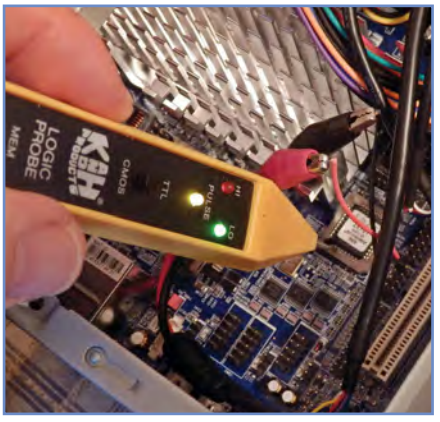

Fig.8.8. Using a logic probe to check the signals present on the BIOS chip of a Mini-ITX motherboard. The probe indicates a signal that is mostly low but also pulsing high (see Fig.8.6)

device or sensor. A permanent logic level can easily be generated by pulling a line up to the logic supply by means of 1kΩ resistor or by temporarily tying a line down to 0V. However, on other occasions, it may be necessary to simulate a pulse rather than a permanent logic state and this can be achieved by means of a hand-held logic pulser.

LED INDICATOR			STATE INDICATED	WAVEFORM
LOW	PULSE	HIGH		
OFF	OFF	ON	Steady logic 1	
ON	OFF	OFF	Steady logic 0	
OFF	OFF	OFF	Open circuit or undefined level	
OFF	BLINK	OFF	Pulse train of near 50% duty cycle at >1MHz	
ON	BLINK	ON	Pulse train of near 50% duty cycle at <1MHz	
OFF	BLINK	ON	Pulse train of high mark:space ratio	
ON	BLINK	OFF	Pulse train of low mark:space ratio	

Fig.8.6. Typical logic probe indications

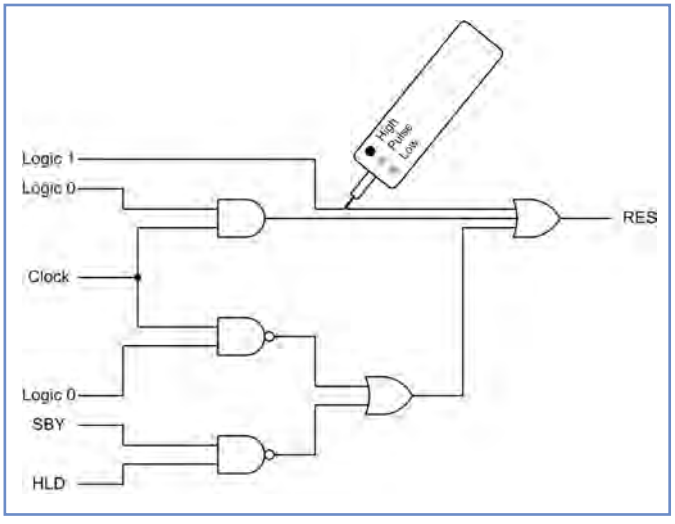

Fig.8.7. Using a logic probe to check a basic logic gate arrangement

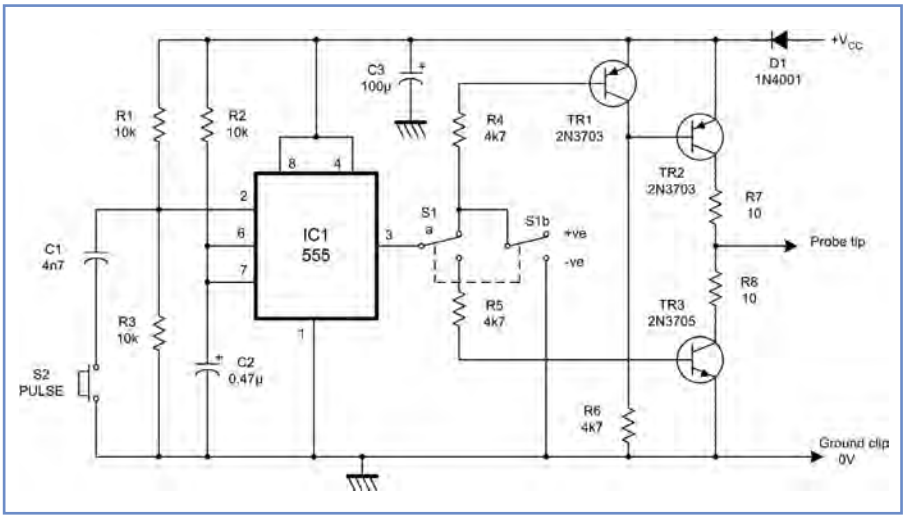

Fig.8.9. A simple logic pulser suitable for home construction

A logic pulser provides a means of momentarily forcing a logic level transition into a circuit regardless of its current state and thus overcomes the need to disconnect or de-solder any of the devices. The polarity of the pulse (produced at the touch of a button) is adjusted so that the node under investigation is momentarily forced into the opposite logical state. During the period before the button is depressed and for the period after the pulse has been completed, the probe tip adopts a tri-state (high impedance) condition. Hence the probe does not permanently affect the logical state of the point in question.

Logic pulsers derive their power supply from the circuit under test in the same manner as logic probes. Here again, it is essential to use the correct logic supply voltage.

A typical logic pulser circuit is shown in Fig.8.9. The circuit comprises a 555 monostable pulse generator triggered from a push-button. The output of the pulse generator is fed to a complementary transistor arrangement in order to make it fully TTL-compatible. As with the logic pulser, this circuit derives its power from the circuit under test.

