

COMP2021 COMP9022

Digital System
Structures

Digital Laboratory Notes

Session 1, 1999

School of Electrical Engineering
School of Computer Science and Engineering
The University of New South Wales

Contents

I Experiments	9
1 Experiment One	13
2 Experiment Two	21
3 Experiment Three	27
4 Experiment Four	33
II Manual	39
5 Safety and Handling	43
6 Introducing the Lab	45
7 Setting up the Lab	49
8 Wiring Lists	57
9 Connecting Components	61
10 Testing Components	65
11 Logic and Circuit Diagrams	71
12 Trouble-Shooting	77
13 Data Sheets	85

Using this Manual

The Manual is not designed to be simply read from beginning to end. The recommended way to use this Manual is to *refer* to particular sections when needed. For example, you may be asked to refer to a specific Manual section during your work on a particular part of an **Experiment**. Alternatively, you may find that something is not working as expected, so referring to the *Testing* or *Troubleshooting* sections may give you some hints.

However, it is recommended that you read Section 5: *Safety and Handling*, before attempting any work with the Digital Logic Lab.

Safety and Handling

This section describes:

- how to use the Digital Logic Lab safely;
- how to handle the Digital Logic Lab components without damaging them.

Your Safety

The Digital Logic Lab is powered by low-voltage batteries, and there is no danger from **electrical shock** even if you touch exposed metal parts when the equipment is powered up.

However, there is some danger of skin punctures or cuts if you do not handle the components with care. The **integrated circuit (IC)** chips, for example, intentionally have sharp connecting **pins**.

Some of the materials used in making **printed circuit boards (PCBs)** are poisonous, and there may be traces of these materials present. This also applies to the connecting wires. So, to avoid ingesting poisonous material:

- do not use your teeth to strip the insulation from connecting wire—you should use a tool such as a **wire-stripper** (see Section 7);
- do not lick your fingers when handling the equipment;
- wash your hands after using the equipment.

Handling Components

Many of the components are small, and are easily damaged physically. For example, if IC **pins** are bent, it is difficult to insert all the pins together in their intended sockets—and the extra force needed may bend them further. Repeated bending and straightening of a pin will eventually cause it to break off, so it is important to keep IC pins straight.

Secondly, some of the **ICs** can easily be damaged electrically. There are two main problems:

- **static electricity**;
- battery power applied to the wrong pins.

There may be a **static electricity** problem if you have to work in an area with low humidity or synthetic carpeting, for example. It is good practice to discharge any build-up of static electricity on yourself before beginning to handle the equipment. (You can usually do this by touching the external metal surface of an **earthed** electrical appliance, such as a refrigerator or computer.

You can avoid applying battery power to the wrong pins of an IC by carefully checking the IC connections you have made *before* you insert the batteries to power up your system. It is good practice to remove at least one of the batteries *before* you make any changes to the wiring, and again after you have finished experimenting. (This will also prolong battery life).

Introducing the Lab

This section describes:

- the types of **components** used in the Digital Logic Lab;
- how Digital Logic Lab components can be connected to form **digital systems**.

Types of Components

Many of the Digital Logic Lab components are already connected together to form the **Digital Logic Lab Board** (see Figure 6.1).

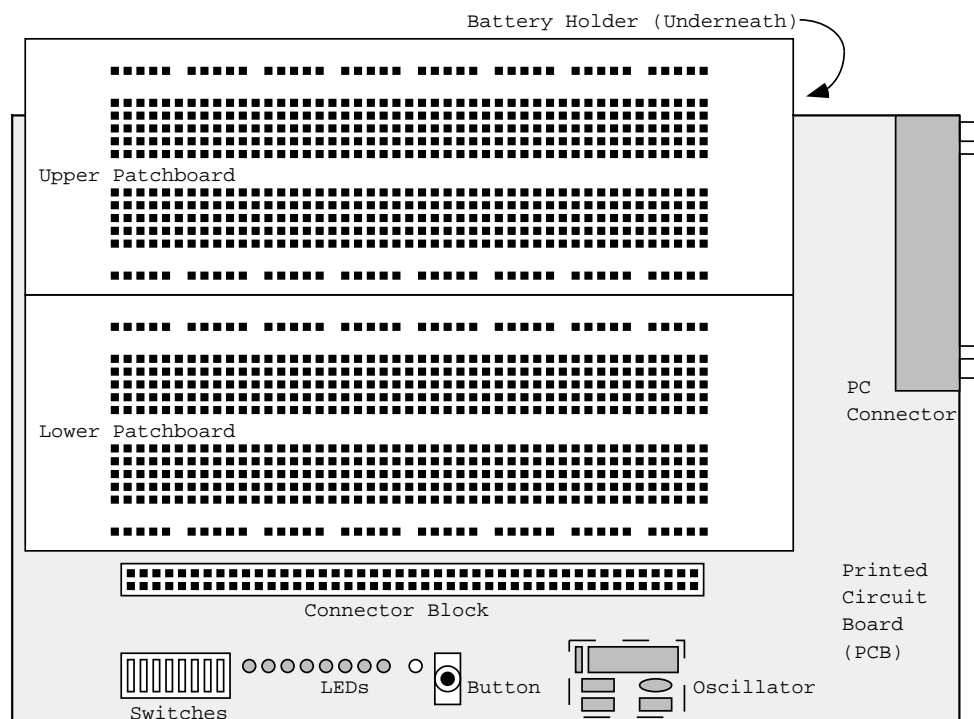


Figure 6.1: Digital Logic Lab Board

The main types of component on the Digital Logic Lab Board are labelled in Figure 6.1, and are briefly described as:

- **PCB:** a green fibreglass card with components mounted on it;
- **Patchboard:** a blue plastic block with several rows of holes;
- **Connector Block:** a black plastic block with two rows of holes;
- **Switches:** a red plastic block of eight small **rocker switches**;
- **LEDs:** a row of nine small **light-emitting diodes**, eight red and one green;
- **Button:** a spring-loaded push-button;
- **Oscillator:** a group of interconnected **electronic components** which generates an **oscillating electrical signal**;
- **Battery Holder:** a plastic moulding on the back of the board to hold the four **batteries** which power the board;
- **PC Connector:** a 25-way socket for connecting to a **PC** (unused in COMP2021 Experiments).

The PCB has four **integrated circuit** components (or **ICs**, or **chips**) mounted on it. These are identified as the black plastic blocks with a line of metal connectors (or **pins**) on each side (hence the name **dual-in-line**, or **DIL**, for this common method of **packaging ICs**).

In addition to the four ICs on the PCB, there is a **diode** (small, mostly-black cylinder), some **resistors** (either multi-coloured striped cylinders or black **single-in-line** packages), and some **capacitors**.

The remaining separate components of the Digital Logic Lab are:

- **ICs:** additional **integrated circuit** components in a black plastic bag;
- **Connecting Wire:** a coil of thin plastic-covered single-strand copper wire.

There is more information about the Digital Logic Lab components and how they work together in later Sections.

Connecting Components

The Digital Logic Lab components on the **PCB** are already partially connected together by copper **tracks** on the board. These connections are permanent and are described in detail in later Sections.

Additional connections between components (to construct interesting digital systems) are made by:

- inserting additional ICs into the **Patchboard**;
- making additional connections using sections of **Connecting Wire**.

Section 9 gives details on how to insert ICs and use the Connecting Wire properly, but the basic idea is that connections can be made:

- between pins of ICs inserted into the Patchboard;
- between pins of ICs and other components, using the **Connector Block**.

For example, battery power is supplied to an IC by connecting a wire from a hole labelled +5V on the Connector Block to a Patchboard hole next to the appropriate pin on the IC. The **Switches**, **LEDs**, **Button**, and **Oscillator** can also be connected to IC pins by using connecting wires between other (labelled) holes on the Connector Block and the appropriate Patchboard holes.

Section 7 gives details of how the Patchboard and Connector Block holes are identified.

Setting up the Lab

This section describes:

- how to identify the Digital Logic Lab components and connections;
- the full set of Digital Logic Lab components.

Component and Connection Identification

It is necessary to be able to *identify* each Digital Logic Lab component. It is also necessary to be able to identify the places on a component where *connections* can be made. For ICs in **DIL** packages, for example, the connection points are called **pins**. For a **Patchboard** or the **Connector Block**, the connection points are actually **holes** (although it is common to use the term “pin” for *all* types of connection).

Section 6 describes these Digital Logic Lab components sufficiently:

- **PCB**;
- **Battery Holder**;
- **PC Connector**;
- **Connecting Wire**.

This section explains more about the remaining components and their connections:

- **Patchboard**;
- **Connector Block**;
- **Power Supply**;
- **Switches**;
- **LEDs**;
- **Button**;
- **Oscillator**;
- **ICs**.

Patchboard

There are two identical Patchboards mounted on the PCB in the Digital Logic Lab. Each Patchboard has a total of 550 connecting **holes**, into which IC **pins** or one end of a section of **Connecting Wire** can be pushed. Inside the Patchboard, each hole connects to exactly four others, as shown in Figure 7.1).

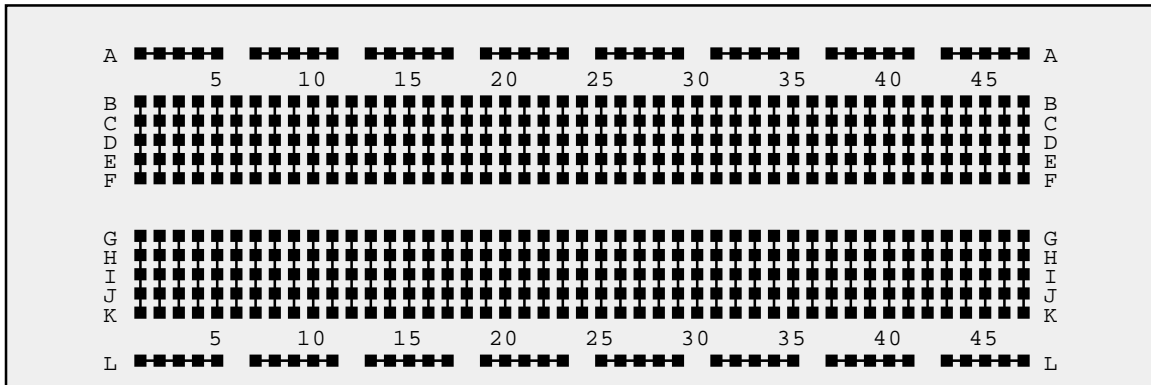


Figure 7.1: Patchboard Connections

Each Patchboard hole is uniquely identifiable, as the *rows* of holes are labelled A – L, and the *columns* are numbered 1 – 47. For example, the hole in the top left corner of a Patchboard is identified as the A1 hole. You can also see that there is no A6 hole.

The connections between each group of five holes (see Figure 7.1) can be described in terms of hole identifiers. For example, A1, A2, A3, A4, and A5 form one interconnected group, while B1, C1, D1, E1, and F1 form another group.

There are two Patchboards (“Upper” and “Lower” as shown in Figure 6.1), so Patchboard holes in the Digital Logic Lab can be uniquely identified by adding a “U” or “L” to the identifier. So the Patchboard hole in the top left corner of the Digital Logic Lab is UA1, while the bottom right Patchboard hole is LL47.

Connector Block

The Connector Block mounted on the PCB in the Digital Logic Lab (see Figure 6.1) has a total of 90 connecting **holes** in two rows. The pair of holes in each column are connected together by tracks on the PCB, as shown in Figure 7.2. The main purpose of the Connector Block is to enable connections to be made to the **peripheral components**, such as the Switches and LEDs.

Although there is no labelling of these holes shown on the board itself, it is convenient to use a similar identification scheme as for the Patchboard, with *rows* of holes A to B, and *columns* 1 to 45. The prefix “C” refers to the “Connector Block”, giving hole identifiers CA1 to CA45, and CB1 to CB45 (see Figure 7.2).

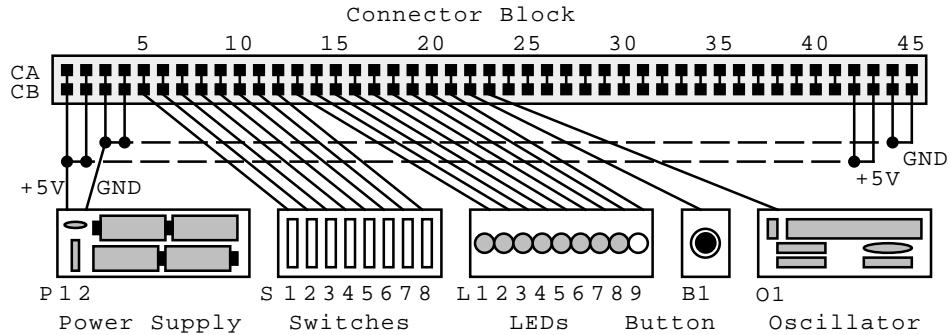


Figure 7.2: Connector Block

Power Supply and Peripherals

The **Power Supply** for the Digital Logic Lab is described in detail in Section 11. It is connected to components mounted on the PCB via **tracks**, and is also connected to Connector Block holes (so that **Connecting Wires** can be used to connect power to ICs inserted in a **Patchboard**). The Power Supply has two connections, identified as P1 and P2, which are available via the 16 Connector Block holes shown in Figure 7.2. Note that it is sometimes preferable to use an identifier which describes the *function* of a connection, rather than its *location* on the board. If so, the functional identifiers +5V and GND can be used instead of P1 and P2.

