BenchSystem™, Getting Started

Tools for Scientists and Engineers

This 4-page setup guide will show you how to assemble a BenchSystem, create a simple program, and capture data to a file.

Hardware Setup

One controller and up to 31 I/O cards can be connected per system. BenchSystem cards can be stacked to minimize the system footprint, or cards can be separated by thousands of feet.

Electrostatic discharge (ESD) is always a concern when handling electronic devices. Wear grounding straps if necessary. Try to first touch the ground of the board when picking it up. All four mounting holes and the LIB connector shrouds connect to the ground.

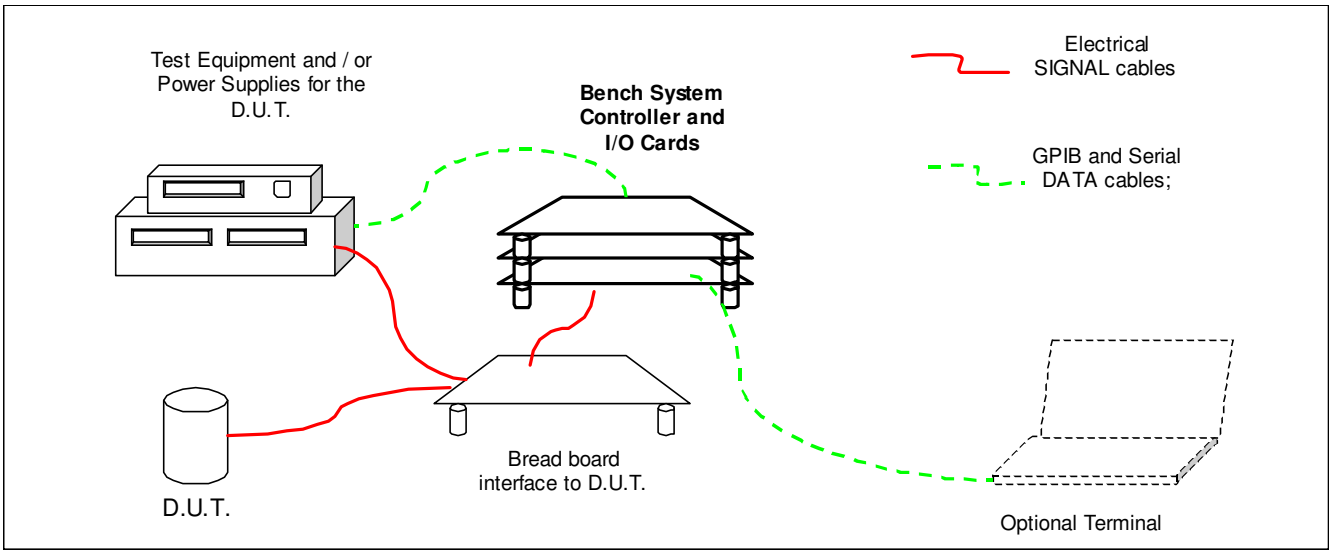

fig-1. A typical Bench control system configuration.

Mount the Cards

Mounting holes are connected to the 12V power ground on each card. Nylon standoffs can be used to isolate the cards from chassis. Make sure there is at least 0.1 inch of clearance beneath the IO connector pins that protrude through the card.

Mount the cards close to the target device and make up any distance from the rest of the system using the long reach of the LIB bus.

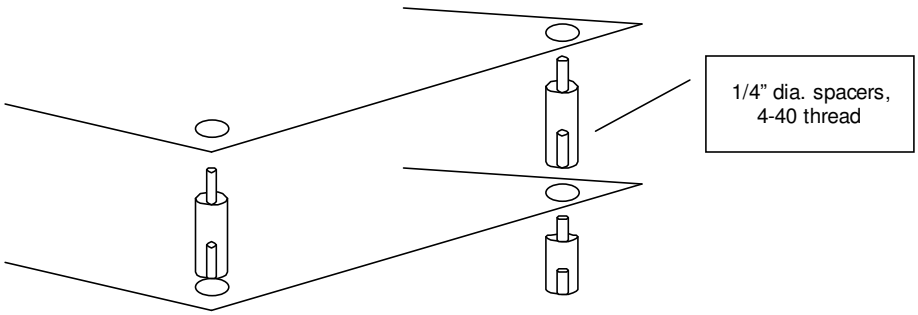

fig-2. A card stacking example using male-female spacers.