Fig.8.10 shows an example of the combined use of a logic pulser and a logic probe for testing a simple J-K bistable. The logic probe is used to check the initial state of the Q and $\overline{Q}$ outputs of the bistable, as shown in Fig.8.10 (a) and (b). Note that the Q and $\overline{Q}$ outputs should be complementary. Next, the logic pulser is applied to the clock (CK) input of the bistable (see Fig.8.10(c)) and the Q output is checked using the logic probe. The application of a pulse (using the trigger button) should cause the Q output of the bistable to change state (see Fig.8.10 (d)).

Serial data communication

With anything more than the most basic logic application there's a need for digital data to be exchanged between participating devices. For example, a microcontroller, LCD display and a wide variety of sensors can all be linked

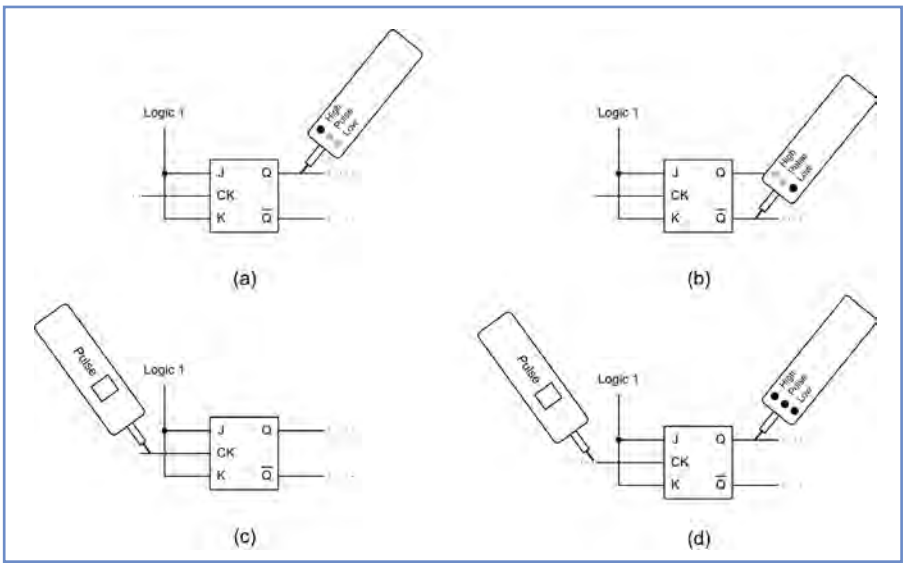

Fig.8.10. A simple logic pulser suitable for home construction

together using one of the popular serial bus connections based on one or more of today's popular and universally available standards such as RS-232, SPI, I²C, and USB).

Serial data communication involves sending a stream of bits, one after another, along a transmission path. Since the data present on a microprocessor bus exists primarily in parallel form, serial I/O techniques are somewhat more complex than those used for simple parallel input and output; serial input data must be converted to parallel (byte wide) data in a form which can be presented to the bus. Conversely, serial output data must be produced from the parallel data present on the internal data bus.

Serial data may be transferred in either synchronous or asynchronous mode. In the former case, transfers are carried out in accordance with a common clock signal (the clock must be available at both ends of the transmission path). Asynchronous operation, on the other hand, involves transmission of data in small packets; each packet containing the necessary information required to decode the data that it contains. Clearly this technique is more complex, but it has the considerable advantage that a commonly available clock signal is not required.

As with programmable parallel I/O devices, a variety of different names are used to describe programmable serial I/O devices, but the asynchronous communications interface adaptor (ACIA) and universal asynchronous receiver/transmitter (UART) are both commonly encountered in serial data communications. Signal connections commonly used with serial I/O devices include:

- *Dn:* Data input/output lines to/from the internal bus
- *RXD:* Received Data (incoming serial data)
- *TXD:* Transmitted Data (outgoing serial data)
- *CTS:* Clear To Send. This (usually active low) signal is taken low by the peripheral when it is ready to accept data from the microprocessor system
- *RTS:* Request To Send. This (usually active low) signal is taken low by the microprocessor system when it is about to send data to the peripheral.

With simple systems (including most popular microcontrollers) signals from serial I/O devices are invariably logic compatible. It should be noted that in general, such signals are unsuitable for anything other than short distance transmission. Reliable data transmission over a greater distance may require specialised line drivers and receivers to provide buffering and level shifting. In noisy environments it can also be advantageous to use balanced transmission using differential signals.

The RS-232D interface is a well-established standard for serial communication between microcontrollers and a wide range of other devices. The original standard dates back to 1987 and is in accordance with international standards CCITT V24, V28 and ISO IS2110. One notable advantage of the RS-232D standard is that it incorporates facilities for loop-back testing, in which data can be sent back to an originating device by looping the TXD line back to the RXD line (see later).

- *Data (TXD, RXD):* RS-232 provides for two independent serial data channels (described as primary and secondary). Both channels provide for full duplex operation (ie, simultaneous transmission and reception). Note that, in practice, both channels are often not used.
- *Handshake control (RTS, CTS):* handshake signals provide the means by which the flow of serial data is controlled, allowing, for example, a DTE to open a dialogue with the DCE prior to actually transmitting data over the serial data path.
- *Timing (TC, RC):* for synchronous (rather than the more usual asynchronous) mode of operation, it is necessary to pass clock signals between the devices. These timing signals provide a means of synchronising the received signal to allow successful decoding.

In practice, few RS-232 implementations make use of the secondary channel featured in the original specification and, since asynchronous (non-clocked) operation is almost invariably used with microcomputer systems, only eight or nine of the original 25 signal lines are regularly used. These lines have the functions shown in Table 8.3.

In asynchronous RS-232 systems, data is transmitted asynchronously as a series of small packets. Each packet represents a single ASCII (or control) character and it must contain sufficient information for the packet to be decoded without the need for a separate clock signal.