The term **peripheral component** is often used to refer to components such as **Switches**, **LEDs**, and **Buttons**, which are part of the **interface** between the digital system and the outside world. The “outside world” includes the human operator of the digital system, who can use Switches and Buttons to supply information to the digital system in a form it can use, and who can observe the LEDs to receive information from the digital system in a form that the human being can recognise. (The **Oscillator** can also be considered a peripheral component, although not part of the *human* interface).

For each of these peripheral components, which are described in more detail in Section 11, there are connections via printed-circuit tracks between them and the Connector Block, as shown in Figure 7.2. The same identification scheme can be

used as previously, so that the eight Switches have one connection each: S1 to S8. There are nine LEDs: L1 to L9, and the Button and Oscillator each have one connection: B1 and O1, respectively.

Table 7.1 summarises the Connector Block connections in the Digital Logic Lab, made via tracks on its PCB. (Note that the Connector Block pins C24 to C41 are not required for the experimental work in COMP2021, and should not be used).

Integrated Circuits

The Digital Logic Lab has some **integrated circuits** (or **ICs**) mounted on the PCB, and there are additional ICs in a black plastic bag. These ICs perform various **digital logic** functions, but they all look much the same—so there is an “identification code” printed on the IC package. Figure 7.3 shows the code (SN74LS00N) for a typical IC (similar to those in the Digital Logic Lab). The code has five parts:

- the **manufacturer** code is “SN”;
- the **digital logic family** code is “74”;
- the **IC technology** code is “LS”;
- the digital logic **function** code is “00”;
- the **packaging** code is “N”.

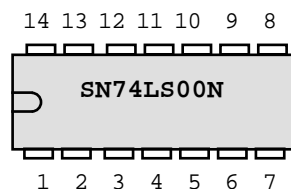


Figure 7.3: Typical IC Package

Figure 7.3 also shows 14 connecting **pins** on the IC package. There is a consistent way of numbering pins on these **dual-in-line** (or **DIL**) packages: looking down on the printed side of the chip, start enumerating pins *anticlockwise* from the “notch” on the package. There is an additional convention: the “power” pin has the highest pin number (pin number 14 for this chip), and the “ground” pin is at the opposite corner of the package (pin number 7).

Pin	=	Pin	=	Pin	Function
CA1	=	CB1	=	P1	Power (+5V)
CA2	=	CB2	=	P1	Power (+5V)
CA3	=	CB3	=	P2	Power (GND)
CA4	=	CB4	=	P2	Power (GND)
CA5	=	CB5	=	S1	Switch 1
CA6	=	CB6	=	S2	Switch 2
CA7	=	CB7	=	S3	Switch 3
CA8	=	CB8	=	S4	Switch 4
CA9	=	CB9	=	S5	Switch 5
CA10	=	CB10	=	S6	Switch 6
CA11	=	CB11	=	S7	Switch 7
CA12	=	CB12	=	S8	Switch 8
CA13	=	CB13	=	L1	LED 1
CA14	=	CB14	=	L2	LED 2
CA15	=	CB15	=	L3	LED 3
CA16	=	CB16	=	L4	LED 4
CA17	=	CB17	=	L5	LED 5
CA18	=	CB18	=	L6	LED 6
CA19	=	CB19	=	L7	LED 7
CA20	=	CB20	=	L8	LED 8
CA21	=	CB21	=	L9	LED 9
CA22	=	CB22	=	B1	Button 1
CA23	=	CB23	=	O1	Oscillator 1
CA24	=	CB24			(unused)
....	
CA41	=	CB41			(unused)
CA42	=	CB42	=	P1	Power (+5V)
CA43	=	CB43	=	P1	Power (+5V)
CA44	=	CB44	=	P2	Power (GND)
CA45	=	CB45	=	P2	Power (GND)

Table 7.1: Connector Block Usage

Digital Logic Lab Component List

Components Supplied

Most of the equipment required to perform experiments with the Digital Logic Lab is already supplied. Table 7.2 gives the full list of these supplied components. Use the identification guidelines (above) to check that there is a complete set.

Quantity	Item Name	Description
1	Board	Digital Logic Lab Board
1	Wire	Coil of Connecting Wire
2	74LS00	14-pin DIL IC
1	74LS08	14-pin DIL IC
1	74LS32	14-pin DIL IC
1	74LS42	16-pin DIL IC
2	74LS86	14-pin DIL IC
3	74LS112	16-pin DIL IC
2	74LS157	16-pin DIL IC
2	74LS192	16-pin DIL IC
2	74LS193	16-pin DIL IC
3	74LS194	16-pin DIL IC
2	74LS283	16-pin DIL IC

Table 7.2: Digital Logic Lab Components (Supplied)

Batteries

The Digital Logic Lab Board is powered by a set of 4 batteries, inserted in the **Battery Holder** underneath the PCB. (Note that batteries are *not* supplied with the Digital Logic Lab).

The batteries should be **Size C**, and high-quality **Alkaline** types are recommended. Rechargeable NiCad batteries should *not* be used, as their voltage rating (1.25V) is too low. One set of batteries is sufficient to perform the COMP2021 experiments for most students (assuming sensible usage such as disconnecting when not actually in use).

Wire-stripper

It is necessary to make a number of **Connecting Wires** from the coil of wire supplied. This involves cutting lengths of wire and stripping the insulation from each end (see Section 9). Although it is *possible* to use a tool such as a pocket knife, it is *recommended* that a purpose-designed **wire-stripper** tool be acquired. This will reduce the likelihood of “nicking” the wire while removing the insulation (which may cause it to break later).

Multimeter

A **multimeter** is an instrument for measuring electrical quantities such as **voltages**, and cheap multimeters are widely available for home and educational use.

A multimeter is *not* required during normal use of the Digital Logic Lab. However, if malfunction of the Digital Logic Lab is suspected, or the **batteries** need testing (see Section 10), access to a multimeter might be useful.

Wiring Lists

This section describes:

- how to produce a **wiring list** as an essential step in the construction of digital systems;
- the wiring list for the **Connector Block**.

A **wiring list** is an essential step in the construction of digital systems for these reasons:

- even simple **digital systems** may have many components and many interconnections;
- non-functioning **electronic components** do not look different from functioning components;
- it is very hard to spot a **wiring error** in a wired-up system;
- fixing wiring errors will often require complete re-wiring.

The purpose of the wiring list is to minimise the possibility of wiring errors, particularly in the *construction* of digital systems.

Producing a Wiring List

The **structural design** of a digital system is a set of decisions about which components to use, and how they should interconnect. A **circuit diagram** is a good way of summarising structural design (see Section 11). The connections are described in terms of their **function** (e.g. “system clock connects to clock input of J-K flip-flop”), which helps the designer check that the design will perform as intended.

The **wiring list** is usually produced *after* the circuit diagram of the system is finalised. It is essentially a translation of the structural design into a **physical design**, where connections are described in terms of **pin numbers** instead of pin *function* (e.g. “pin 4 of IC 6 connects to pin 13 of IC 8”). This type of description is preferable for guiding the *construction* of the system, as it focusses on the locations of connections without unnecessary information about what they are used for.

Both a structural description (e.g. circuit diagram) and a physical description (e.g. wiring list) are needed for successful design, construction, and testing of digital systems.

A wiring list is generally expressed as a tabular list like that of Table 8.1.

Component	Pin	=	Component	Pin
IC1	1	=	IC4	2
IC1	2	=	IC7	3
...
IC4	2	=	IC7	4
IC4	4	=	IC4	8
...

Table 8.1: Wiring List Structure

There are two columns which identify **components** of the system, and two columns which identify connecting **pins** of those components. Each entry (i.e. row) describes exactly *one* connection between a particular pin of one component and another pin of another (or the same) component. For example, Table 8.1 describes a connection between pins 4 and 8 of the component identified as IC4. Note that where several pins are mutually interconnected, this should be expressed as individual point-to-point connections in the wiring list (e.g. the three pins IC1 pin 1, IC4 pin 2, and IC7 pin 4 are connected together, and this is achieved by the *two* connections given in the table—a connection IC1 pin 1 to IC7 pin 4 should only be included if a separate connection is actually required).

Using the wiring list to construct (or check) the connections in a digital system is quite straightforward, because each entry (i.e. row) of the table specifies exactly one connection between two physical locations. For each entry:

- use the component and pin identification given in the wiring list entry to locate the two end-points of the connection;
- make (or check) the connection between the two points;
- mark the wiring list entry as “done” (or “checked”).

Wiring List for the Connector Block

The identification scheme used in Section 7 for the connection **holes** in the Digital Logic Lab is effectively the same as that used in the general **wiring list** Table 8.1.

The “Components” are the Upper Patchboard, Lower Patchboard, Connector Block, Switches, etc, each of which has its unique identifier (U, L, C, S, etc, respectively). The “Pins” are identified either by a single number, or by a row-and-column code (e.g. G47). The full wiring list for the Digital Logic Lab has thousands of entries, but that part of it which involves connections to the **Connector Block** is shown in Table 8.2.

Component	Pin	=	Component	Pin
C	A1	=	C	B1
C	A2	=	C	B2
C	A3	=	C	B3
C	A4	=	C	B4
C	A5	=	C	B5
C	A6	=	C	B6
C	A7	=	C	B7
C	A8	=	C	B8
C	A9	=	C	B9
C	A10	=	C	B10
C	A11	=	C	B11
C	A12	=	C	B12
C	A13	=	C	B13
C	A14	=	C	B14
C	A15	=	C	B15
C	A16	=	C	B16
C	A17	=	C	B17
C	A18	=	C	B18
C	A19	=	C	B19
C	A20	=	C	B20
C	A21	=	C	B21
C	A22	=	C	B22
C	A23	=	C	B23
C	A42	=	C	B42
C	A43	=	C	B43
C	A44	=	C	B44
C	A45	=	C	B45

Component	Pin	=	Component	Pin
C	B1	=	P	1
C	B2	=	P	1
C	B3	=	P	2
C	B4	=	P	2
C	B5	=	S	1
C	B6	=	S	2
C	B7	=	S	3
C	B8	=	S	4
C	B9	=	S	5
C	B10	=	S	6
C	B11	=	S	7
C	B12	=	S	8
C	B13	=	L	1
C	B14	=	L	2
C	B15	=	L	3
C	B16	=	L	4
C	B17	=	L	5
C	B18	=	L	6
C	B19	=	L	7
C	B20	=	L	8
C	B21	=	L	9
C	B22	=	B	1
C	B23	=	O	1
C	B42	=	P	1
C	B43	=	P	1
C	B44	=	P	2
C	B45	=	P	2

Table 8.2: Wiring List for Connector Block

Connecting Components

This section describes:

- how to insert **ICs**;
- how to prepare **Connecting Wires**;
- how to use the Connecting Wires to make electrical connections between components.

To use the Digital Logic Lab to construct a **digital system**, it is necessary to:

- insert a number of **ICs** in the **Patchboard** area;
- then make electrical connections (using **Connecting Wires** between **Patchboard** and/or **Connector Block** holes.

Inserting and Removing ICs

Section 5 contains guidelines on how to handle the ICs safely. It is also important to *design* the digital system, before you begin to *construct* it (see Section 11 and Section 8).

The recommended procedure for inserting a DIL IC is:

- ensure that all IC **pins** are straight and parallel with each other;
- use **Patchboard** row **F** for one line of pins on the **DIL** package, and Patchboard row **G** for the other line of pins;
- place the IC lightly on the Patchboard and confirm that *every* pin sits loosely within its intended hole;
- press the IC down vertically and firmly until the body of the IC rests on the Patchboard.

If you are inserting more than one IC:

- to facilitate removal later, leave a few unused Patchboard holes between one IC and the next;
- to minimise wiring errors, insert ICs so that they are all oriented the same way (which means that pin 1 of all devices will be in the same Patchboard row).

The recommended procedure for removing a DIL IC is:

- remove any **Connecting Wires** first;
- find two “levering tools” (e.g. small screwdrivers, ballpoint pens or straightened paper-clips) which can slide under the body of the IC in the groove between Patchboard rows F and G;
- lever both ends of the IC simultaneously, so that the IC lifts vertically;
- put the IC back in its storage bag immediately to avoid damage.

Preparing Connecting Wires

It is necessary to make a number of **Connecting Wires** from the coil of wire supplied (use *only* this wire—it has exactly the right dimensions to fit the holes in the Patchboard and Connector Block). This involves cutting lengths of wire and stripping the insulation from each end. In Section 7, it was recommended that a purpose-designed **wire-stripper** tool be used for this purpose.

Figure 9.1 shows a typical **Connecting Wire**. The bare ends need to be at least 5mm long to reach the metal contact area at the bottom of the Patchboard holes, but bare ends longer than 10mm will risk unintended metal contact with neighbouring wires. Unnecessarily long Connecting Wires may make wiring difficult to check, so it is recommended that various lengths be used (e.g. 5cm, 10cm, and 15cm).



Figure 9.1: Connecting Wire

It is recommended that Connecting Wires are kept straight when not in use. (A wire can break if repeatedly bent or twisted, and it is useful to be able to estimate its length quickly).