Connect the Cards to Power and Communications

The Local Interface Bus (LIB) provides communications and power for the controller and up to 31 I/O cards. A pair of phone-style connectors is located at the rear corner of each card. These connectors are electrically identical to facilitate daisy chaining the bus. Pay attention to the voltage drops on long runs; as some cards can draw 250 mA. A 120 ohm termination resistor between COMM A and COMM B is required at one end of the LIB for short runs, and at both ends for long runs (greater than 20 feet).

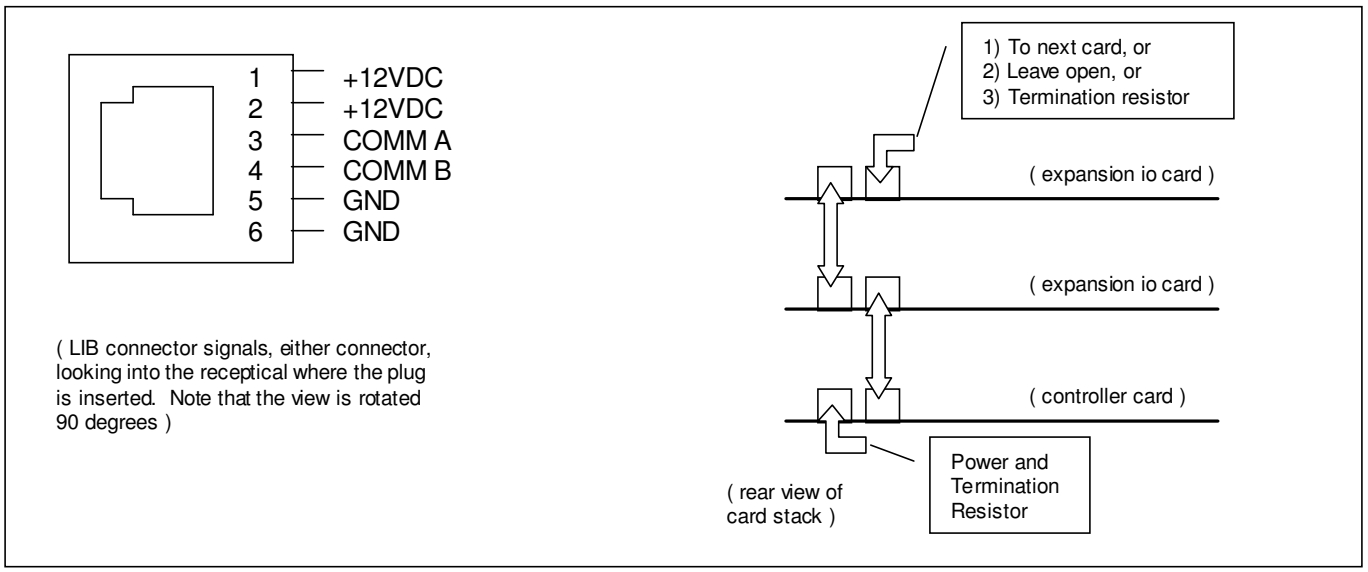

fig-3. Pinouts and a typical card configuration with LIB wiring.

Connect the I/O Cards to the Control and Measurement Signals

A breadboard is often the tidiest way to interface between the Device Under Test (DUT) and the various supplies and test equipment. Provide connectors for every interface. Consider screw terminal blocks and test points instead of dangling wires. Label the connections for future set up. User interface buttons, display modules, and lamps can also be attached to the breadboard.