ASCII characters are represented by seven bits. The upper-case letter 'A', for example, is represented by the seven-bit binary word 1000001. To send the letter 'A' via RS-232 extra bits must be added to indicate the start and end of the data packet. These are known as the start and stop bits respectively. In addition, we may wish to include a further bit to provide a simple parity-error detecting facility.

Let's look at an example where there is one start bit, seven data bits, one parity bit and two stop bits. The start of the data packet is signaled by the start bit, which is always low irrespective of the contents of the packet. The seven data bits representing the ASCII character follow the start bit. A parity bit is added to make the resulting number of 1s in the group either odd (odd parity) or even (even parity). Finally, two stop bits are added. These are both high. The TTL representation of this character is shown in Fig.8.11 (a).

Table 8.3 Nine-pin RS-232 configuration

Pin	Designation	Function
1	FG	Ground connection
2	TXD	Serial Transmitted data
3	RXD	Serial Received data
4	DTR or RTS	Data Terminal Ready or Request To Send
5	CTS	Clear To Send
6	DSR	Data Set Ready
7	SG	Signal Ground
8	DTR	Data Terminal Ready
9	RI	Ring Indicator

Important note: Not all signals are implemented with current equipment and some pins may be used for different functions. For example, pin-9 is sometimes used for a positive logic supply voltage.

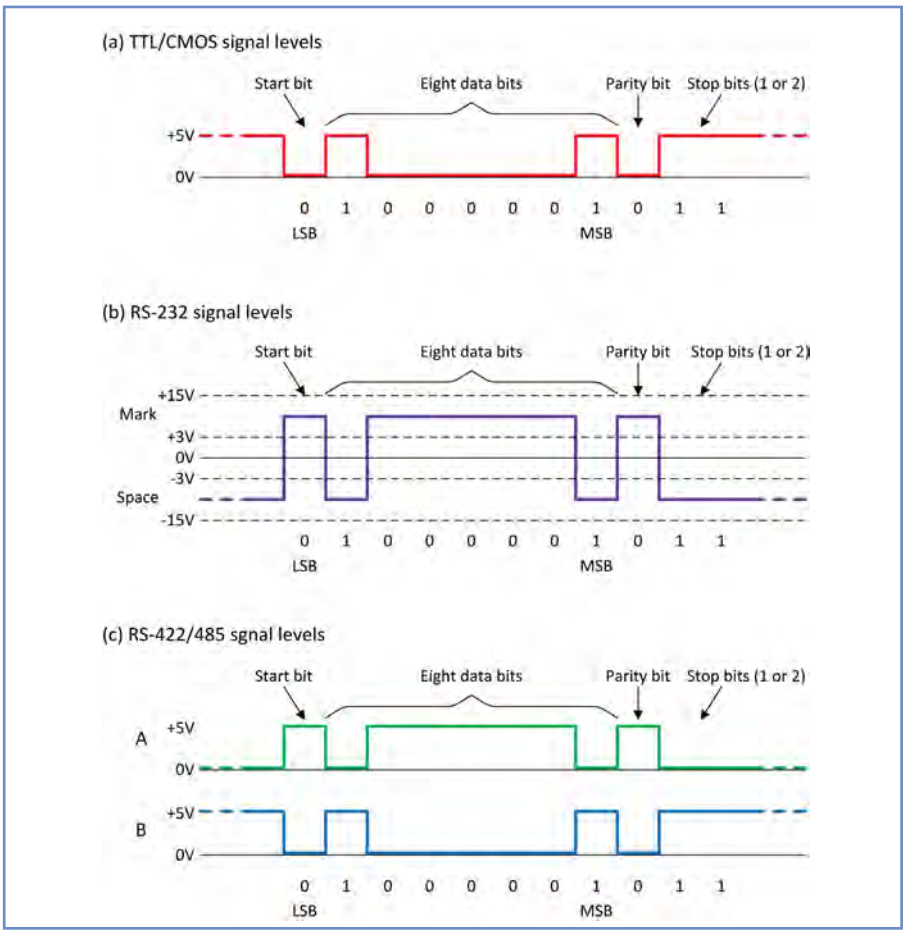

Fig.8.11. Serial data representation

The complete asynchronously transmitted data word thus comprises eleven bits (note that only seven of these actually contain data). In binary terms the word can be represented as: 01000001011. In this example, even parity has been used and thus the ninth (parity bit) is a 0. One of the most commonly used RS-232 schemes involves eight data bits, no parity bit and one stop bit. This is commonly referred to as '8N1'.

The voltage levels employed in a true RS-232 data interface are markedly different from those used within a microcomputer system. A positive voltage (of between +3V and +25V) is used to represent a logic 0 (or SPACE) while a negative voltage (of between –3V and –25V) is used to represent a logic 1 (or MARK). The line signal corresponding to the ASCII character 'A' is shown in Fig.8.11 (b). The level shifting (from TTL to RS-232 signal levels, and vice versa) is usually accomplished using line drivers and line receivers.

