Electronics for Model Railways

Chapter 17

EzyBus

By Davy Dick

Electronics for Model Railways

By Davy Dick

In memory of Margaret

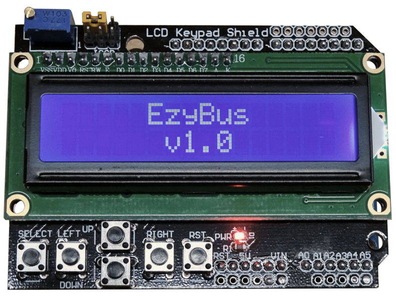
Contents

- Chapter 1 - Basic Electronics
- Chapter 2 - Motors and DC controllers
- Chapter 3 – Layout wiring
- Chapter 4 – Track wiring
- Chapter 5 – Point wiring
- Chapter 6 – Point motors & servos
- Chapter 7 – Power supplies & cutouts
- Chapter 8 – Batteries
- Chapter 9 – Digital Command Control
- Chapter 10 – Track occupancy detectors
- Chapter 11 - RFID
- Chapter 12 - Scenic lighting
- Chapter 13 – Train lighting
- Chapter 14 - Adding sound
- Chapter 15 - Animations
- Chapter 16 - CBUS
- Chapter 17 - EzyBus
- Chapter 18 – Interfacing techniques
- Chapter 19 - Construction methods
- Chapter 20 – Transistors, ICs and PICs
- Chapter 21 - PICs & Arduinos
- Chapter 22 - 3D printing
- Chapter 23 – Computers & model railways
- Chapter 24 – Assembling a tool kit
- Chapter 25 – Soldering
- Chapter 26 – Using test equipment
- Chapter 27 – Pocket Money Projects
- Chapter 28 - Abbreviations & Acronyms
- Appendix – The Model Electronic Railway Group

EzyBus

An Arduino-based layout control system

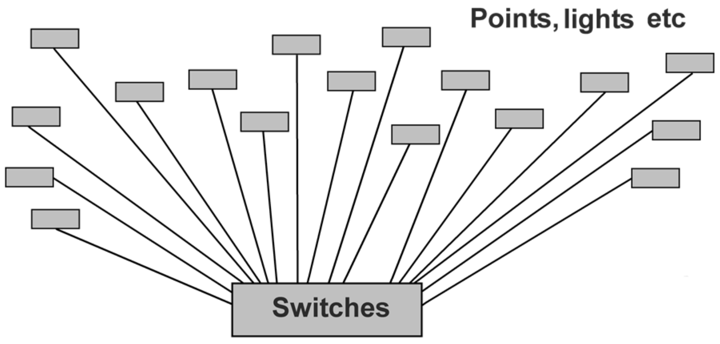
Like CBUS, this system uses a two-wire cable to connect a collection of input and output modules together.



This system that carries out all basic layout control functions (points, signals, gates, animations, lights, feedback, etc.) - and can be set up quickly and run without the need for a computer.
The system can handle up to 128 inputs (switches, pushbuttons, train detectors, etc.) and provides control of up to 128 servos and/or digital outputs.
It uses only three different types of module and avoids tiny surface mount components.
The points, etc. are set up from an LCD screen and five buttons, using on-screen menu prompts.
It also has helpful built-in diagnostics and reporting facilities.

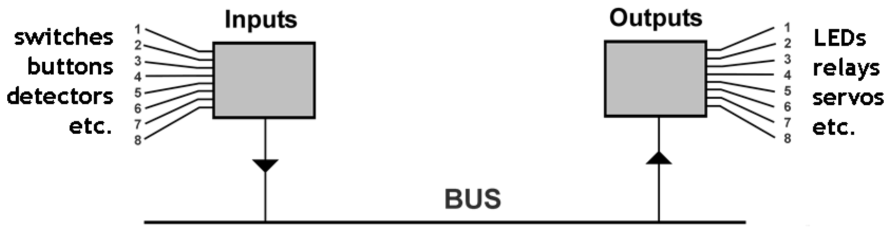
The 'old' way

For many years, hobbyists ran separate wires between every switch on their control panel to every point or light on their layout.
The larger the layout, the more wires to confuse, to break, to have bad joints, etc.
If your layout consists of a number of baseboards (e.g. in a portable layout for exhibitions), you have to use large multi-pin plugs and sockets between each baseboard and suffer potential bad connections.

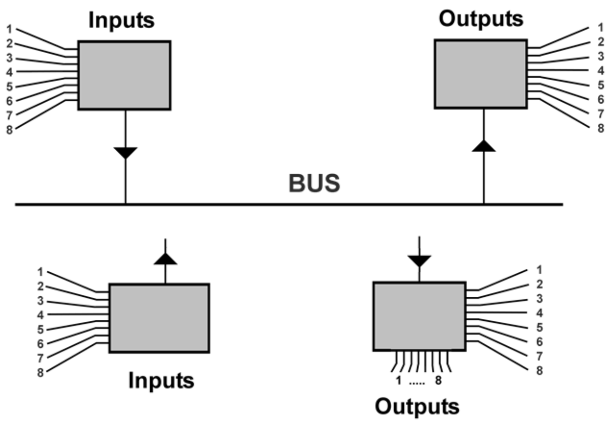


The 'bus' way

Instead of having separate wires all over the place, you run a couple of wires (the *'bus'*) round your layout and connect a collection of modules on to it. You can install the module with switches in your control panel and a selection of output modules round the layout.