Using Connecting Wires

For all but the simplest digital systems, it is strongly recommended that a **wiring list** be used (see Section 8). Assuming that a wiring list is being used, which has one “entry” per connection, this is the recommended procedure for making the connection between two holes:

- use the hole identification information given in the wiring list entry to locate the two holes on the Digital Logic Lab Board;
- choose (or make) a Connecting Wire that is at least 25% longer than the distance between the holes;
- bend the wire in a smooth curve so that the two ends are parallel and about the right distance apart (sharp bends in the wire might break it);
- slide one end vertically into its hole until the insulation on the wire prevents further insertion;
- repeat with the other end;
- check again with the wiring list entry and, if correct, mark the entry as “done”.

When removing Connecting Wires, remove *one* end of *one* wire at a time by pulling it vertically from its hole. This will avoid damage to the hole, and careful removal and straightening of the Connecting Wires will enable them to be re-used many times.

Testing Components

This section describes:

- how to insert and remove **batteries**;
- the **self-test** procedure.

It is possible to test the main **electronic components** of the Digital Logic Lab without the need for specialised **electronic testing** equipment such as a **multimeter** or an **oscilloscope**. The **self-test** procedure requires a sequence of simple tests to be performed, each of which tests a particular component, and which builds on the results of previous (successful) tests. If any of the individual tests appear to *fail*, refer to Section 12.

Installing Batteries

Section 7 describes the type of **batteries** required for the Digital Logic Lab. The **Battery Holder** underneath the PCB holds the four batteries which should be inserted according to the diagram visible within the Battery Holder itself.

It is recommended that at least one battery is removed from the Battery Holder whenever the Digital Logic Lab is not being used, and usually when changes are being made to the **Connecting Wires**. (It is quite possible to destroy an IC by applying battery power to its non-power pins, even momentarily). An exception to this rule is the sequence of tests in the **self-test** procedure (below), where it is not necessary to remove the batteries between each test, because there are no ICs on the Patchboard to be damaged.

The Self-Test Procedure

Refer to Section 7 for information about how to identify **holes** on the **Patchboard** and **Connector Block**. Refer to Section 9 for information about how to prepare **Connecting Wires**. Refer to Section 8 for information about using **wiring lists**. Once again, if any of the individual tests appear to *fail*, refer to Section 12.

Component	Pin	=	Component	Pin
C	B1	=	C	B13

Table 10.1: Testing the Power Supply

Testing the Power Supply

- remove any **batteries** from the **Battery Holder**;
- remove any **Connecting Wires**;
- remove any **ICs** from the **Patchboard**;
- make the connection shown in Table 10.1 (i.e. use a **Connecting Wire** to make a connection between hole CB1 and hole CB13);
- install the four **batteries**;
- check the leftmost **LED (L1)**;
- if L1 is *on*, the **Power Supply** is *working*.

Note that this test cannot show if the batteries are sufficiently charged to power **ICs** adequately—check with *Further Testing* below.

Testing the Button

Component	Pin	=	Component	Pin
C	B13	=	C	B22

Table 10.2: Testing the Button

- perform the **Power Supply** test (above);
- if successful, remove the **Connecting Wire**;
- make the connection shown in Table 10.2;
- check the leftmost **LED (L1)**;

- if L1 is *on*, the **Button** is *not working*;
- press and hold down the **Button**;
- if L1 stays *off*, the **Button** is *not working*;
- release the **Button**;
- if L1 goes *off*, the **Button** is *working*.

Testing the Oscillator

Comp- onent	Pin	=	Comp- onent	Pin
C	B13	=	C	B23

Table 10.3: Testing the Oscillator

- perform the **Power Supply** test (above);
- if successful, remove the **Connecting Wire**;
- make the connection shown in Table 10.3;
- check the leftmost **LED** (L1);
- if L1 stays *off*, the **Oscillator** is *not working*;
- if L1 stays *on*, the **Oscillator** is *not working*;
- if L1 repeatedly goes *on* then *off* in a regular **cycle**, taking less than one second per cycle, the **Oscillator** is *working*.

Testing the LEDs

- perform the **Power Supply** test (above);
- if successful, remove the **Connecting Wire**;
- make the connections shown in Table 10.4;
- check all nine **LEDs**: L1 to L9;
- if all **LEDs** are *on*, all **LEDs** are *working*.

Component	Pin	=	Component	Pin
C	B1	=	C	B13
C	A13	=	C	B14
C	A14	=	C	B15
C	A15	=	C	B16
C	A16	=	C	B17
C	A17	=	C	B18
C	A18	=	C	B19
C	A19	=	C	B20
C	A20	=	C	B21

Table 10.4: Testing the LEDs

Component	Pin	=	Component	Pin
C	A5	=	C	B13
C	A6	=	C	B14
C	A7	=	C	B15
C	A8	=	C	B16
C	A9	=	C	B17
C	A10	=	C	B18
C	A11	=	C	B19
C	A12	=	C	B20

Table 10.5: Testing the Switches

Testing the Switches

- perform the **Power Supply** test (above);
- if successful, remove the **Connecting Wire**;
- make the connections shown in Table 10.5;
- push down the *upper* end (marked 0) of all eight **Switches**: S1 to S8;
- check the eight **LEDs**: L1 to L8;
- if any **LED** is *on*, the corresponding **Switch** is *not working*;
- push down the *lower* end (marked 1) of all eight **Switches**: S1 to S8;
- check the eight **LEDs**: L1 to L8;
- if all **LEDs** are *on*, all **Switches** are *working*.

Further Tests

It was mentioned in *Testing the Power Supply* (above) that additional testing may be needed to confirm that the batteries are sufficiently charged to power the **ICs** adequately. If a standard **multimeter** is available:

- perform the **Power Supply** test (above);
- if successful, use the **multimeter** to measure the D.C. **voltage** between holes CA1 and CA3;
- if this voltage is at least +4.75, the **Power Supply** is *working* and able to power the **ICs** adequately.

If this test fails, it is necessary to obtain a new set of batteries (see Section 7).

If there is *any doubt* that the batteries are adequate, either because the **multimeter** test cannot be performed, or for other reasons, it is again a cost-effective solution simply to obtain a new set of batteries.

Logic and Circuit Diagrams

This section describes:

- the **logic diagram** as a useful way of relating the **structure** of a digital system to its **function**;
- the **circuit diagram** as a useful way of describing the complete **structure** of a digital system;
- circuit diagrams for Digital Logic Lab **subsystems**.

A **logic diagram** is an essential step in the design of digital systems for these reasons:

- even simple **digital systems** may have many **digital logic** components and many interconnections;
- it enables the designer to relate the **structure** of any part of the system to its intended function;
- being a **graphical** description, it is easier to comprehend than a **textual** description (i.e. “a picture is worth a thousand words”);

The main purpose of the logic diagram is to describe the *structure* of the digital system. It is called a “logic” diagram because it shows just the **digital logic** components and interconnections responsible for the **logic function** of the digital system.

Producing a **circuit diagram** is an essential *additional* step in the design of digital systems for these reasons:

- there are additional non-digital components and interconnections to be included in the digital system’s overall structure;
- an accurate **wiring list** can only be produced from a full circuit diagram;

The purpose of the circuit diagram is to describe *all* electrical components and connections in the *structure* of the digital system.

Producing a Logic Diagram

A **wiring list** (see Section 8) is used to guide the *construction* of a digital system, and so it intentionally focusses on the locations of connections without unnecessary information about what they are used for. So a wiring list is unsuitable as a way of documenting the **functional** aspects of the digital system's structure.

The **structural design** of a digital system is a set of decisions about which components to use, and how they should interconnect. These decisions are based on:

- knowledge of the required overall **behaviour** of the system, i.e. a specification of its **function**;
- knowledge of the **behaviour** of the available types of component, especially **digital logic** components, used in digital system design.

A **logic diagram** is a good way of summarising the functional aspects of the structural design of the digital system. The components are represented **graphically**, using agreed **symbols** to indicate their function, or (for more specialised or complex components), a “box” shape plus descriptive text. Figure 11.1 shows an example of each of these two styles. (Each of these two components is used in the Digital Logic Lab, and its function is described in a *Data Sheet*, Section 13):

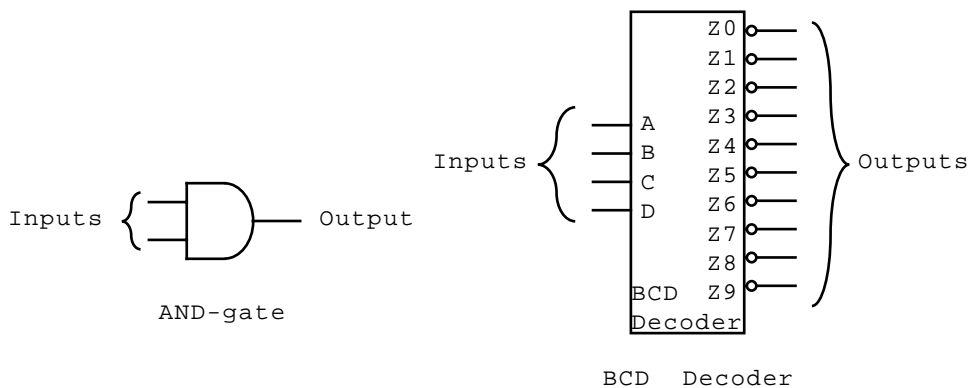


Figure 11.1: Digital Logic Components

Logic components have well-defined places where electrical **connections** can be made to other components. An **output** connection *supplies* information produced by the component (as an electrical **signal**), whereas an **input** connection *receives* information for use by the component. It is useful to attach suitably-descriptive **functional names** to some of the output signals (e.g. `door_closed`), as an aid to

understanding the behaviour of the system. Most connections shown in the logic diagram will simply connect an output of one component to an input of another component. Figure 11.1 shows the logic inputs and outputs of an **AND-gate** and a **BCD Decoder**.

Figure 11.2 is part of the logic diagram for a microwave oven controller. Its **function** is to allow cooking to occur only if the microwave door is closed, *and* the timer is set, *and* the start switch is activated. The **structural design** uses two **AND-gates**, and the important signals have been given **functional names**.

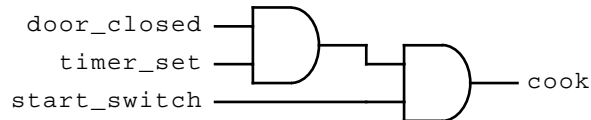


Figure 11.2: Example of a Logic Diagram

Producing a Circuit Diagram

A logic diagram can be expanded into a full circuit diagram by including details such as the electrical **power** connections, which do not contribute directly to the particular logic **function** of the system, but which are necessary aspects of its eventual physical construction. For example, the **AND-gate** in Figure 11.1 is likely to be available in a **DIL** package, along with three other **AND-gates**. In addition to the logic inputs and outputs of the **AND-gates**, the package has the two connections necessary to supply power to the four **AND-gates**. So a **circuit diagram** should include *additional* information which identifies connections (both logic and power connections) in terms of numbered **pins** on named packages, as well as any **functional name** for the signal.

Figure 11.3 is a circuit diagram which corresponds to the logic diagram of Figure 11.2. A 74LS08-type **DIL** package (see Section 13) is being used, and is given the identifying name **IC3** in the circuit diagram.

Circuit Diagrams for the Digital Logic Lab

Some digital systems are constructed entirely from **digital logic** components, but others have parts (or **subsystems**) which contain non-digital (or **analog**) electronic components such as **resistors** and **capacitors**, and even **electromechanical** components such as switches. It is useful to be able to include these components along with

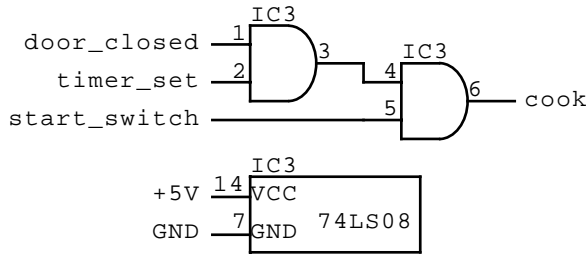


Figure 11.3: Example of a Circuit Diagram

the digital components in the system’s circuit diagram. The following subsections give the circuit diagrams for the electronic subsystems making up the Digital Logic Lab. (Section 7 describes the overall structure of the Digital Logic Lab).

Circuit Diagram for the Power Supply

Figure 11.4 is a circuit diagram for the **Power Supply** subsystem of the Digital Logic Lab Board. Four fresh C-type batteries in series develop at least +6.4V, which is reduced by the series diode to an output voltage at P1 within the required range: +4.75V to +5.25V

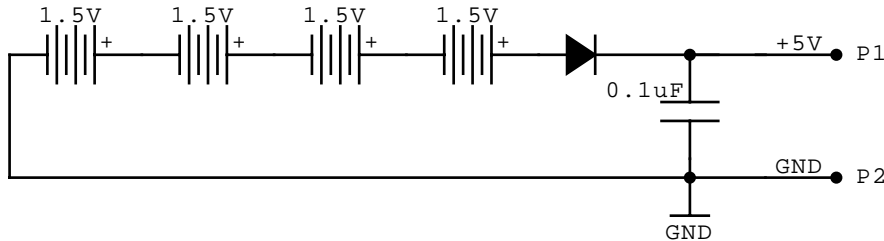


Figure 11.4: Power Supply Circuit Diagram

Circuit Diagram for the Switches

Figure 11.5 is a circuit diagram for the first of the eight switches in the **Switches** subsystem of the Digital Logic Lab Board. The rocker switch has two positions: in the “0” position, the output S1 is LOW (i.e. connected to GND); in the “1” position, the output S1 is HIGH (i.e. connected to +5V via a current-limiting 4.7K resistor). The other switches have the same circuit diagram.