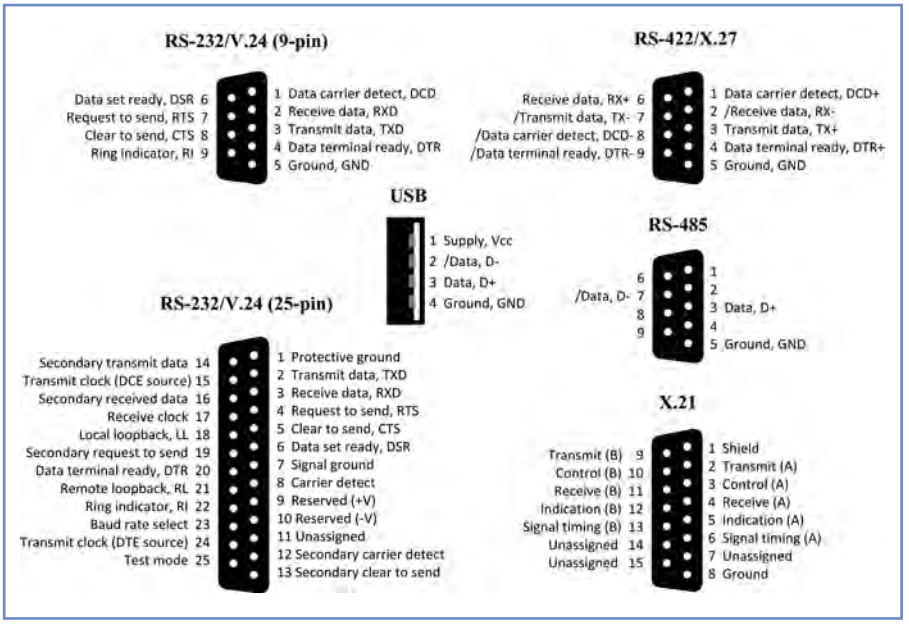

Fig.8.12. Pin connections used for some popular data communication interfaces

Other standards

To overcome some of the limitations of the original RS-232 specification several further standards have been introduced. These generally provide for better line matching, increased distance capability and faster data rates. Notable among these systems are RS-422 (a balanced system which caters for a line impedance as low as 50Ω), RS-423 (an unbalanced system which will tolerate a line impedance of 450Ω minimum), and RS-449 (a very fast serial data standard that uses several modified circuit functions and a 37-way D connector).

RS-422

The RS-422 interface is a balanced system (differential signal lines are used) that employs lower line voltage levels than those used with RS-232. SPACE is represented by a line-voltage level in the range +2V to +6V, while MARK is represented by a line-voltage level in the range, 2V to –6V (see Fig.8.11). RS-422 caters for a line impedance of as low as 50Ω and supports data rates up to 10Mbps.

RS-423

Unlike RS-422, RS-423 employs an unbalanced line configuration (a single signal line is used in conjunction with signal ground). Line voltage levels of +4V to +6V and –4V to –6V represent SPACE and MARK respectively and the standard specifies a minimum line terminating resistance of 450Ω. RS-423 supports a maximum data rate of 100kbps.

RS-449

The RS-449 interface is a further enhancement of RS-422 and RS-423; it caters for data rates up to 2Mbps and provides for upward compatibility with RS-232. Ten extra circuit functions have been provided, while three of the original interchange circuits have been abandoned. In order to minimise confusion, and since certain changes have been made to the definition of circuit functions, a completely new set of circuit abbreviations has been developed. In addition, the standard requires 37-way and 9-way D-connectors, the latter being necessary where use is made of the secondary channel interchange circuits.

Data communication test equipment

Several specialised test instruments and accessories are available for testing data communication equipment, including the following items.

Patch boxes

These low-cost devices facilitate the cross connection of RS-232 (or equivalent) signal lines. The equipment is usually fitted with two D-type connectors (or ribbon cables fitted with a plug and socket) and all lines are brought out to a patching area into which links may be plugged. In use, these devices are connected in series

with the RS-232 serial data path and various patching combinations are tested until a functional interface is established. If desired, a dedicated cable may then be manufactured in order to replace the patch box.

Gender changers

These normally comprise an extended RS-232 connector that has a male connector at one end and a female connector at the other. Gender changers permit mixing of male and female connector types (note that the convention is male at the DTE and female at the DCE).

Null modems

Like gender changers, these devices are connected in series with an RS-232C serial data path. Their function is simply that of changing the signal lines so that a DTE is effectively configured as a DCE. Null modems can easily be set up using a patch box or manufactured in the form of a dedicated null-modem cable.

Line monitors

These display the logical state (in terms of MARK or SPACE) present on the most commonly used data and handshaking signal lines. LEDs provide the user with a rapid indication of which signals are present and active within the system.

Breakout boxes

These provide access to the signal lines and invariably combine the features of patch box and line monitor. In addition, switches or jumpers are usually provided for linking lines on either side of the box. Connection is almost invariably via two 25-way ribbon cables terminated with connectors.

Oscilloscopes

An oscilloscope can be used to display waveforms of signals present on data lines. It is thus possible to detect the presence of noise and glitches as well as measuring signal voltage levels and rise and fall times. A compensated (×10) oscilloscope probe will normally be required in order to minimise distortion caused by test-lead reactance. A digital storage facility can be invaluable when displaying transitory data.

Interface testers

These are somewhat more complex than simple breakout boxes and generally incorporate facilities for forcing lines into MARK or SPACE states, detecting 'glitches', measuring baud rates, and also displaying the format of data words. Such instruments are, not surprisingly, rather expensive.

Multimeters

A general-purpose multimeter (see Part 1) can be useful when testing static line voltages, cable continuity and terminating resistances. A standard multi-range digital instrument will be adequate for most applications, and an audible continuity testing range can be useful when checking data cables.

Universal Serial Bus (USB)

Offering true plug-and-play capability coupled with high data rates, the universal serial bus (USB) has become the de-facto standard for interconnecting a wide range of microcontrollers and computers to an equally wide range of peripheral devices, sharing the available bandwidth through a host-scheduled, token-based protocol. In a conventional USB connection, the USB data (D+ and D–) and power (V_{BUS} and GND) are carried using a four-wire shielded cable. V_{BUS} is nominally +5V at the source, and cable lengths can be up to several metres. To guarantee input voltage levels and proper termination impedance, biased terminations are normally used at each end of the cable.