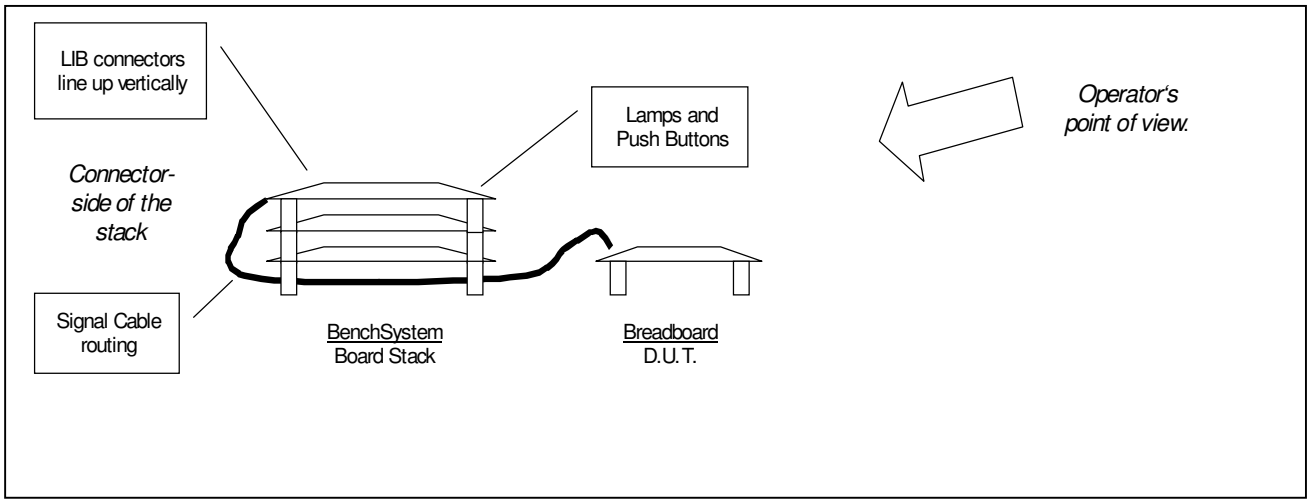

fig-4. Suggested Stack assembly and Signal Cable routing to give the operator clear access to the Breadboard.

Controller Card Setup

Connect the test equipment as shown in figure 5, using GPIB and 9-pin serial modem cables. You will need to know the address settings of the equipment on the GPIB port.

Connect a PC or Workstation to the Console port using a modem cable. Configure the Console Terminal per the following example which assumes a Windows platform running HyperTerminal:

> Start:Programs:Accessories:Communications:HyperTerminal.
> HyperTerm:Properties:Connect to: ***Direct to Comm***:Configure:***9600 baud, 8 data, No Parity, 1 stop, none*** flow control.
> HyperTerm:Properties:Settings: ***ANSI*** emulation: ASCII Setup: ***Send line feeds, Echo typed characters***

Note: Some terminals use the term **'half duplex'** instead of **'echo'**. Set them to half duplex.

Turn On the 12Vdc power. The Power Indicators on all cards should be ON. All Status indicators should be OFF. Look for a greeting on the Console terminal.

Troubleshooting: If there is no greeting on the console, press "RESET" on the controller to try again. If that fails, verify your terminal settings and modem cable.