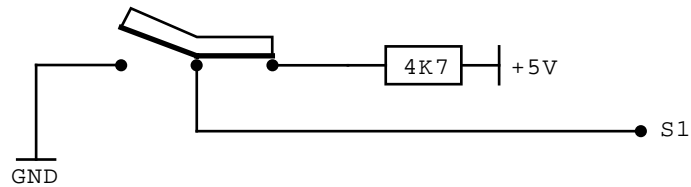


Figure 11.5: Switch Circuit Diagram

Circuit Diagram for the LEDs

Figure 11.6 is a circuit diagram for the first of the nine LEDs in the **LEDs** subsystem of the Digital Logic Lab Board. A HIGH voltage at L1 will enable current (limited by the 1.5K resistor) to flow through the LED causing illumination; a LOW at L1 will not enable illumination. The other LEDs have the same circuit diagram, except that L9 has a current-limiting 1.0K resistor to enable its green light to have similar brightness to the other red ones.

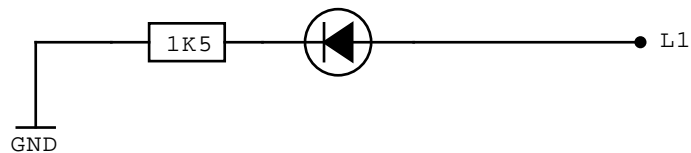


Figure 11.6: LED Circuit Diagram

Circuit Diagram for the Pushbutton

Figure 11.7 is a circuit diagram for the **Button** subsystem of the Digital Logic Lab Board. There is a debounce latch circuit constructed from cross-coupled CMOS Schmitt inverters, and a third inverter acts as a signal-conditioning amplifier. The spring-loaded pushbutton has two positions, with the upper “not pushed” position supplying a GND signal to one side of the latch, which in turn causes a LOW output at B1. As the button is pushed, it first breaks the upper connection, while the latch retains its previous state. Then the lower connection is made, forcing the latch to switch states, which sets the B1 output to HIGH. Any subsequent bouncing (i.e. making and breaking the lower connection) will not alter the latch state; only making the upper connection will do that.

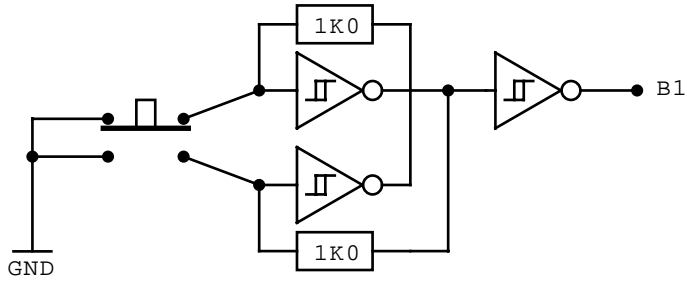


Figure 11.7: Button Circuit Diagram

Circuit Diagram for the Oscillator

Figure 11.8 is a circuit diagram for the **Oscillator** subsystem of the Digital Logic Lab Board. The basic oscillator comprises a CMOS Schmitt inverter with resistor/capacitor timing, and a second inverter acts as a signal-conditioning amplifier. Assuming the capacitor is at a LOW voltage, the inverter's output is HIGH, so the resistor will be charging the capacitor. As the capacitor voltage rises, it will cross the threshold at which the inverter switches to a LOW output, which will enable the resistor to begin discharging the capacitor. Different rising and falling thresholds (a property of Schmitt gates) ensures that there is a finite time (a substantial fraction of a second in this case) associated with both oscillator states.

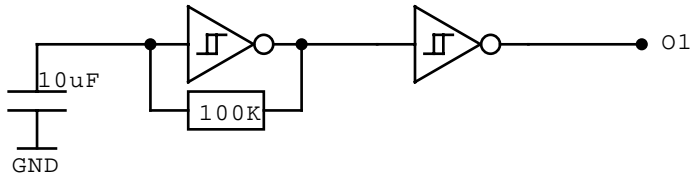


Figure 11.8: Oscillator Circuit Diagram

Trouble-Shooting

This section describes:

- general ideas for **debugging** systems constructed using the Digital Logic Lab;
- some common **faults** in systems constructed using the Digital Logic Lab.

What is “debugging”?

If a system has one or more “**bugs**” in it, the system mostly functions as expected, but not in all circumstances. So “**debugging**” is the process of:

- **testing** the system to uncover the existence of any bugs;
- **diagnosing** the cause of each bug;
- **rectifying** the system to remove each bug.

Simple systems, if designed correctly, should not have bugs. But as systems become more complex, such as many digital systems are nowadays, the risk of bugs occurring during the system’s design and construction increases. So designers of digital systems have to know how to fix bugs in (i.e. “debug”) their designs.

The most important idea in **debugging** is to do as little of it as possible! This does not mean that, once you have built your system, you don’t need to test it, but rather that you should be as thorough and careful as possible in the *design* and *construction* of the system in the first place, which will minimise the occurrence of bugs needing to be fixed. The reason for this approach is simple—it is always much less time-consuming to *avoid* bugs by careful design than it is to *diagnose* and *rectify* them.

Bugs are revealed by unexpected system behaviour, either during normal use of the system, or during systematic **testing** of it. Bugs may be due to faults in any or all of:

- the *logic* design of the system;
- the *circuit* design of the system;
- the *construction* of the system;

- the *components* of the system;

There is a useful distinction between a system with bugs in it, and a system which has *become* faulty after a period of working correctly. In the latter case, the most likely source of the new fault is component failure. But remember that electronic digital components are extremely reliable, and do not “wear out” during use. So it is quite likely that what *seems* to be a new fault in an otherwise working system is in fact a bug whose effects have been masked until now.

For commercially-produced digital systems, the increasing use of computer-aided design (**CAD**) and computer-aided manufacturing (**CAM**) has reduced the likelihood of circuit design and construction faults, leaving logic design as the main source of faults. When using manual design and construction, such as experimenting with the Digital Logic Lab, bugs are much more likely to be due to faults in design or construction, rather than in the components themselves. So most of this Section is about debugging, rather than what to do about faulty components!

There is both good news and bad news about the debugging of *digital* systems. The good news is that each test usually produces either a “right” or “wrong” result—with analog systems there might be “nearly right” results which could be difficult to interpret. The bad news is that most digital systems of interest are **complex**, both in behaviour and structure, so that many tests are needed before the system can be declared bug-free.

Types of testing

The simplest type of test is sometimes called a **go/no go test**. When the test is performed, the system either appears to “work” or does not. This test is most useful in debugged systems, where the testing is to check for the component-failure type of fault—if the test succeeds, further testing may not be necessary at that time, but a failure indicates the need for further **diagnostic testing**. For example, a number of go/no go tests are performed each time a PC is switched on—if the startup screen eventually appears, the PC is “working”; if not, there is likely to be a fault in the system somewhere.

When a previously-working system fails a go/no go test, there is an assumption that there is a component fault somewhere, so more detailed **diagnostic testing** is used to identify the location of the fault. For example, the PC could be started with an alternative boot disk, which, if now successful, would identify the original disk as a likely source of the problem. Alternatively, some optional components of the system could be disconnected (e.g. printers, scanners, network) before retesting. In principle, this approach can be continued at an ever-increasing level of detail, until

the fault is isolated to a component (e.g. disk drive, power supply, memory card) which the user is prepared to replace. If the system now passes the original go/no go test, further testing by the user of the presumed-faulty component is unnecessary (although the *manufacturer* of that component might be interested!)

This **top-down** approach to testing a system (i.e. a go/nogo test followed by increasingly-detailed diagnostic testing), is *not* appropriate when debugging a system. Passing a go/nogo test for the first time might make the designer feel good, but it doesn't give much information about the full range of behaviour a system is usually required to exhibit. Equally, failing a go/nogo test doesn't help to identify the *type* of bug (logic/circuit design, or construction/component fault).

For some very simple systems, it may be possible to perform **exhaustive testing** when debugging. This means devising a test for every possible behaviour of the system, and then performing all of them. Unfortunately, this is impractical for most systems of interest. For example, to exhaustively test a four-function calculator, you have to perform all the possible calculations it is capable of!

When debugging a system, the most effective approach is to start by individually testing the *components* (ideally exhaustively), then by testing interconnections of tested components (i.e. *subsystems*), and finally testing the whole system. This is the exact reverse of the **top-down** approach, and is therefore called the **bottom-up** approach. There are two main advantages with this approach. Firstly, debugging can occur in parallel with development of the system, enabling faults to be diagnosed (and hence rectified) early. Secondly, there is usually an exponential relationship between the **complexity** of a system (as measured by number of components, interconnections, inputs, etc) and the number of tests required to exhaustively test it. So testing two n -component subsystems is almost always much less expensive than testing a single $2n$ -component system.

Debugging example

Section 11 introduced a circuit for part of a microwave oven controller, shown in Figure 12.1. Suppose this system is to be constructed using the Digital Logic Lab, and then debugged.

This system is small enough to be exhaustively tested, in which case the circuit would first be constructed in full (by carefully following the **wiring list**), including the power supply connections. Exhaustive testing means supplying every possible combination of binary signals to the three inputs, and observing the value of the single binary output in each case. To perform the sequence of tests efficiently and systematically, a **test rig** (i.e. a convenient way of supplying binary inputs and

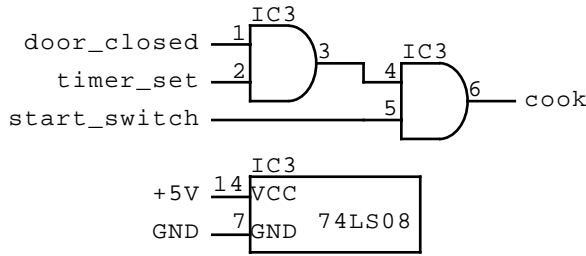


Figure 12.1: Example of a Circuit Diagram

observing binary outputs) is usually necessary. In this case, a suitable **test rig** has a separate **Switch** connected to each circuit input, and an **LED** connected to the circuit output. Another requirement is a **testing check list** document describing the **test rig** configuration (i.e. the connections to circuit inputs and outputs), and with each entry listing the input values for each test, the expected output values, and the results actually observed. For this example, exhaustive testing requires 8 tests, as shown in the **testing check list** shown in Table 12.1.

door_closed S1	timer_set S2	start_switch S3	cook L1	Result?
0	0	0	Off	
0	0	1	Off	
0	1	0	Off	
0	1	1	Off	
1	0	0	Off	
1	0	1	Off	
1	1	0	Off	
1	1	1	On	

Table 12.1: Exhaustive testing checklist

The preferred **bottom-up** approach to debugging this circuit involves testing the components (gates) individually, then their interconnection. With this approach, a suitable **test rig** involves an additional internal **test point**, namely Pin 3 of IC3, which enables the **testing check list** shown in Table 12.2 to be used.

The first group of 4 tests exhaustively test one of the gates. If these tests are successful, the second gate (whose output is also the system output) is tested by the second group of 4 tests. Note that, in this case (but not always), it is possible to individually test the gates without requiring them to be temporarily disconnected from the rest of the

door_closed S1	timer_set S2	IC3/Pin3 L1	start_switch S3	cook L2	Result?
0	0	Off	-	-	
0	1	Off	-	-	
1	0	Off	-	-	
1	1	On	-	-	
0	0	-	0	Off	
0	0	-	1	Off	
1	1	-	0	Off	
1	1	-	1	On	

Table 12.2: Bottom-up testing checklist

system.

In this case, both exhaustive testing and bottom-up testing require 8 tests. This is misleading, because of the simplicity of the system. For example, even a slightly more complex circuit, with two 3-input AND-gates (instead of 2-input gates) requires 32 tests for exhaustive testing but only 16 tests for bottom-up testing. As system complexity increases, this difference rapidly becomes enormous.

Common logic design faults

Faults that can occur at the **logic design** stage include:

- having an incomplete or contradictory specification of the required system function;
- having an incomplete or contradictory description of the function of a component to be used in the system;
- making faulty design decisions.

The first two types of fault will lead to unexpected system behaviour (i.e. **bugs**), but are difficult to avoid in many systems, for the same reason that **exhaustive testing** (see above) is often impractical—the level of **complexity** is too high.

Logic design includes deciding which components to use, and how to connect them together. So using an unsuitable component, or using it inappropriately, are examples of faulty design decisions.

Common circuit design faults

Faults that can occur at the **circuit design** stage include:

- incomplete or misunderstood pin usage for components;
- not meeting the electronic limitations of components or circuit connections.

The first type of fault includes such things as omitting connections to “unused” inputs on an IC—it is prudent to provide a connection to *all* input pins, even though each circuit connection has some cost. For example, the circuit in Figure 12.1 involves a 74LS08 chip, which contains two additional unused gates. Ideally, the inputs of these gates should be connected to GND, say.

The other group of faults are to do with the “analog” aspects of the system. For example, there are limits to how many logic inputs can be connected to a logic output for reliable behaviour, and there are guidelines for the reliable distribution of power in systems with many components.

Common construction faults

Faults that can occur at the **construction** stage include:

- a mismatch between the **circuit diagram** and **wiring list**;
- a mismatch between the **wiring list** and the actual wiring;
- poor physical connections of pins or wires;

In an ideal world, there should never be examples of the first two types of fault, but in reality, there are. For example, even with automated design techniques, it is possible for an obsolete version of a wiring list to be used during actual construction.