One of the advantages of USB over other bus systems is its ability to support hot-connection and hot-disconnection from the bus. This important feature requires that the host's system software is not only able to recognise the connection and disconnection of devices, but is also able to reconfigure the system dynamically. All modern operating systems have this facility.

USB devices attach to the USB through ports on hubs that incorporate status indicators to indicate the attachment or removal of a USB device. The host queries the hub to retrieve these indicators. In the case of an attachment, the host enables the port and addresses of the USB device through the device's control pipe at the default address.

The host assigns a unique USB address to the device and then determines if the newly attached USB device is a *hub* or a *function*. The host then establishes its end of the control pipe for the USB device using the assigned USB address and endpoint number zero.

If the attached USB device is a hub and USB devices are attached to its ports, then the above procedure is followed for each of the attached USB devices. Alternatively, if the attached USB device is a function, then attachment notifications will be handled by appropriate host software.

When a USB device has been removed from one of a hub's ports, the hub will disable the port and provide an indication of device removal to the host. The relevant USB system software must handle this indication. Note that if the removed USB device is a hub, the USB system software must handle the removal of the hub as well as any USB devices that were previously attached to the system through the hub.

'Enumeration' is the name given to the allocation of unique addresses to devices attached to a USB bus. Because USB allows devices to attach or detach from the USB at any time, bus enumeration is an on-going activity for the USB system software. Additionally, bus enumeration includes the detection and processing of removals.

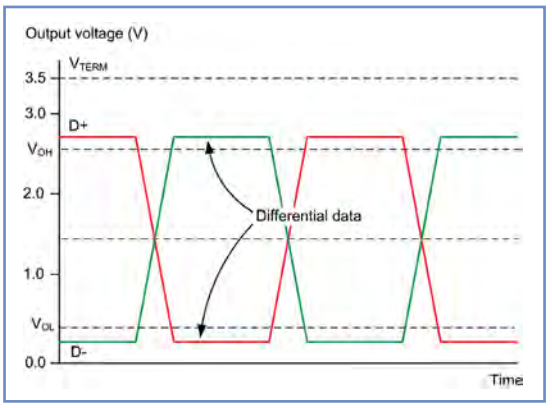

Fig.8.13. Signal levels present in a USB interface

Table 8.4 USB pin connections (see Fig.8.12) and conventional colours

Pin	Function	Colour
1	V_{BUS}	Red
2	D–	White
3	D+	Green
4	GND	Black

As mentioned earlier, USB employs two differential data lines (D+ and D–) and two power connections. CMOS buffers are normally used to drive the relatively low impedance of the USB cable and the signal voltage present on the D+ and D– must be kept within the ranges shown in Fig.8.13. Note that the terminating voltage (logic high) should be within the range 3.0 to 3.5V.

Detection of device connection is accomplished by means of pull-up and pull-down resistors placed respectively at the input/output of a port. USB pull-down resistors normally have a value of 15kΩ, while pull-up resistors have a value of 1.5kΩ. An interface adapter like that shown in Fig.8.14 can be extremely useful if you need to convert USB signal voltages to TTL-compatible signals (see Fig.8.14).

Loopback testing

The technique of loopback testing can be useful if you need to test a serial data interface; and is accomplished by looping back the transmitted data (TXD) back to the received data (RXD) line. Fig.8.15 shows the connections required to carry out a loopback test on an Arduino Uno microcontroller. When

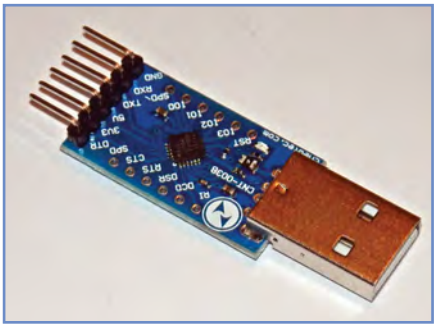

Fig.8.14. A low-cost TTL-to-USB interface

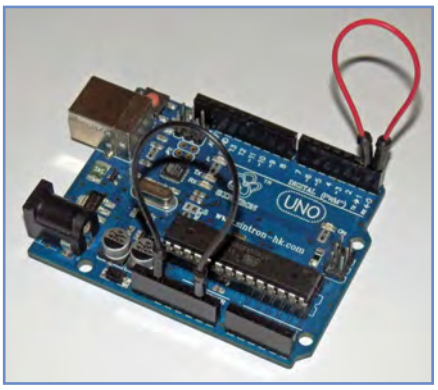

Fig.8.15. Carrying out a loopback test on an Arduino Uno microcontroller. The RESET line must be connected to GND (black link wire) and the TXD line to the RXD line (red link wire)

the links have been made the board is connected to a host computer via the USB interface and the serial monitor is then started, as shown in Fig.8.16. A string of ASCII text characters is first entered in the serial monitor before clicking on the Send button. The serial data then makes the round trip (by USB and RS-232) and is finally sent back to the host where it appears in the received data window. The data link can usefully be tested at different baud rates (in this case we have selected the fastest bit rate of 115200 baud).

Gearing up: Digital test equipment

Depending on the complexity of the circuit, digital test gear can be as basic as a hand-held logic probe and a few simple accessories. For more sophisticated logic, such as microcontrollers and microprocessors, a digital storage oscilloscope (DSO) is a useful acquisition (see Part 2). A DSO will allow you to capture a sample of data and then display it for detailed analysis at some later time. Data may be captured on a continuous basis or a trigger event selected in order to initiate data capture (note that it is possible to capture data both before and after a trigger event).

