**QUALITY POLICY**

To be a Global Leader of Innovative, Competitive and Eco friendly Electronic Equipment, Software Products and Turn-key Solutions for Industry and Technology Training. We will achieve this by enhancing Customer Satisfaction based on Research, Modern manufacturing techniques and continuous improvement in Quality of the products and the Services. The key drivers to growth will be focus on Employees and Customers, Use of latest Technology, Intense Marketing, Service Support and High Ethical Practices.

An ISO 9001: 2000 company

[Scientech Technologies Pvt. Ltd.](http://www.scientech.bz)
4-Bit Parallel Adder/ Subtractor

Table of Contents

1. Introduction 4
2. Theory 5
3. Experiment 1 To study parallel adding of 4 bit with carry. 13
4. Experiment 2 To study parallel subtraction of 4 bit with carry. 16
5. Datasheet 19
6. Warranty 22
7. List of Service Centers 23
8. List of Accessories 24

Introduction

DB19 is a compact, easy to use digital logic experiment board. This is useful for the students to get acquainted with the basic arithmetic process (addition and subtraction) used in the microprocessors. This board is especially designed to be used with SCIENTECH Digital Lab ST2611 which has built in fixed and variable DC power supplies, clock source, logic high and low input facility with output indicators, logic probe, seven segment LEDs for direct readings.

List of Boards:

<table>
<thead>
<tr>
<th>Model</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB01</td>
<td>Logic Gates</td>
</tr>
<tr>
<td>DB02</td>
<td>Universal Gate- NAND/NOR</td>
</tr>
<tr>
<td>DB03</td>
<td>EX-OR Gate Implementation</td>
</tr>
<tr>
<td>DB04</td>
<td>Demorgan's Theorem</td>
</tr>
<tr>
<td>DB05</td>
<td>EX-OR Gate Application</td>
</tr>
<tr>
<td>DB06</td>
<td>Code Conversion (Binary to Gray &amp; Gray to Binary)</td>
</tr>
<tr>
<td>DB07</td>
<td>Code Conversion (BCD to Excess-3 code)</td>
</tr>
<tr>
<td>DB08</td>
<td>Binary Adder -Subtractor</td>
</tr>
<tr>
<td>DB09</td>
<td>Encoder - Decoder</td>
</tr>
<tr>
<td>DB10</td>
<td>Multiplexer – Demultiplexer</td>
</tr>
<tr>
<td>DB11</td>
<td>Encoder- Decoder</td>
</tr>
<tr>
<td>DB12</td>
<td>Shift register (4 bit SIPO)</td>
</tr>
<tr>
<td>DB13</td>
<td>4 Bit Synchronous Binary Counter</td>
</tr>
<tr>
<td>DB15</td>
<td>BCD to 7- Segment Decoder</td>
</tr>
<tr>
<td>DB16</td>
<td>Digital to Analog Converter (R-2R ladder)</td>
</tr>
<tr>
<td>DB17</td>
<td>3 Digit Event Counter</td>
</tr>
<tr>
<td>DB21</td>
<td>Fiber Optic Digital Link</td>
</tr>
<tr>
<td>DB27</td>
<td>Transfer Characteristics (TTL and CMOS Inverters)</td>
</tr>
<tr>
<td>DB28</td>
<td>Monostable Multivibrator</td>
</tr>
<tr>
<td>DB29</td>
<td>CMOS and Crystal Oscillator</td>
</tr>
<tr>
<td>DB30</td>
<td>Adder/ Subtractor (4-Bit/8-Bit)</td>
</tr>
<tr>
<td>DB31</td>
<td>Decoder/Demultiplexer</td>
</tr>
<tr>
<td>DB32</td>
<td>Modulo-N programmable counter</td>
</tr>
<tr>
<td>DB34</td>
<td>4 Bit Magnitude Comparator</td>
</tr>
<tr>
<td>DB35</td>
<td>4 Bit Shift Register</td>
</tr>
</tbody>
</table>

...and many more
**Theory**

Before going into the architecture of a binary adder/subtractor, it is necessary at this point to understand how positive and negative numbers are represented in binary notation.