In digital systems, connections are usually made by metal-to-metal sliding contact (e.g. **IC pins** in sockets), by **welding**, or by **soldering**. There are long-established guidelines for ensuring that highly-reliable connections are made, but mistakes can occur, especially in large-scale systems with many thousands of connections.

Faulty components

It is very unlikely that component failure will occur in the Digital Logic Lab, but the following are possible:

- failure of a **Connecting Wire**;
- failure of one of the separate **ICs**;
- failure of one of the Digital Logic Lab subsystems;

If mishandled, **Connecting Wires** can break, occasionally without obviously appearing broken. If a **Connecting Wire** is suspect, proving it defective may be less efficient than simply substituting a new one!

ICs fail either through physical misuse (causing damaged pins), or through electrical misuse (e.g. connecting power to an output pin). In the latter case, simply replacing the IC is not sensible—it will also fail for the same reason! It will be necessary to diagnose the reason for IC failure before attempting substitution.

Section 10 describes a set of **diagnostic tests** for the various on-board subsystems of the Digital Logic Lab. If these tests prove that the Experiments cannot be completed because of component failure, the *Information Booklet* should be consulted, where the procedure for obtaining a replacement Digital Logic Lab is described.

Data Sheets

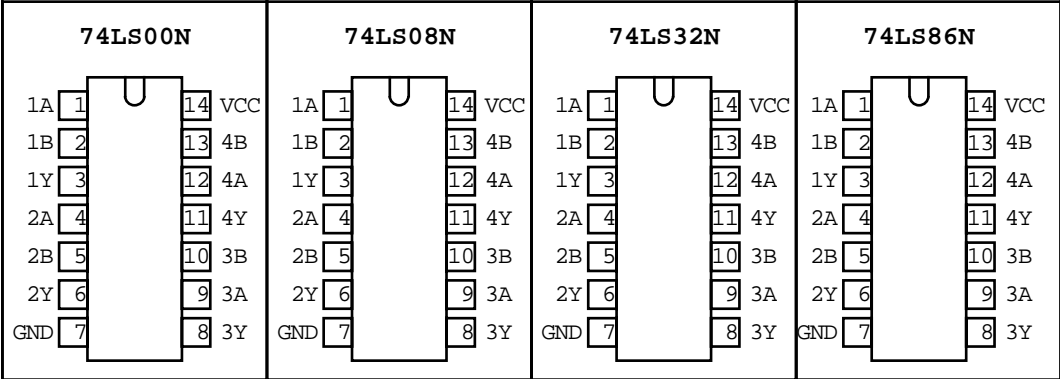
This section contains Data Sheets for the Digital Logic Lab IC components:

- 74LS00
- 74LS08
- 74LS32
- 74LS86
- 74LS42
- 74LS112
- 74LS157
- 74LS192
- 74LS193
- 74LS194A
- 74LS283

74LS00, 74LS08, 74LS32, 74LS86 QUAD 2-INPUT GATES

1

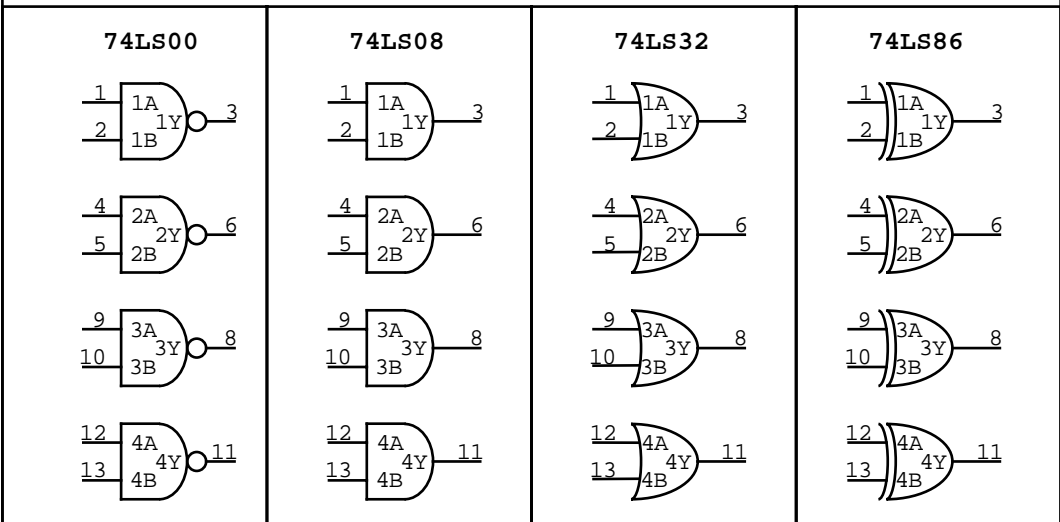
Pin Configurations (Top View)



Function Tables

74LS00 NAND			74LS08 AND			74LS32 OR			74LS86 EXCLUSIVE OR		
INPUTS A B	OUTPUT Y		INPUTS A B	OUTPUT Y		INPUTS A B	OUTPUT Y		INPUTS A B	OUTPUT Y	
L L	H		L L	L		L L	L		L L	L	
L H	H		L H	L		L H	H		L H	H	
H L	H		H L	L		H L	H		H L	H	
H H	L		H H	H		H H	H		H H	L	

Logic Diagrams



74LS00, 74LS08, 74LS32, 74LS86 QUAD 2-INPUT GATES
2
Maximum Ratings over Operating Free-Air Temperature Range

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		-0.5		7	V
VI Input voltage		-0.5		7	V
Operating free-air temperature range		0		70	°C
Storage temperature range		-65		150	°C

Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		4.75	5	5.25	V
IOH High-level output current				-400	μA
IOL Low-level output current				8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

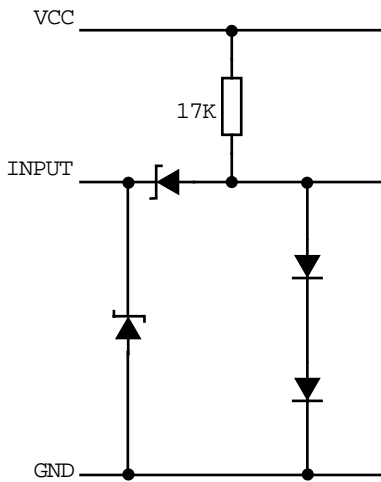
PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage			0.8		V
VIK Input clamp voltage	VCC=MIN, II=-18mA			-1.5	V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V			0.2	mA
IIH High-level input current	VCC=MAX, VI=2.7V		20	40	μA
IIL Low-level input current	VCC=MAX, VI=0.4V		-0.4	-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		5	10	mA

Switching Characteristics, VCC = 5V, TA = 25°C

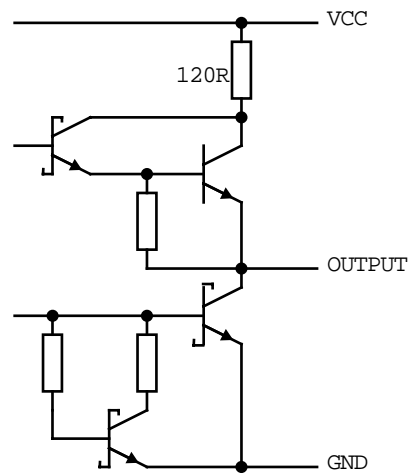
PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
TPLH Propagation delay, low-to-high output	Other input=L, CL/RL=15pF/2K	7	12	23	ns
TPHL Propagation delay, high-to-low output	Other input=L, CL/RL=15pF/2K	6	10	17	ns
TPLH Propagation delay, low-to-high output	Other input=H, CL/RL=15pF/2K	12	20	30	ns
TPHL Propagation delay, high-to-low output	Other input=H, CL/RL=15pF/2K	7	13	22	ns

Schematics of Inputs and Outputs

EQUIVALENT OF EACH INPUT



TYPICAL OF ALL OUTPUTS



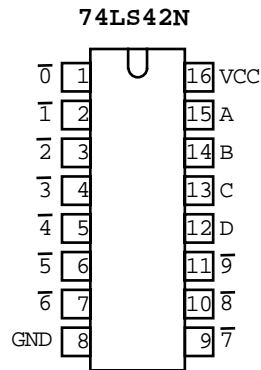
74LS42 4-LINE TO 10-LINE DECODER (BCD TO DECIMAL)

1

Description

This monolithic decimal decoder consists of eight inverters and ten four-input NAND gates. The inverters are connected in pairs to make BCD input data available for decoding by the NAND gates. Full decoding of valid input logic ensures that all outputs remain off for all invalid input conditions.

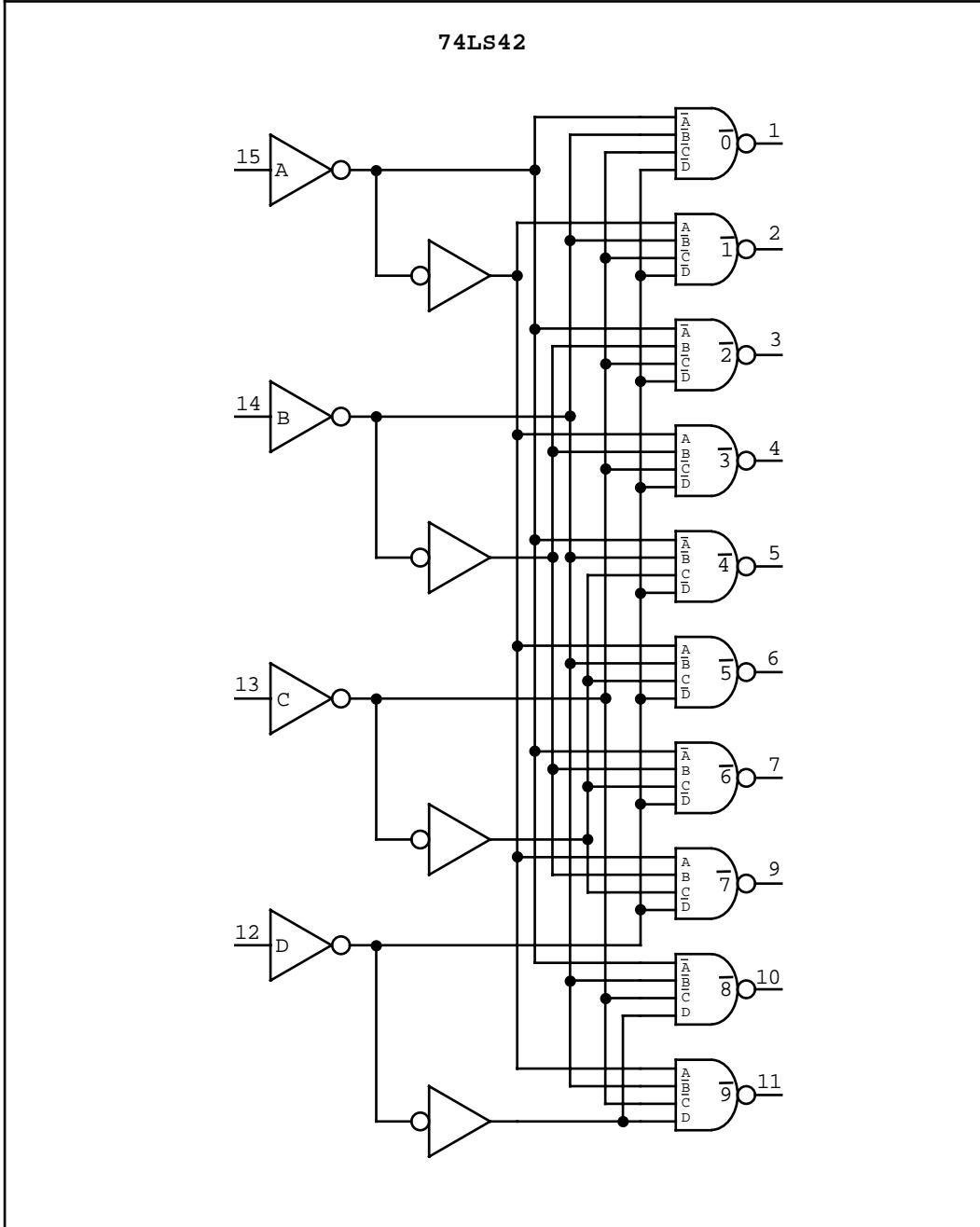
Pin Configuration (Top View)



Function Table

74LS42 BCD TO DECIMAL DECODER													
INPUTS				OUTPUTS									
D	C	B	A	0	1	2	3	4	5	6	7	8	9
L	L	L	L	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	H	H	L	H	H	H	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H	H	H	H
L	H	L	L	H	H	H	H	L	H	H	H	H	H
L	H	L	H	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
L	H	H	H	H	H	H	H	H	H	H	L	H	H
H	L	L	L	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
H	L	H	L	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
H	H	L	L	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	H	L	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

Logic Diagram



74LS42 4-LINE TO 10-LINE DECODER (BCD TO DECIMAL)**3****Maximum Ratings over Operating Free-Air Temperature Range**

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		-0.5		7	V
VI Input voltage		-0.5		7	V
Operating free-air temperature range		0		70	°C
Storage temperature range		-65		150	°C

Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		4.75	5	5.25	V
IOH High-level output current				-400	μA
IOL Low-level output current				8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage			0.8		V
VIK Input clamp voltage	VCC=MIN, II=-18mA			-1.5	V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V		0.2		mA
IIH High-level input current	VCC=MAX, VI=2.7V		20	40	μA
IIL Low-level input current	VCC=MAX, VI=0.4V		-0.4	-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		7	13	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
TPLH Propagation delay, low-to-high output	CL/RL=15pF/2K		15	30	ns
TPHL Propagation delay, high-to-low output	CL/RL=15pF/2K		15	30	ns

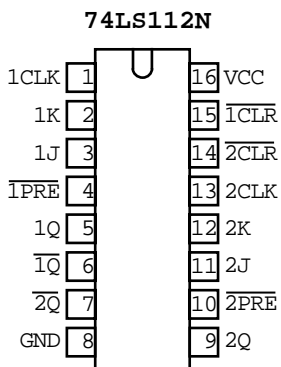
74LS112 DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR

1

Description

This device contains two independent J-K negative-edge-triggered flip-flops. A low level at the preset or clear inputs sets or resets the outputs regardless of the levels of the other inputs. When preset and clear are inactive (high), data at the J and K inputs meeting the setup time requirements are transferred to the outputs on the negative-going edge of the clock pulse. Clock triggering occurs at a voltage level and is not directly related to the rise time of the clock pulse. Following the hold time interval, data at the J and K inputs may be changed without affecting the levels at the outputs.