Fig.8.17 shows how an external USB 'scope can be used to monitor the signal lines on a Node MCU microcontroller. The resulting display (captured and stored for analysis) is shown in Fig.8.18). If you need to debug more complex microprocessor-based systems on a regular basis then a dedicated logic analyser can be a useful investment. Unfortunately, such instruments can be rather expensive, but they do become available from time to time both second-hand and from on-line auction sources. Fig.8.19 shows a vintage SA3 Logic Analyser that can capture 40 data channels at a rate of 10 million samples per second. Instruments like this can often be purchased for as little as £50.

If you are working on a strictly limited budget it is still possible to enjoy logic analysis by using a low-cost USB bus interface like that shown in Fig.8.20. This handy gadget provides you with eight TTL-compatible input channels and is designed for use in conjunction with computer-based data-capture software,

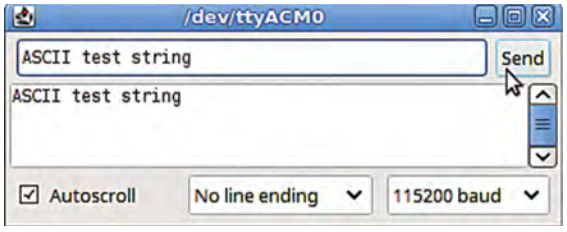

Fig.8.16. The serial monitor window showing the transmitted and received data

Fig.8.17. A USB digital storage 'scope with auxiliary digital inputs connected to digital I/O lines on a Node MCU microcontroller

Fig.8.18. Captured digital data using LabNation's SmartScope software

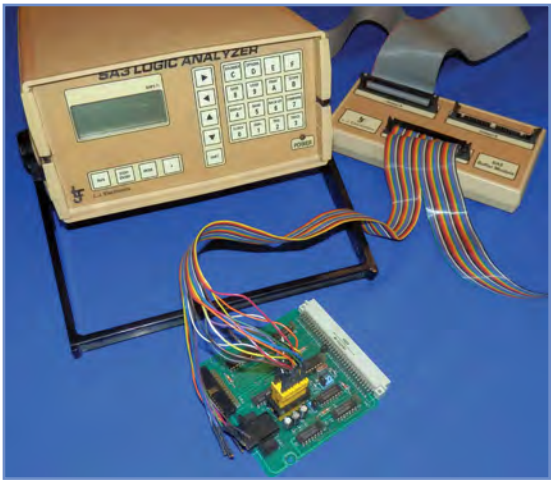

Fig.8.19. Vintage 40-channel SA3 Logic Analyser

Get it right when when carrying out digital measurements

- When using a logic probe or pulser, take care to avoid short-circuits on adjacent pins or tracks
- When using a logic probe or pulser, ensure that you have connected the supply leads correctly and that the supply voltage is the same as that used on the logic that you are testing
- Before making measurements on logic circuits it is always worth checking that the supply voltage(s) are within the expected range (a low, high or missing supply voltage can produce misleading results)
- Always observe anti-static procedures when working on logic devices and particularly when removing and replacing them from a circuit board
- When using an oscilloscope to observe logic signals, set the input to DC and always use a compensated ×10 probe to minimise loading on the circuit under investigation (see *Teach-In 2018: Part 2* for details)
- If a bus line indicates an indeterminate state (ie, when neither logic 1 nor logic 0 is indicated when using a logic probe) it may be useful to momentarily pull the line high or force it low, and note the changes produced.

Fig.8.20. Ultra-low-cost 8-channel USB logic analyser

such as sigrok PulseView (from: **https://sigrok.org/wiki/PulseView**).

In addition to equipment for capturing and analysing logic signals, a variety of accessories will help you make effective connection to the circuit or system under examination. Fig.8.23 shows a typical selection, including adapters, a line monitor, probes and IC test clips. Finally, Fig.8.24 and 8.25 respectively show how a breakout boards and IC test clips are used in typical measurement situations.

Fig.8.21. Connecting the low-cost logic analyser to a Node MCU microcontroller

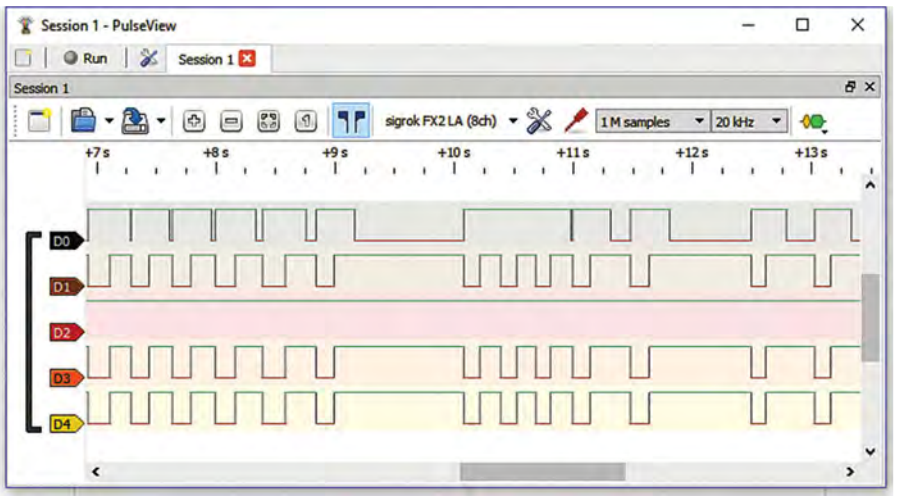

Fig.8.22. Using PulseView to capture and analyse data from a Node MCU microcontroller