**Basic Concepts Behind the Binary System :**

To understand binary numbers, begin by recalling elementary school math. When we first learned about numbers, we were taught that, in the decimal system, things are organized into columns:

```
H | T | O
1 | 9 | 3
```

Such that "H" is the hundreds column, "T" is the tens column, and "O" is the ones column. So the number "193" is 1-hundreds plus 9-tens plus 3-ones.

Years later, we learned that the ones column meant $10^0$, the tens column meant $10^1$, the hundreds column $10^2$ and so on, such that

```
10^2|10^1|10^0
1 | 9 | 3
```

The number 193 is really $\{(1*10^2) + (9*10^1) + (3*10^0)\}$.

As you know, the decimal system uses the digits 0-9 to represent numbers. If we wanted to put a larger number in column $10^n$ (e.g., 10), we would have to multiply $10*10^n$, which would give $10^{n+1}$, and be carried a column to the left. For example, putting ten in the $10^0$ column is impossible, so we put a 1 in the $10^1$ column, and a 0 in the $10^0$ column, thus using two columns. Twelve would be $12*10^0$, or $10^0(10+2)$, or $10^1+2*10^0$, which also uses an additional column to the left (12).

The binary system works under the exact same principles as the decimal system, only it operates in base 2 rather than base 10. In other words, instead of columns being

```
10^2|10^1|10^0
```

They are

```
2^2|2^1|2^0
```

Instead of using the digits 0-9, we only use 0-1 (again, if we used anything larger it would be like multiplying $2*2^n$ and getting $2^n+1$, which would not fit in the $2^n$ column. Therefore, it would shift you one column to the left. For example, "3" in binary cannot be put into one column. The first column we fill is the right-most column, which is $2^0$, or 1. Since $3>1$, we need to use an extra column to the left, and indicate it as "11" in binary $(1*2^1) + (1*2^0)$.

**Examples :**

What would the binary number 1011 be in decimal notation?

Try converting these numbers from binary to decimal:

- $10$
- $111$
- $10101$
- $11110$

Remember :

```
2^4|2^3|2^2|2^1|2^0
1 | 1 | 1 | 1 | 0
```

**Binary Addition :**

Consider the addition of decimal numbers :

```
23
+ 48
```

We begin by adding $3+8=11$. Since 11 is greater than 10, a one is put into the 10's column (carried), and a 1 is recorded in the one's column of the sum. Next, add $\{(2+4) +1\} \text{ (the one is from the carry) } =7$, which is put in the 10's column of the sum. Thus, the answer is 71.

Binary addition works on the same principle, but the numerals are different. Begin with one-bit binary addition :

```
2^1|2^0
```

They are

```
2^1|2^0
```
1+1 carries us into the next column. In decimal form, 1+1=2. In binary, any digit higher than 1 puts us a column to the left (as would 10 in decimal notation). The decimal number "2" is written in binary notation as "10" (1*2^1) + (0*2^0). Record the 0 in the ones column, and carry the 1 to the twos column to get an answer of "10." In our vertical notation,

```
  1
+1
  10
```

The process is the same for multiple-bit binary numbers:

```
  1010
+1111
_____
```

- Step one : Column 2^0: 0+1=1. Record the 1. Temporary Result: 1; Carry: 0
- Step two : Column 2^1: 1+1=10. Record the 0, carry the 1. Temporary Result: 01; Carry: 1
- Step three : Column 2^2: 1+0=1 Add 1 from carry: 1+1=10. Record the 0, carry the 1. Temporary Result: 001; Carry: 1
- Step four : Column 2^3: 1+1=10. Add 1 from carry: 10+1=11. Record the 11. Final result: 11001

Alternately:

```
  11 (carry)
1010
+1111
11001
```

Always remember:

- 0+0=0
- 1+0=1
- 1+1=10

Try a few examples of binary addition:

```
111    101    111
  +110    +111    +111
_______  ______  ______
```

**Two’s Complement Representation**:

Two's complement is the most popular method of signifying negative integers in computer science. It is also an operation of negation (converting positive to negative numbers or vice versa) in computers which represent negative numbers using two's complement. Its use is ubiquitous today because it doesn't require the addition and subtraction circuitry to examine the signs of the operands to determine whether to add or subtract, making it both simpler to implement and capable of easily handling higher precision arithmetic. As well, 0 has only a single representation, obviating the subtleties associated with negative zero.