Pin Configuration (Top View)

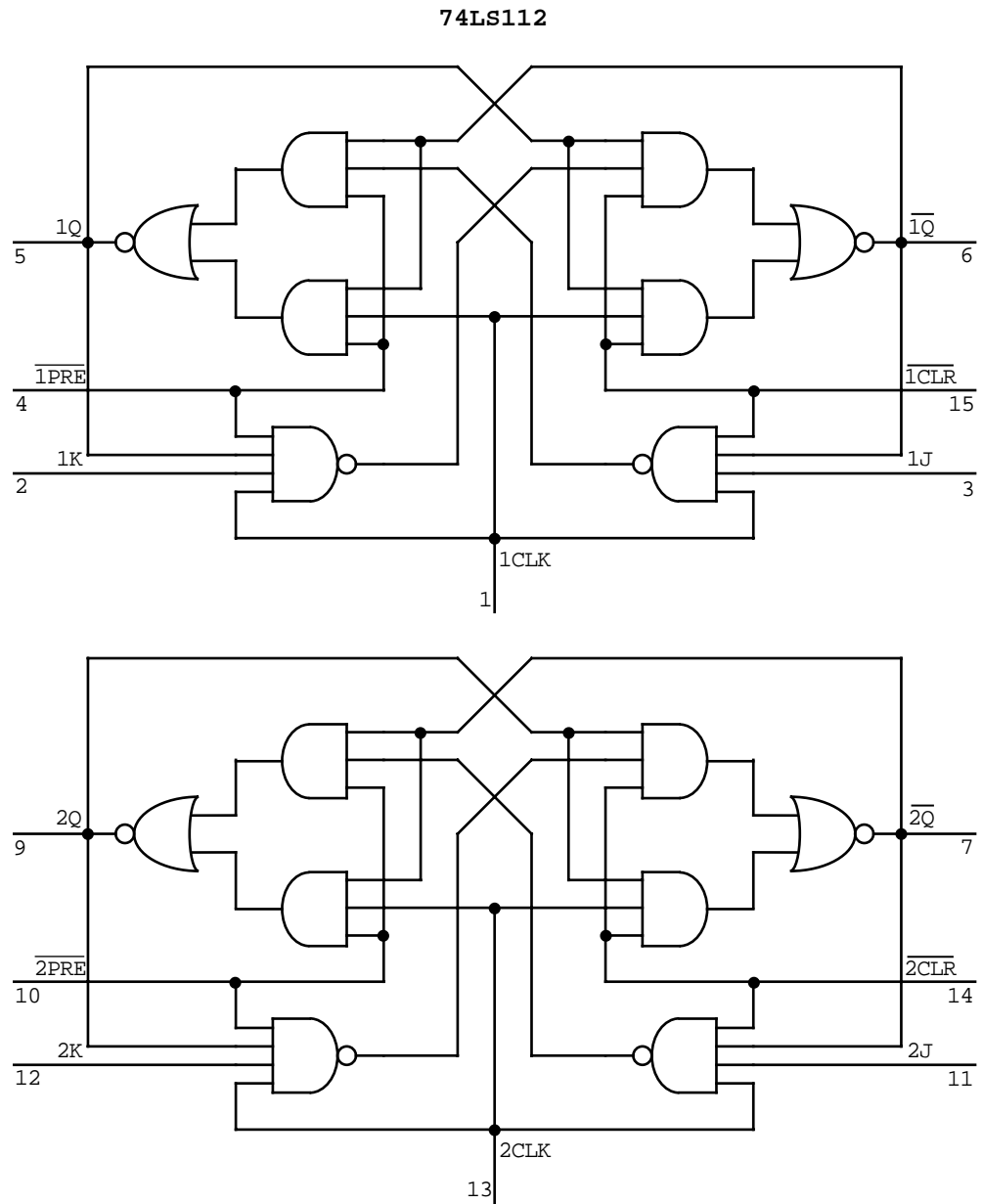


Function Table

74LS112 DUAL J-K FLIP-FLOPS

INPUTS					OUTPUTS	
$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	CLK	J	K	Q	$\overline{\text{Q}}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H (Temporary)
H	H	↓	L	L	Q	$\overline{\text{Q}}$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	$\overline{\text{Q}}$	Q
H	H	H	X	X	Q	$\overline{\text{Q}}$

Logic Diagram



74LS112 DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR

3

Maximum Ratings over Operating Free-Air Temperature Range

PARAMETER	MIN	NOM	MAX	UNIT
VCC Supply voltage			7	V
VI Input voltage			7	V
Operating free-air temperature range	0		70	°C
Storage temperature range	-65		150	°C

Recommended Operating Conditions

PARAMETER	MIN	NOM	MAX	UNIT
VCC Supply voltage	4.75	5	5.25	V
IOH High-level output current			-400	μA
IOL Low-level output current			8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage			0.8		V
VIK Input clamp voltage	VCC=MIN, II=-18mA			-1.5	V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V			0.4	mA
IIH High-level input current	VCC=MAX, VI=2.7V			80	μA
IIL Low-level input current	VCC=MAX, VI=0.4V			-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		4	6	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
Fclock Clock frequency	CL/RL=15pF/2K	0		30	MHz
TW Pulse duration	CL/RL=15pF/2K	20			ns
TSU Setup time for data before CLK ↓	CL/RL=15pF/2K	25			ns
TH Hold time for data after CLK ↓	CL/RL=15pF/2K	0			ns
TPLH Propagation delay, low-to-high output	CL/RL=15pF/2K		15	20	ns
TPHL Propagation delay, high-to-low output	CL/RL=15pF/2K		15	20	ns

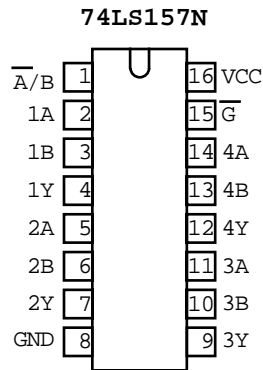
74LS157 QUAD 1-BIT DATA SELECTOR/MULTIPLEXERS

1

Description

This monolithic data selector/multiplexer contains inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input \overline{G} is provided. A 4-bit word is selected from the A or B source by $\overline{A/B}$ and is routed to the 4-bit output Y.

Pin Configuration (Top View)

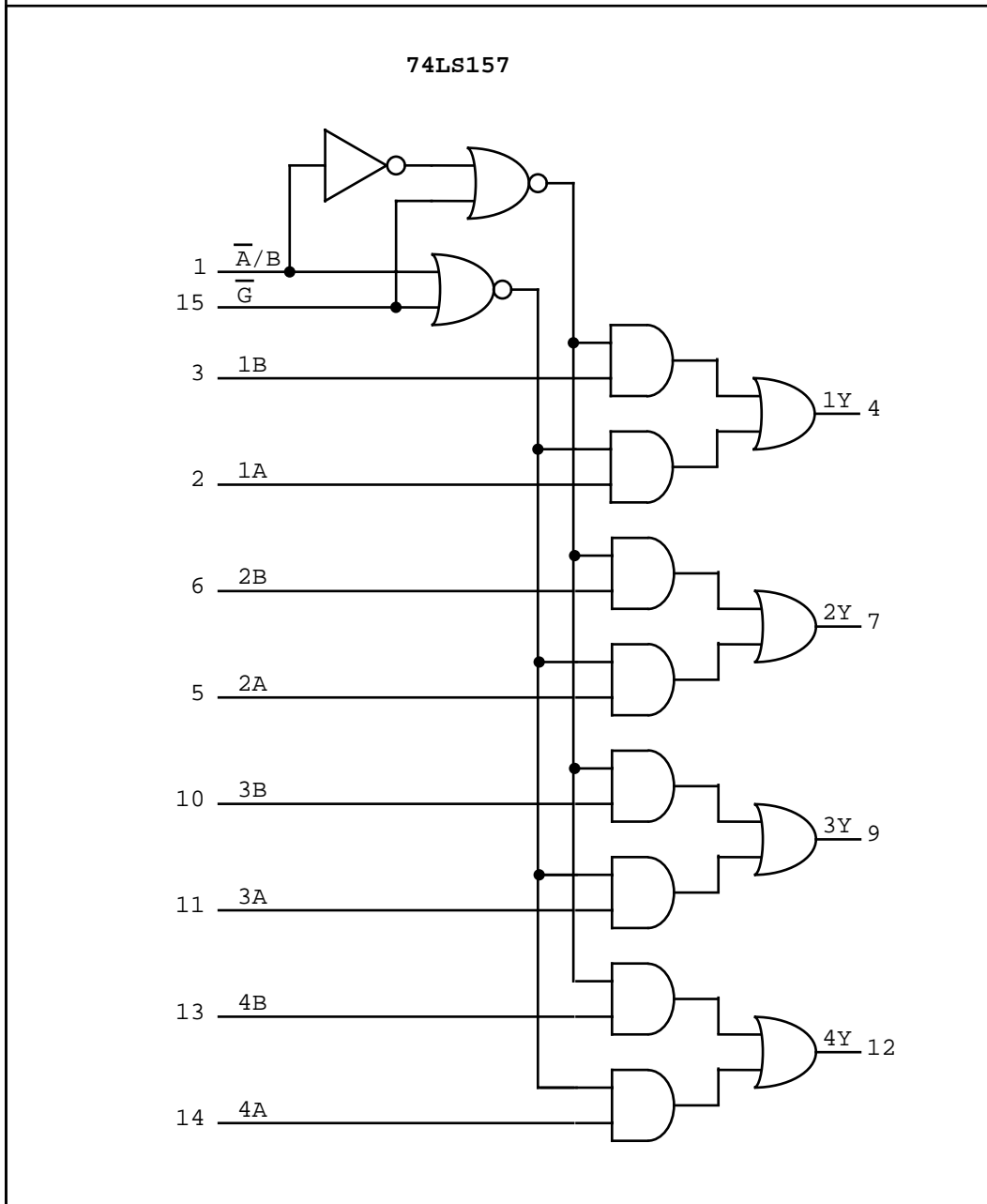


Function Table

74LS157 QUAD 1-BIT DATA SELECTOR/MULTIPLEXERS

	INPUTS				OUTPUTS
	\overline{G}	$\overline{A/B}$	A	B	Y
	L	L	L	L	L
	L	L	L	H	L
	L	L	H	L	H
	L	L	H	H	H
	L	H	L	L	L
	L	H	L	H	H
	L	H	H	L	L
	L	H	H	H	H
	H	L	L	L	L
	H	L	L	H	L
	H	L	H	L	L
	H	L	H	H	L
	H	H	L	L	L
	H	H	L	H	L
	H	H	H	L	L
	H	H	H	H	L

Logic Diagram



74LS157 QUAD 1-BIT DATA SELECTOR/MULTIPLEXERS

3

Maximum Ratings over Operating Free-Air Temperature Range

PARAMETER		MIN	NOM	MAX	UNIT
VCC	Supply voltage	-0.5		7	V
VI	Input voltage	-0.5		7	V
Operating free-air temperature range		0		70	°C
Storage temperature range		-65		150	°C

Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
VCC	Supply voltage	4.75	5	5.25	V
IOH	High-level output current			-400	μA
IOL	Low-level output current			8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH	High-level input voltage	2			V
VIL	Low-level input voltage			0.8	V
VIK	Input clamp voltage	VCC=MIN, II=-18mA		-1.5	V
VOH	High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX		2.7 3.4	V
VOL	Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35 0.5	V
II	Input current @ max. input voltage	VCC=MAX, VI=7V		0.2	mA
IIH	High-level input current	VCC=MAX, VI=2.7V		20 40	μA
IIL	Low-level input current	VCC=MAX, VI=0.4V		-0.4 -0.8	mA
IOS	Short-circuit output current	VCC=MAX		-20 -100	mA
ICC	Supply current	VCC=MAX (grounded inputs)		8 16	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
TPLH/TPHL	Propagation delay, data to output	CL/RL=15pF/2K		9 14	ns
TPLH/TPHL	Propagation delay, enable to output	CL/RL=15pF/2K		14 21	ns
TPLH/TPHL	Propagation delay, select to output	CL/RL=15pF/2K		16 25	ns

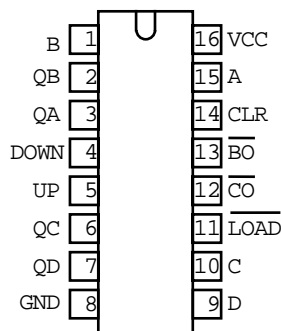
74LS192, 74LS193 4-BIT UP/DOWN COUNTERS**1****Description**

These monolithic circuits are synchronous reversible (up/down) counters. The 192 is a BCD counter (i.e. 0 to 9) and the 193 is a 4-bit binary counter. Synchronous operation is provided by having all master-slave flip-flops clocked simultaneously so that the outputs change together. A low-to-high transition on either of the count inputs (UP or DOWN), while the other input is high, changes the count value appropriately.