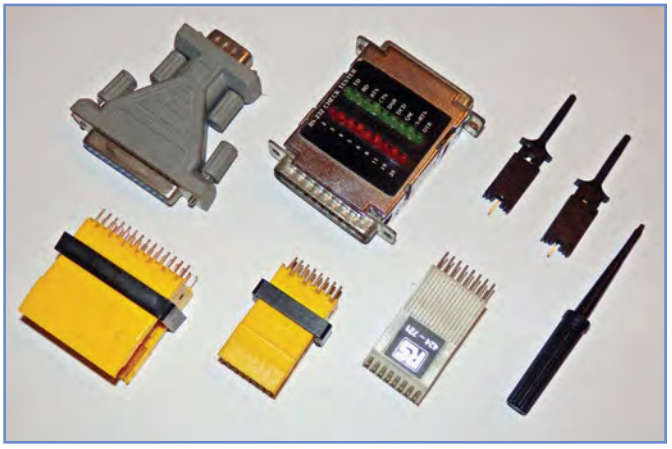

Fig.8.23. A variety of digital test gear accessories, including a 9-way to 25-way serial adapter, RS-232 line monitor, and various probes and IC test clips

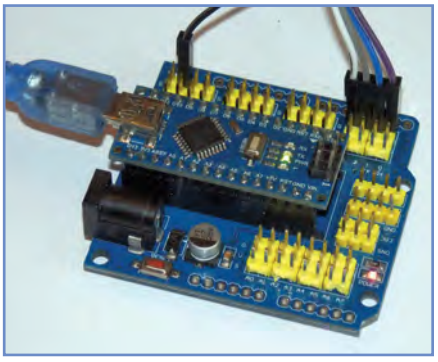

Fig.8.24. Using a breakout board to check logic signals present on an Arduino Nano

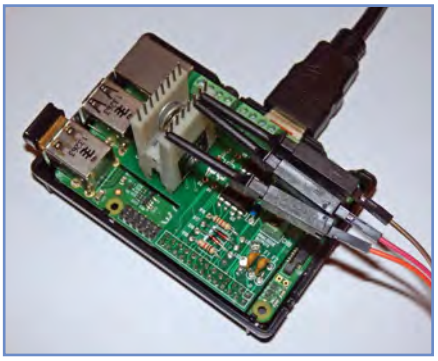

Fig.8.25. Using an IC test clip to check logic signals present on a Raspberry Pi expansion board

Test Gear Project: A simple logic probe

Our simple logic probe will provide you with a handy device for observing digital signals. Despite the lack of a pulse-stretching facility (see earlier) it is still possible to differentiate between static and clocked logic signals and make a rough assessment of duty cycle and mark-to-space ratio. This can be useful when it is necessary to determine whether a logic line has become stuck