In an n-bit binary number, the most significant bit is usually the $2^{n-1}$'s place. But in the two's complement representation, its place value is negated; it becomes the $-2^{n-1}$'s place and is called the sign bit. If the sign bit is zero, the value is non-negative, the same as an ordinary binary number. But if the sign bit is 1, the value is negative. To negate a two's complement number, invert all the bits then add 1 to the result.

If all bits are 1, the value is $-1$. If the sign bit is 1 but the rest of the bits are 0, the value is the most negative number, $-2^{n-1}$ for an n-bit number. The absolute value of the most negative number cannot be represented with the same number of bits.

A two's complement 8-bit binary numeral can represent every integer in the range $-128$ to $+127$. If the sign bit is 0, then the largest value that can be stored in the remaining seven bits is $2^7 - 1$, or 127.

Using two's complement to represent negative numbers allows only one representation of zero, and to have effective addition and subtraction while still having the most significant bit as the sign bit.
Calculating two's complement:

In finding the two's complement of a binary number, the bits are inverted, or "flipped", by using the bitwise NOT operation; the value of 1 is then added to the resulting value. Bit overflow is ignored, which is the normal case with zero.

For example, beginning with the signed 4-bit binary representation of the decimal value 5:

\[(5)_{10} = 0101\]

The first bit is 0, so the value represented is indeed a positive 5. To convert to -5 in two's complement notation, the bits are inverted; 0 becomes 1, and 1 becomes 0:

\[1010\]

At this point, the numeral is the ones' complement of the decimal value 5. To obtain the two's complement, 1 is added to the result, giving:

\[1011 (-5)\]

The result is a signed binary numeral representing the decimal value -5 in two's complement form. The most significant bit is 1, so the value is negative.

The two's complement of a negative number is the corresponding positive value. For example, inverting the bits of -5 (above) gives:

\[0100\]

And adding one gives the final value:

\[0101 (5)\]

The decimal value of a two's complement binary number is calculated by taking the value of the most significant bit, where the value is negative when the bit is one, and adding to it the values for each power of two where there is a one. Example:

\[1011 (-5) = -8 + 0 + 2 + 1 = (-2^7 + 2^6 + ...) = -5\]

Note that the two's complement of zero is zero: inverting gives all ones, and adding one changes the ones back to zeros (the overflow is ignored). Also the two's complement of the most negative number representable (e.g. a one as the sign bit and all other bits zero) is itself. This happens because the most negative number's "positive counterpart" is occupied by "0", which gets classed as a positive number in this argument. Hence, there appears to be an 'extra' negative number.

A more formal definition of two's complement negative number (denoted by \(N^*\) in this example) is derived from the equation \(N^* = 2^n - N\), where \(N\) is the corresponding positive number and \(n\) is the number of bits in the representation.

For example, to find the 4 bit representation of -5:

\[N (\text{base } 10) = 5, \text{ therefore } \overline{N} (\text{base } 2) = 0101\]

\[n = 4\]

Hence:

\[N^* = 2^n - N = [2^4]_{\text{base2}} - 0101 = 10000 - 0101 = 1011\]

N.B. You can also think of the equation as being entirely in base 10, converting to base 2 at the end, e.g.:

\[N^* = 2^n - N = 2^4 - 5 = [11]_{\text{base10}} = [1011]_{\text{base2}}\]

Obviously, "\(N^* = 11\)" isn't strictly true but as long as you interpret the equals sign as "is represented by", it is perfectly acceptable to think of two's complements in this fashion.