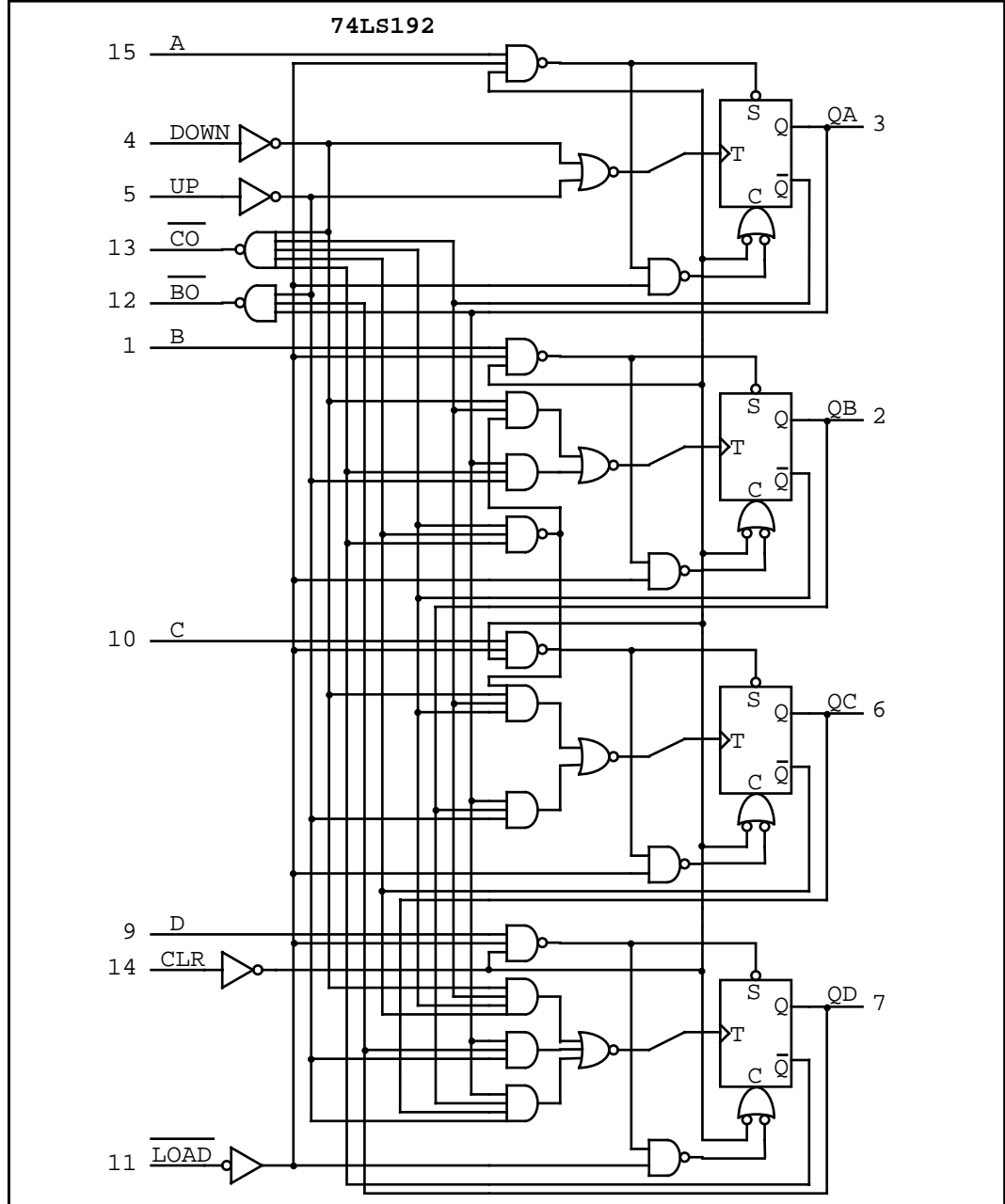
The counters may be preset by applying the desired 4-bit count value to the A - D inputs and asserting the LOAD input low. Presetting takes precedence over any count pulses.

The counters may be cleared (all outputs low) by asserting CLR high. Clearing takes precedence over counting and presetting. This enables the counters to be used as modulo-N dividers.

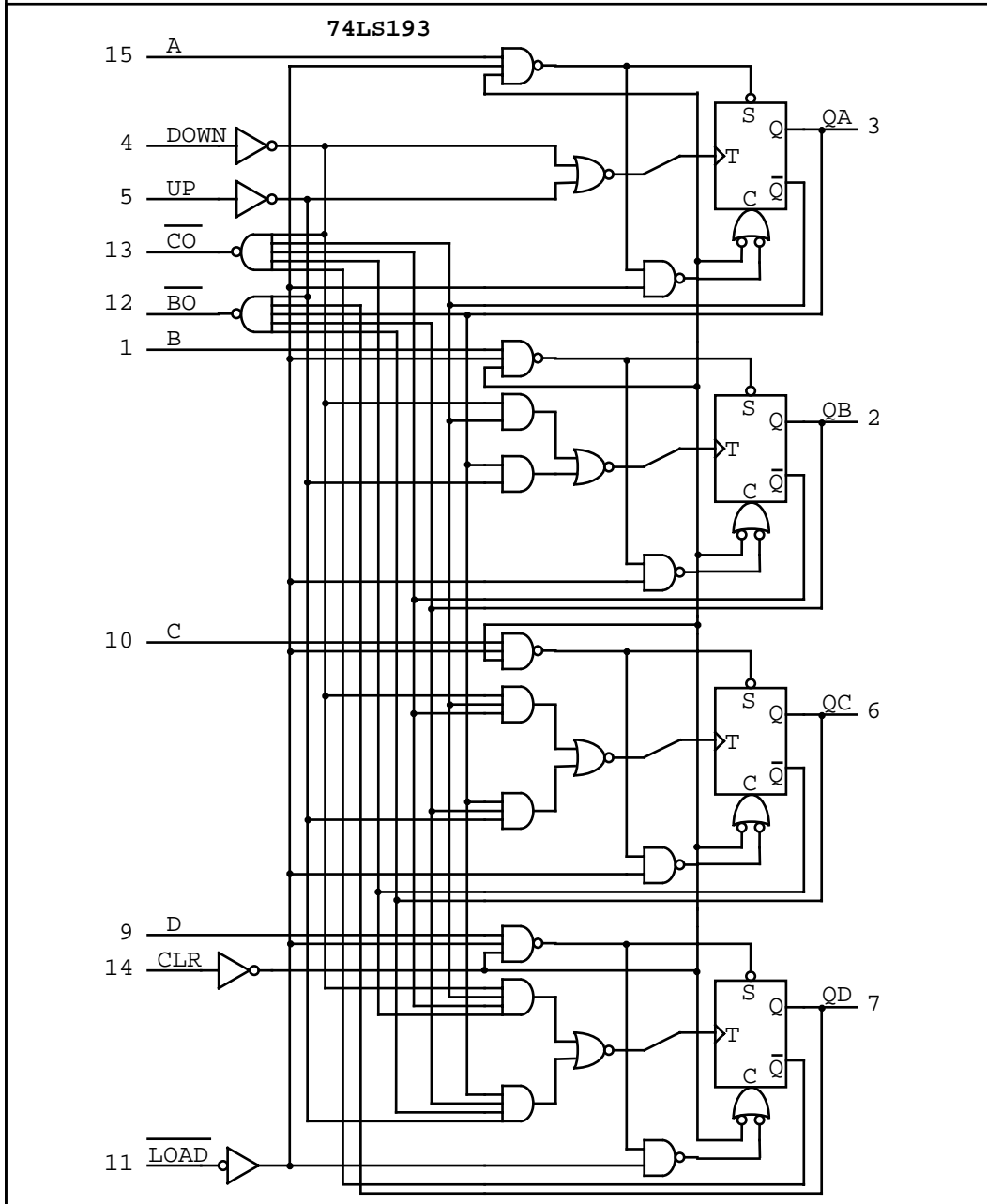
The counters may be cascaded by connecting the $\overline{\text{CO}}$ and $\overline{\text{BO}}$ outputs to the UP and DOWN inputs, respectively, of the succeeding counter.

Pin Configuration (Top View)**74LS192N/74LS193N**

Logic Diagrams



Logic Diagrams (Contd)



74LS192, 74LS193 4-BIT UP/DOWN COUNTERS

4

Maximum Ratings over Operating Free-Air Temperature Range

PARAMETER	MIN	NOM	MAX	UNIT
VCC Supply voltage	-0.5		7	V
VI Input voltage	-0.5		7	V
Operating free-air temperature range	0		70	°C
Storage temperature range	-65		150	°C

Recommended Operating Conditions

PARAMETER	MIN	NOM	MAX	UNIT
VCC Supply voltage	4.75	5	5.25	V
IOH High-level output current			-400	μA
IOL Low-level output current			8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage				0.8	V
VIK Input clamp voltage	VCC=MIN, II=-18mA			-1.5	V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V			0.2	mA
IIH High-level input current	VCC=MAX, VI=2.7V		20	40	μA
IIL Low-level input current	VCC=MAX, VI=0.4V		-0.4	-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		18	34	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
FCLK Clock frequency	CL/RL=15pF/2K			35	MHz
TPLH/TPHL Propagation delay, cascade UP clock	CL/RL=15pF/2K		17	26	ns
TPLH/TPHL Propagation delay, cascade DOWN clock	CL/RL=15pF/2K		16	24	ns
TPLH/TPHL Propagation delay, clock to output	CL/RL=15pF/2K		27	38	ns
TPLH/TPHL Propagation delay, preset to output	CL/RL=15pF/2K		24	40	ns
TPLH/TPHL Propagation delay, clear to output	CL/RL=15pF/2K		23	35	ns

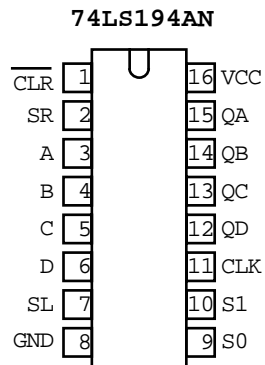
74LS194A 4-BIT UNIVERSAL SHIFT REGISTER

1

Description

This bidirectional shift register is designed to incorporate virtually all of the features a system designer may want in a shift register. The circuit features parallel inputs, parallel outputs, right-shift and left-shift serial inputs, mode-control inputs, and a direct overriding clear line. The register has four distinct modes of operation, namely: Inhibit clock (do nothing); Shift right (QA towards QD); Shift left (QD towards QA); Parallel load.

Pin Configuration (Top View)