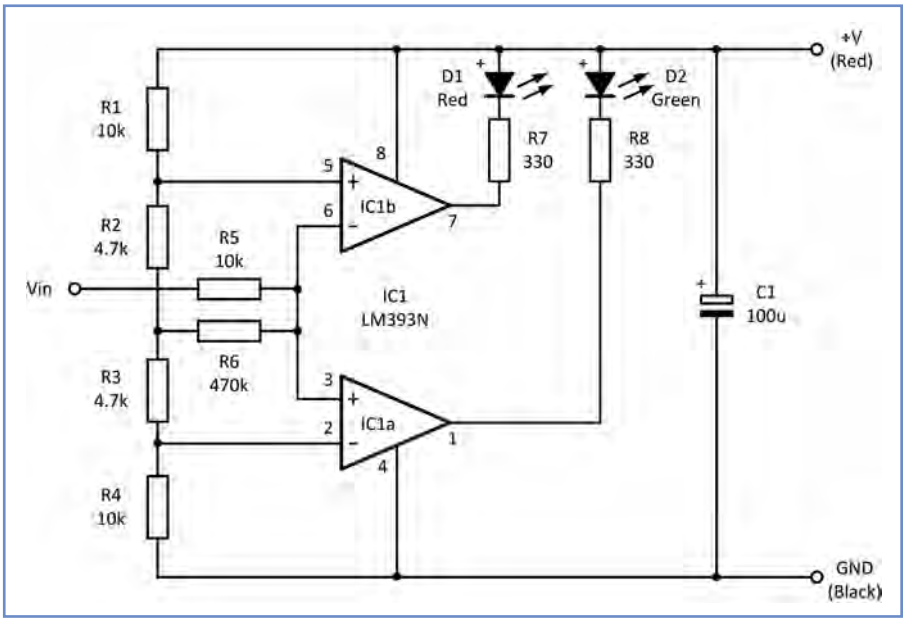

Fig.8.26. Complete circuit of the simple logic probe

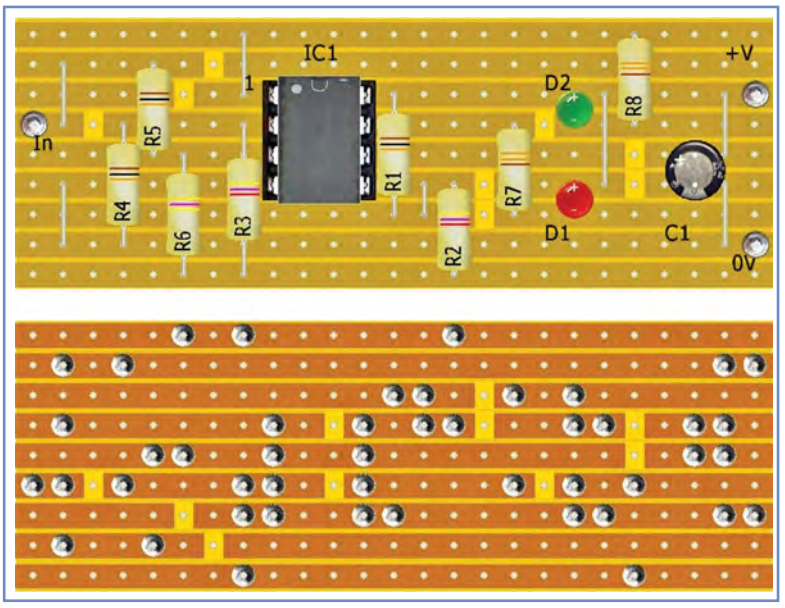

Fig.8.27. Stripboard layout of the simple logic probe

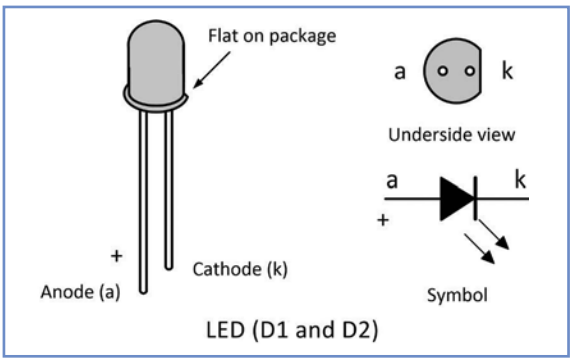

Fig.8.28. LED pin connections

Fig.8.29. Internal wiring of the simple logic probe

Fig.8.30. External appearance of the simple logic probe

or static (ie, permanently held at one or other logic level).

The complete circuit of our *Test Gear Project* is shown in Fig.8.26. The circuit comprises an LM393 dual comparator (IC1) and two LED indicators (D1 and D2) that indicate the state of the probe tip. If neither indicator is illuminated the probe is indicating a floating, indeterminate or tri-state condition.

You will need

Perforated copper stripboard (9 strips, each with 25 holes)
ABS logic probe case
Short length of twin insulated cable (see text)
2 insulated crocodile clips (black / red)
3 0.040-inch terminal pins
1 Miniature DPDT toggle switch (S1)
1 LM393N 8-pin DIL dual comparator
1 5mm red LED (D1)
1 5mm green LED (D2)
3 10kΩ resistors (R1, R4 and R5)
2 4.7kΩ resistors (R2 and R3)
1 470kΩ resistor (R6)
2 330Ω resistors (R7 and R8)
1 100µF 16V radial electrolytic (C1)

Assembly is straightforward and should follow the component layout shown in Fig.8.27. Note that the '+' symbol shown on D1 indicates the more positive (anode) terminal of the LED. The pin connections for the LED are shown in Fig.8.28. The reverse side of the board (NOT an X-ray view) is also shown in Fig.8.27. Note that there's a total of ten track breaks to be made. These can be made either with a purpose-designed spot-face cutter or using a small drill bit of appropriate size. There are also seven links that can be made with tinned copper wire of a suitable diameter or gauge (eg, 0.6mm/24SWG).

When soldering has been completed it is very important to carry out a careful visual check of the board as well as an examination of the track side of the board looking, for solder splashes and unwanted links between tracks. The internal and rear-panel wiring of the test signal source is shown in Fig.8.29. Finally, the PCB should be placed in the logic probe case (it should fit snugly inside the case with two holes drilled in the removable panel for D1 and D2). The probe enclosure used for the prototype was a Teko LP1 Probe Case, measuring 145×30×21mm and available from Rapid Electronics (Order code 31-0335).

The probe tip should be connected to the input terminal pin via a short length of insulated wire. The supply is connected using a twin insulated lead (400mm is ideal) terminated with red and black crocodile clips and soldered to the +V and 0V pins on the circuit board (see Fig.8.29).

Testing

Before use, it is important to test the logic probe, ensuring that the logic levels are correctly identified. Connect the supply leads to a 5V DC power source as shown in Fig.31. The probe tip is taken to the slider of a 1kΩ potentiometer that can then be adjusted to produce an input voltage of between 0V and +5V. The voltage at the probe tip is indicated using a digital multimeter and the voltage range for logic 0 (green LED illuminated) and logic 1 (red LED illuminated) can then be observed.

If the logic probe is working correctly the ranges should be 0V to 1.7V for logic 0 (green) and 3.3V to 5V for logic 1 (red). Note that neither LED should be illuminated for input voltages between about 1.7V and 3.3V. If this is not the case, check the orientation of IC1, the

Table 8.5 Threshold voltage levels for low and high logic states

Supply voltage	3V	3.3V	5V	9V	12V	15V
Low threshold	1.0V	1.1V	1.7V	3.1V	4.1V	5.1V
High threshold	2.0V	2.2V	3.3V	5.9V	7.9V	9.9V

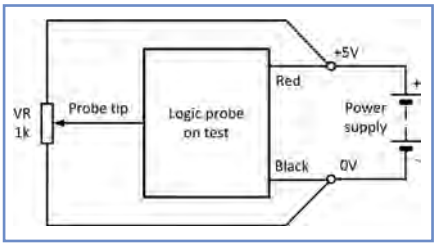

Fig.8.31. Test circuit for the simple logic probe

polarity of D1 and D2, and the circuit board wiring (checking that the track breaks and links have all been made correctly). Table 8.5 shows the typical threshold voltage levels when the logic probe is used with different supply voltages. Note that, under no circumstances should the supply voltage be allowed to exceed 15V.

Next month

In next month's *Teach-In 2018* we will bring the series to a close with some advice on designing and building your own test gear. We will also include an index to all previous parts of this series.

Fig.8.32. Typical supply lead connections. On this Nano breakout board the red crocodile clip is taken to '3V3' and the black is taken to 'GND'