Subtraction:

Subtraction in binary is done by adding two's complement numbers requires no special processing if the operands have opposite signs: the sign of the result is determined automatically. For example, subtracting 5 from 15:

\[\begin{array}{c}
15 \\
- 5 \\
10
\end{array}\]

Which become

\[\begin{array}{c}
1111 \\
- 0101
\end{array}\]

2's complement of 5 is

1's complement of 5 = 1010

\[\begin{array}{c}
\hline
+ 1 \\
1011
\end{array}\]
Now adding them will give undesired results.

\[
\begin{align*}
111 & \quad \text{(carry)} \\
1111 & \quad \text{(15)} \\
+1011 & \quad \text{(-5)} \\
1010 & \quad \text{(10)}
\end{align*}
\]

This process depends upon restricting to 4 bits of precision; a carry to the (nonexistent) 4th most significant bit is ignored, resulting in the arithmetically correct result of 10.

The last two bits of the carry row (reading right-to-left) contain vital information: whether the calculation resulted in an arithmetic overflow, a number too large for the binary system to represent (in this case greater than 4 bits). An overflow condition exists when a carry (an extra 1) is generated into but not out of the far left sign bit, or out of but not into the sign bit. As mentioned above, the sign bit is the leftmost bit of the result.

In other terms, if the last two carry bits (the ones on the far left of the top row in these examples) are both 1's or 0's, the result is valid; if the last two carry bits are "1 0" or "0 1", a sign overflow has occurred. Conveniently, an XOR operation on these two bits can quickly determine if an overflow condition exists.

Computers usually use the method of complements to implement subtraction. But although using complements for subtraction is related to using complements for representing signed numbers, they are independent; direct subtraction works with two's complement numbers as well. Like addition, the advantage of using two's complement is the elimination of examining the signs of the operators to determine if addition or subtraction is needed.

**The Adder / subtractor :**

We can connect full adders as shown in the figure 1 given below to add or subtract binary numbers. The circuit is laid out from right to left, similar to the way we add binary numbers. Therefore the least significant column is on the right, and the most significant column is on the left. The boxes labeled FA are full adders. The Carry out from each full adder is the carry in to the next higher full adder. The numbers being processed are \( A_3 A_2 A_1 A_0 \) and \( B_3 B_2 B_1 B_0 \), while the answer is \( S_3 S_2 S_1 S_0 \). With 4-bit arithmetic, the final carry is ignored for reasons given earlier. With 8-bit arithmetic, the final carry is the carry into the addition of the upper bytes.

**Addition :**

During an addition, the **Add/Sub** signal is deliberately kept in the low state, i.e. at '0' level. Therefore the binary number \( B_3 B_2 B_1 B_0 \) passes through the controlled inverter with no change. The full adder then produces the correct sum.

**Subtraction :**

During a subtraction, the **Add/Sub** signal is deliberately kept in the high state, i.e. at '1' level. Therefore the controlled inverter produces the one's complement of \( B_3 B_2 B_1 B_0 \). Furthermore, because **Sub** is the carry in to the first full adder, which is equivalent to adding 1 to the one's complement of \( B_3 B_2 B_1 B_0 \), and thus producing the correct result. This provide the 2's complement of \( B_3 B_2 B_1 B_0 \) which being added to \( A_3 A_2 A_1 A_0 \).
Experiment 1

Objective:
To study 4 Bit Parallel adder.

Apparatus required:
1. Digital board, DB19.
2. DC Power Supply of +5V.