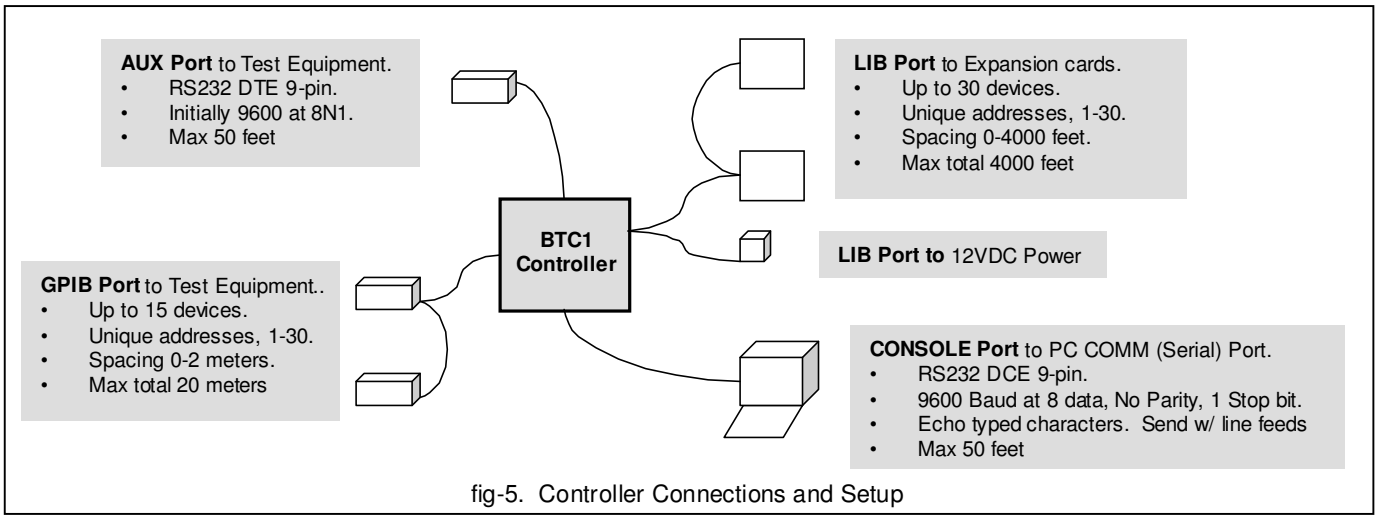

fig-5. Controller Connections and Setup

Set the I/O Card Addresses

Every I/O card on the Local Interface Bus must have a unique address between 1-31. If two or more cards have the same address the system will lock up when commands are sent to the cards.

It is a good idea to use *1-30 for card addresses* and use 31 for maintenance. The controller card has no address.

Change the address on each and every I/O card per the following example, replacing the '4' with unique values from 1 through 30. Use the **Backspace** key, not the mouse, when fixing typos in a command line.

In the following example we set the address of an I/O card to 4:

> At the console, type **'putlib 31,a4?'** and press ENTER with one hand, *while* holding in the card's **ADDR** button with another.
>
> See the address in the second argument of the response. If there is no response press **ctrl-c** to unlock the controller, and try it again.

fyi: The card will now respond to this address, i.e. enter **'putlib 4,?'** (*without* pressing the ADDR button) to read its status again.

fyi: To simply read the unknown address of any I/O card, enter **'putlib 31,?'** *while* holding in the card's **ADDR** button. If another card on the bus were set to address 31 then there would be a conflict here, which is why we should use 31 for maintenance only.

Create and Run a Program

Open a text editor on your computer and type in the following lines of code (use spaces, tabs and blank lines to give it an organized appearance). Note that the 'start' and 'quit' labels (in the left-hand column) are followed by colons; the rest are statements and are followed by semicolons. Be sure to press Enter after typing the last line of the program. You can use any editor that can "Save As Text". See the "BSOP datasheet.pdf" for complete information about the language and syntax.

```
int       x ;

start:    if ( x > 4 )  goto  quit ;
          x = x + 1 ;
          console ( x @ "," ) ;
          goto  start ;

quit:     exit ;
```

Save the program on your computer as a **text** file, call it demo1.txt. We will now get ready to upload the program to the controller.

Type (and Enter) **help** at the terminal to see a list of the Console commands and syntax. Use the **Backspace** key, not the mouse, to reposition the cursor when fixing typos in a command line.

> The controller will sometimes "**lock up**" while waiting for an instrument response that isn't coming ; due to a wrong address, etc. When this happens press **ctrl-c** at the terminal, or press 'Reset' at the controller card, to return to Console mode.

Type (and Enter) **load** at the console terminal. There should be a prompt to press ESC when done loading. Now upload demo1.txt from the console terminal, i.e. with HyperTerminal pull down **Transfer:Send Text File**, find demo1.txt, and send it. Press the **ESC** key after the entire file has transferred.

Type (and Enter) **list** to verify the program is as you expect it to be. To make changes: edit the file, save it, and reload it.

Type (and Enter) **run** to run the program. The output should be "1,2,3,4,5, " followed on the next line by the prompt '>' without any error messages.

Edit the file: Insert the statement **'waitms 1000;'** after the 'console…;' statement to slow down the loop. Load and run it.

Type (and Enter) **step** to single-step through each statement and observe the flow of the program. Press the space bar to step the program. If you don't get the output numbers, enable hyperterminal:settings:ASCIIsetup:Append incoming linefeeds; because the output is being overwritten.

To use HyperTerminal to capture data to a file, pull down **Transfer:Capture Text** and create a text file. All subsequent transfers from the controller will go into the file. Run demo1.txt and then terminate the capture **Transfer:Capture Text:stop**. You can open the file with your text editor to look it over, clean it up, etc.

The program will stay on the controller until another is loaded, power is cycled, or the Reset Button is pressed. Type **save** to store the program in the controller's non-volatile memory, so it will run automatically at Power up or at Reset. Try it. Type **erase** to remove the program from non-volatile memory. Try it.

That's it! **You will now be able to automate testing and measurement operations by yourself.** Dig into the data sheets and enjoy your independence.