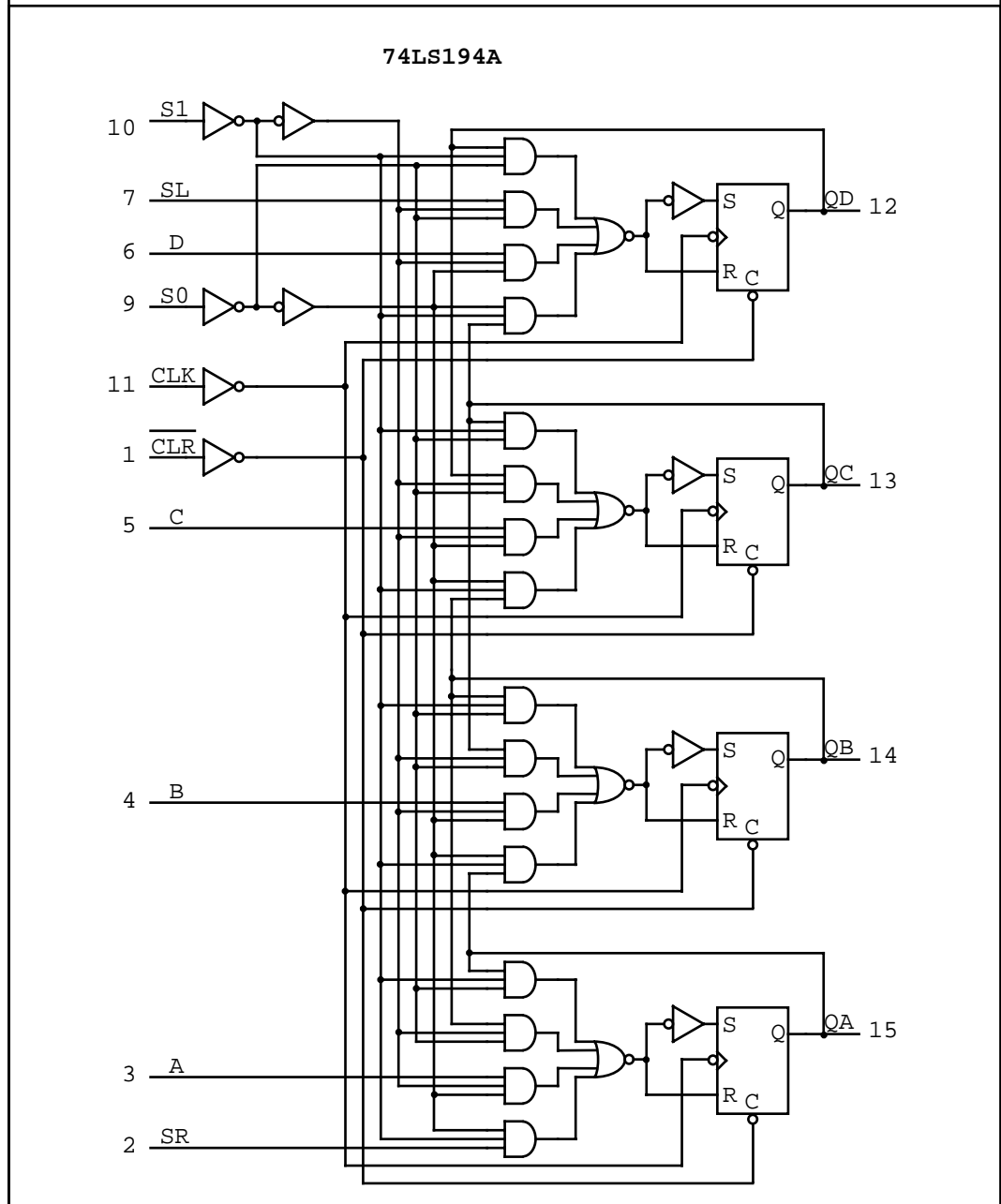


Function Table

74LS194A 4-BIT UNIVERSAL SHIFT REGISTER

74LS194A 4-BIT UNIVERSAL SHIFT REGISTER													
INPUTS										OUTPUTS			
CLR	MODE		CLK	SERIAL		PARALLEL				QA	QB	QC	QD
	S1	S0		SL	SR	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	QA	QB	QC	QD
H	H	H	↑	X	X	A	B	C	D	A	B	C	D
H	L	H	↑	X	H	X	X	X	X	H	QA	QB	QC
H	L	H	↑	X	L	X	X	X	X	L	QA	QB	QC
H	H	L	↑	H	X	X	X	X	X	QB	QC	QD	H
H	H	L	↑	L	X	X	X	X	X	QB	QC	QD	L
H	L	L	X	X	X	X	X	X	X	QA	QB	QC	QD

Logic Diagram



74LS194A 4-BIT UNIVERSAL SHIFT REGISTER**3****Maximum Ratings over Operating Free-Air Temperature Range**

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		-0.5		7	V
VI Input voltage		-0.5		7	V
Operating free-air temperature range		0		70	°C
Storage temperature range		-65		150	°C

Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		4.75	5	5.25	V
IOH High-level output current				-400	µA
IOL Low-level output current				8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage			0.8		V
VIK Input clamp voltage	VCC=MIN, II=-18mA			-1.5	V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V			0.2	mA
IIH High-level input current	VCC=MAX, VI=2.7V		20	40	µA
IIL Low-level input current	VCC=MAX, VI=0.4V		-0.4	-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		15	23	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
FMAX Clock frequency	CL/RL=15pF/2K	25	36		MHz
TPHL Propagation delay, clear to output	CL/RL=15pF/2K		19	30	ns
TPLH Propagation delay, clock to output	CL/RL=15pF/2K		14	22	ns
TPHL Propagation delay, clock to output	CL/RL=15pF/2K		17	26	ns

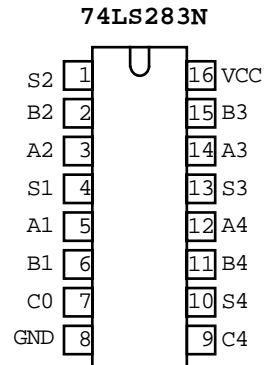
74LS283 4-BIT FULL ADDER WITH FAST CARRY

1

Description

This monolithic full adder adds two 4-bit binary words A and B, including a carry-in (C0), to give a 4-bit output word S and the carry-out (C4). There is lookahead across all four bits to generate the carry-out in 10 ns

Pin Configuration (Top View)



Function Table

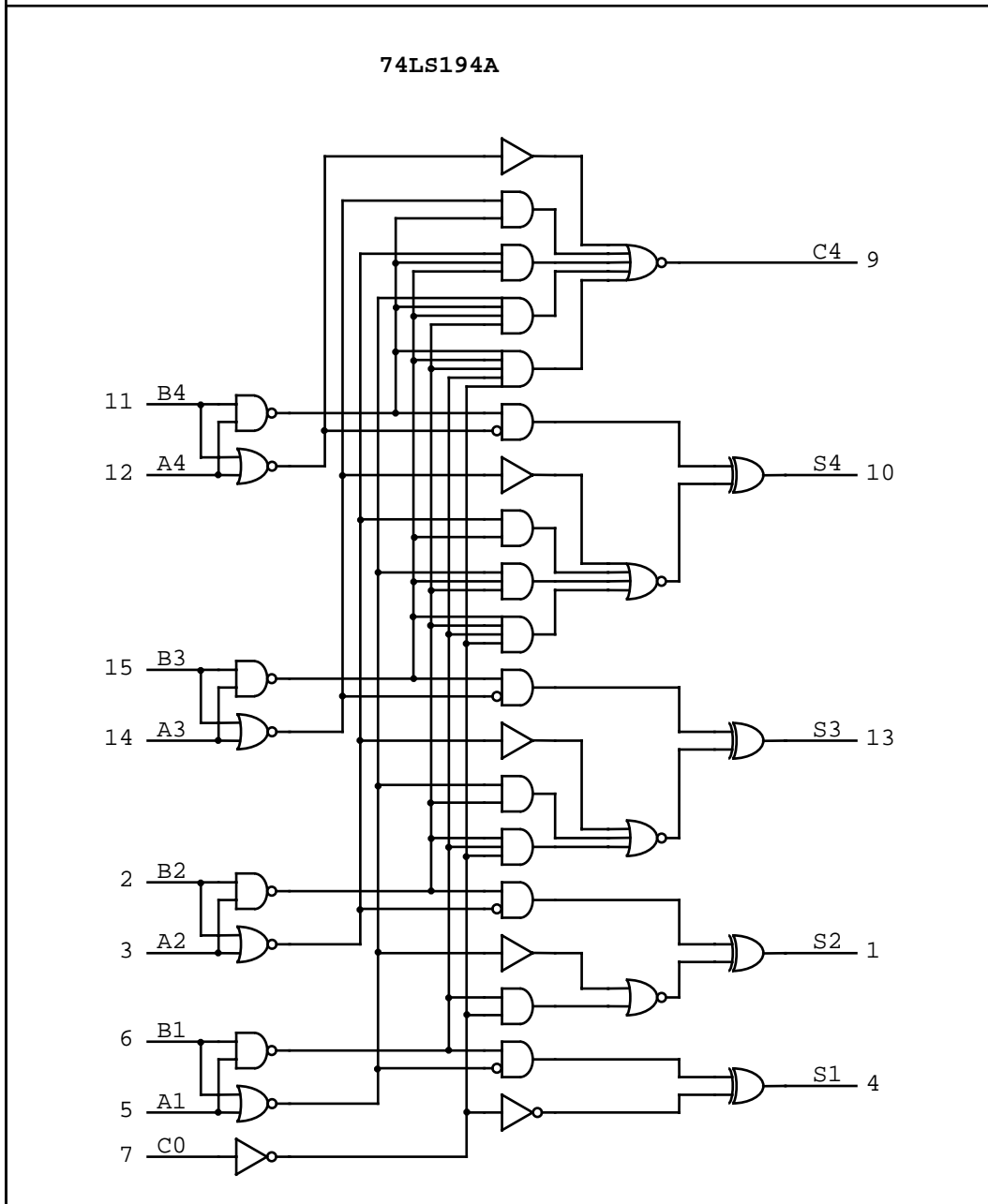
74LS283 4-BIT FULL ADDER WITH FAST CARRY

INPUTS				OUTPUTS					
				IF C0=L			IF C0=H		
				IF C2=L			IF C2=H		
A1	B1	A2	B2	S1	S2	C2	S1	S2	C2
A3	B3	A4	B4	S3	S4	C4	S3	S4	C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

74LS283 4-BIT FULL ADDER WITH FAST CARRY

2

Logic Diagram



74LS283 4-BIT FULL ADDER WITH FAST CARRY
3
Maximum Ratings over Operating Free-Air Temperature Range

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		-0.5		7	V
VI Input voltage		-0.5		7	V
Operating free-air temperature range		0		70	°C
Storage temperature range		-65		150	°C

Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
VCC Supply voltage		4.75	5	5.25	V
IOH High-level output current				-400	µA
IOL Low-level output current				8	mA

Electrical Characteristics (Operating Free-Air Temperature Range)

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
VIH High-level input voltage		2			V
VIL Low-level input voltage			0.8		V
VIK Input clamp voltage	VCC=MIN, II=-18mA		-1.5		V
VOH High-level output voltage	VCC=MIN, VIL=MAX, IOH=MAX	2.7	3.4		V
VOL Low-level output voltage	VCC=MIN, VIH=2V, IOL=MAX		0.35	0.5	V
II Input current @ max. input voltage	VCC=MAX, VI=7V		0.2		mA
IIH High-level input current	VCC=MAX, VI=2.7V		20	40	µA
IIL Low-level input current	VCC=MAX, VI=0.4V		-0.4	-0.8	mA
IOS Short-circuit output current	VCC=MAX	-20		-100	mA
ICC Supply current	VCC=MAX (grounded inputs)		20	35	mA

Switching Characteristics, VCC = 5V, TA = 25°C

PARAMETER	TEST CONDITIONS	MIN	TYP.	MAX	UNIT
TPLH/TPHL Propagation delay, C0 to any S	CL/RL=15pF/2K		16	24	ns
TPLH/TPHL Propagation delay, any AI,BI to SI	CL/RL=15pF/2K		15	24	ns
TPLH/TPHL Propagation delay, C0 to C4	CL/RL=15pF/2K		11	20	ns
TPLH/TPHL Propagation delay, any AI,BI to C4	CL/RL=15pF/2K		11	17	ns

Index

- Alkaline, 54
- analog, 73
- AND-gate, 73

- batteries, 46, 55, 65, 66
- Battery Holder, 46, 49, 54, 65, 66
- BCD Decoder, 73
- behaviour, 72
- bottom-up, 79, 80
- bug, 77, 81
- Button, 46, 47, 49, 51, 67, 75

- CAD, 78
- CAM, 78
- capacitor, 46, 73
- chip, 46
- circuit design, 82
- circuit diagram, 57, 71, 73, 82
- complex, 78
- complexity, 79, 81
- component, 45, 58
- Connecting Wire, 46, 47, 49–51, 55, 61, 62, 65–67, 69, 83
- connection, 72
- Connector Block, 46, 47, 49, 57, 59, 61, 65
- construction, 82
- cycle, 67

- debugging, 77
- demonstration, 12
- diagnosing, 77
- diagnostic test, 78, 83
- digital logic, 52, 71–73
- digital logic family, 52
- Digital Logic Lab Board, 45
- digital system, 45, 57, 61, 71

- DIL, 46, 49, 52, 61, 73
- diode, 46
- dual-in-line, 46, 52

- earthed, 44
- electrical shock, 43
- electromechanical, 73
- electronic component, 46, 57, 65
- electronic testing, 65
- exam questions, 12
- exhaustive test, 79, 81
- Experiment, 41

- fault, 77
- function, 52, 57, 71–73
- functional name, 72, 73

- go/no go test, 78
- graphical, 71, 72

- hole, 49, 50, 58, 65

- IC, 43, 46, 49, 51, 52, 61, 66, 69, 83
- IC pin, 82
- IC technology, 52
- input, 72
- integrated circuit, 43, 46, 52
- interface, 51

- Late Assessment, 11
- LED, 46, 47, 49, 51, 66, 67, 69, 75, 80
- light-emitting diode, 46
- logic design, 81
- logic diagram, 71, 72
- logic function, 71

- manufacturer, 52
- multimeter, 55, 65, 69

oscillating electrical signal, 46
Oscillator, 46, 47, 49, 51, 67, 76
oscilloscope, 65
output, 72

packaging, 46, 52
partner, 11
Patchboard, 46, 47, 49, 51, 61, 65, 66
PC, 46
PC Connector, 46, 49
PCB, 43, 46, 49
peripheral component, 50, 51
physical design, 57
pin, 43, 46, 49, 50, 52, 57, 58, 61, 73
power, 73
Power Supply, 49, 51, 66, 67, 69, 74
printed circuit board, 43

rectifying, 77
report, 11, 12
resistor, 46, 73
rocker switch, 46

self-test, 65
signal, 72
single-in-line, 46
Size C, 54
soldering, 82
static electricity, 44
structural design, 57, 72, 73
structure, 71
subsystem, 71, 73
Switch, 46, 47, 49, 51, 69, 74, 80
symbol, 72

test point, 80
test rig, 79, 80
testing, 77
testing check list, 80
textual, 71
top-down, 79
track, 46, 51

voltage, 55, 69

welding, 82
wire-stripper, 43, 55, 62
wiring error, 57
wiring list, 57, 58, 63, 65, 71, 72, 79,
82