Logic diagram:
(Logic 1 = +5V & Logic 0= GND)

Procedure:
• To Study the Addition procedure of 4 Bit Parallel Adder precede as follow:
  1. Connect +5V and ground to their indicated position on DB19 experiment board from DC power supply.
  2. For the Addition of 8 and 3 i.e. 8+3.
  3. Set $A_4 A_3 A_2 A_1$ at the binary value of 8; i.e. 1000 respectively
  4. Set $B_4 B_3 B_2 B_1$ at the binary value of 3; i.e. 0011 respectively
  5. Set the Add/Sub Switch at Add i.e. 0 level or low level
  6. Switch ON the power supply.
  7. Observe output on LED and match it with the truth table Table1 given in the theory.
  8. Note the LED output for $S_4 S_3 S_2 S_1$ and carry bit.
  9. Note the output at observation table.
 10. The output will be 11 that is i.e. output at $S_4 S_3 S_2 S_1$ will be equal to 1011 and carry will be ‘0’.
 11. Repeat above steps 2-10 for different values

Calculation:
1. Add 12 and 10

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>+10</td>
<td>+1010</td>
</tr>
<tr>
<td>22</td>
<td>10110</td>
</tr>
</tbody>
</table>

\[2^3(1) + 2^2(0) + 2^1(1) + 2^0(0) = 16 + 0 + 4 + 2 + 0 = 22\]
2. Add 15 and 2

\[
\begin{array}{c|c}
15 & 1111 \\
+ 2 & + 0010 \\
17 & 10001 \\
\end{array}
\]

\[2^3(1) + 2^2(0) + 2^1(0) + 2^0(1) + 2^0(1) = 16 + 0 + 0 + 0 + 0 = 17\]

Observation table:

<table>
<thead>
<tr>
<th>S. No.</th>
<th>A_4</th>
<th>A_3</th>
<th>A_2</th>
<th>B_4</th>
<th>B_3</th>
<th>B_2</th>
<th>B_1</th>
<th>Carry</th>
<th>S_4</th>
<th>S_3</th>
<th>S_2</th>
<th>S_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Experiment 2

Objective:
To study 4 Bit Parallel Subtractor.

Apparatus required:
1. Digital board, DB19.
2. DC Power Supply of +5V.

Logic diagram:
(Logic 1 = +5V & Logic 0 = GND)

Fig. 3
Procedure:

- To Study the Addition procedure of 4 Bit Parallel Subtractor precede as follow:

1. Connect +5V and ground to their indicated position on DB19 experiment board from DC power supply.
2. For the Subtraction of 8 and 3 i.e. 8-3.
3. Set A₄ A₃ A₂ A₁ at the binary value of 8; i.e. 1000 respectively
4. Set B₄ B₃ B₂ B₁ at the binary value of 3; i.e. 0011 respectively
5. Set the Add/Sub Switch at Sub i.e. 1 level or High level
6. Switch ON the power supply.
7. Observe output on LED and match it with the truth table Table1 given in the theory.
8. Note the LED output for S₄ S₃ S₂ S₁ and carry bit which also use as Borrow bit.
9. Note the output at observation table.
10. The output will be ‘5’ that is i.e. output at S₄ S₃ S₂ S₁ will be equal to 0101 and carry will be ‘1’.
11. Repeat above steps 2-10 for different values

Note: Always take the value of A higher then B in case of subtraction. The Borrow/Carry bit won’t count in Subtraction

Calculation:

1. Sub 12 and 10

<table>
<thead>
<tr>
<th>A₄</th>
<th>A₃</th>
<th>A₂</th>
<th>A₁</th>
<th>B₄</th>
<th>B₃</th>
<th>B₂</th>
<th>B₁</th>
<th>Carry</th>
<th>S₄</th>
<th>S₃</th>
<th>S₂</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$2^{3}(0) + 2^{2}(0) + 2^{1}(1) + 2^{0}(0) = 0 + 0 + 2 + 0 = 2$
**SN54/74LS83A**

**FUNCTIONAL DESCRIPTION**

The 74LS83 is a 4-bit binary full adder. It takes in two 4-bit binary numbers and provides a sum (S) and a carry (C) output. The Carry output is generated based on the carry generate (CyG) and carry propagate (CyP) conditions. The device is useful in digital systems for performing addition operations.

**Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C0</th>
<th>S</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Operating Ranges**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Voltage</th>
<th>MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>5V</td>
<td>25</td>
</tr>
</tbody>
</table>

**Connection Diagram**

[Diagram of the 74LS83 circuitry and pinout]

---

**Ordering Information**

- SN54LS83C
- SN54LS83D
- SN74LS83C
- SN74LS83D
## Warranty

1. We guarantee the instrument against all manufacturing defects during 24 months from the date of sale by us or through our dealers.

2. The guarantee covers manufacturing defects in respect of indigenous component and material limited to the warranty extended to us by the original manufacturer, and defect will be rectified as far as lies within our control.

3. The guarantee will become INVALID.
   - a) If the instrument is not operated as per instruction given in the instruction manual.
   - b) If the agreed payment terms and other conditions of sale are not followed.
   - c) If the customer resells the instrument to another party.
   - d) Provided no attempt have been made to service and modify the instrument.

4. The non-working of the instrument is to be communicated to us immediately giving full details of the complaints and defects noticed specifically mentioning the type and sr. no. of the instrument, date of purchase etc.

5. The repair work will be carried out, provided the instrument is dispatched securely packed and insured with the railways. To and fro charges will be to the account of the customer.

### DESPATCH PROCEDURE FOR SERVICE

Should it become necessary to send back the instrument to factory please observe the following procedure:

1. Before dispatching the instrument please write to us giving full details of the fault noticed.

2. After receipt of your letter our repairs dept. will advise you whether it is necessary to send the instrument back to us for repairs or the adjustment is possible in your premises.

Dispatch the instrument (only on the receipt of our advice) securely packed in original packing duly insured and freight paid along with accessories and a copy of the details noticed to us at our factory address.
### List of Service Centers

1. Scientech Technologies Pvt. Ltd.
   90, Electronic Complex
   Pardeshipura,
   INDORE – 452010
   Ph : (0731) 2570301
   Email : info@scientech.bz
   Fax : (0731) 2555643

2. Scientech Technologies Pvt. Ltd.
   First Floor, C-19,
   F.I.E., Patparganj Industrial Area,
   DELHI – 110092
   Ph : (011) 22157370, 22157371
   Fax : (011) 22157369
   Email : ndel@scientech.bz

3. Scientech Technologies Pvt. Ltd.
   1st floor No 1, Karpagam Gardens
   Main Road Adyar,
   CHENNAI – 600020
   Ph : (044) 42187548, 42187549
   Fax : (044) 42187549
   Email : chennai@scientech.bz

4. Scientech Technologies Pvt. Ltd.
   202/19, 4th main street
   Ganganagar,
   BANGALORE- 560032
   Ph : (080) 41285011
   Fax : (080) 41285022
   Email : bangalore@scientech.bz

5. Scientech Technologies Pvt. Ltd.
   8,1st floor, 123-Hariram Mansion,
   Dada Saheb Phalke road,
   Dadar (East),
   MUMBAI –400014
   Ph : (022) 56299457
   Fax : (022) 24168767
   Email : stplmum@scientech.bz

   988, Sadashiv Peth,
   Gyan Prabodhini Lane,
   PUNE – 411030
   Ph : (020) 24461673
   Fax : (020) 24482403
   Email : pune@scientech.bz

7. Scientech Technologies Pvt. Ltd
   SPS Apartment, 1st Floor
   2, Ahmed Mamoji Street,
   Behind Jaiwal Hospital,
   Liluah, HOWRAH-711204 W.B.
   Ph : +(033) 65266800
   Email : kolkata@scientech.bz
   Mob No : 9433029888

8. Scientech Technologies Pvt. Ltd
   Flat No. 205, 2nd Floor,
   Lakshminaryana Apartments
   ‘C’ wing, Street No. 17,
   Himaytngar,
   HYDERABAD- 500029
   Ph : (040) 55465643
   Email : hyd@scientech.bz
   Mob No : 9247712763

### List of Accessories

1. 2mm patch cord (red) 16”................................. 1 No.
2. 2mm patch cord (black) 16”.............................. 1 No.