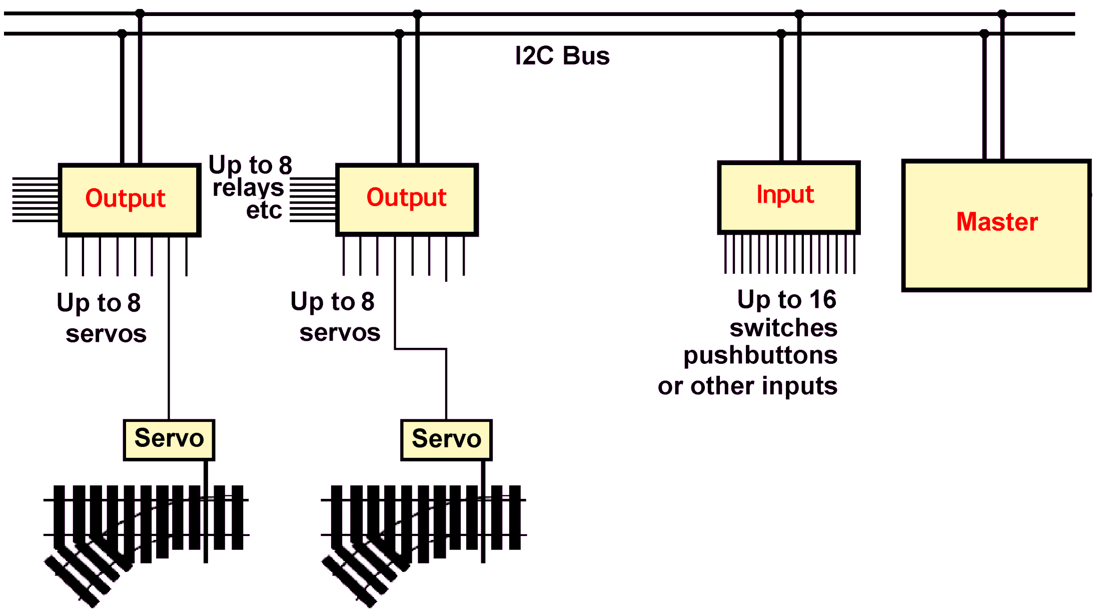


The module with the switches sends a message on the bus and one of the output modules will act on it (e.g. to switch on a light or move point). That way, you can match switches to outputs.



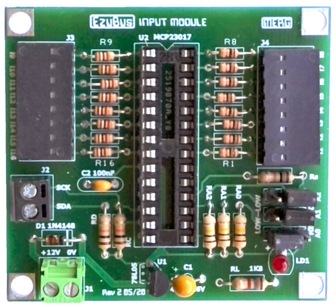
Now, if you want to extend the system, you simply add more input and output modules to the existing bus wires – no extra wiring required.

Here is an example of EzyBus with a basic 16-input, 16-output system.



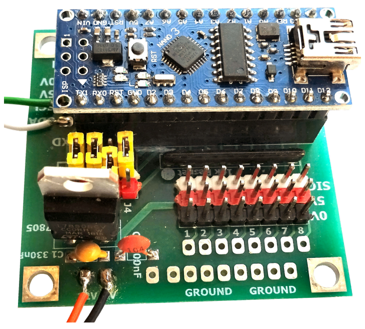
Larger implementations still use only the three same basic modules.

The input module



Each module can handle up to 16 inputs.
Up to 8 separate input modules can be added to a system, offering a total of 128 different inputs.
The inputs can be switches, push buttons, micro switches, reed switches, relay contacts, or train detectors, (any device that takes a pin down to 0V).

The output module



Each module has a row of 3-pin connectors for connecting up to 8 servos.
The servos can be used for points, gates, signals, animations, etc.
It also has another 8 digital connection pads whose outputs can either be made high or low.
The digital outputs can be used for other non-servo point controllers, frog switching or for control of lights, sounds, relays, motors, etc.

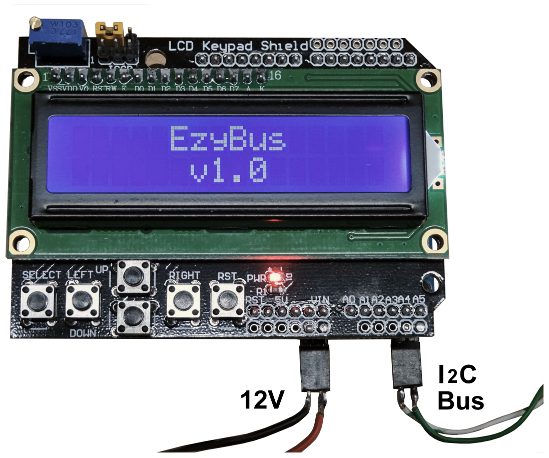
You can have any combination of servos and digital outputs. You could use a module as all servos, all digital outputs, half and half, or any combination.

The master controller

The output modules act as *'slaves'*; they only do what they are told, with the real system knowledge stored in the master module.
The pins of output modules are programmed from the menu of the master controller, which has an LCD screen and a number of push buttons.
The LCD menu screen and set-up buttons are only used to configure a new addition, or to alter the settings on a particular output pin.
Once set up, the master controller remains connected to handle all activities. You only need to return to the master controller if you want to add or change something.

All modules are powered from the same regulated 12V DC supply

Connecting it together



You only need one controller but you will usually have multiple input modules and output modules. This system is not based on CANbus. Instead, all modules are connected together by another two-wire bus (known as the I²C bus) that carries the messages and the timing, with all modules sharing the same 0V connection.

Apart from the 12V bus, a separate 2-wire bus runs round the layout and the various modules connect to bus along the way where required. The I²C bus pins are labelled SDA (Serial Data) and SCL (Serial Clock). SDA sends the data back and forth and the SCL keeps everything synchronised. The controller and the input and output modules all connect to the same SDA and SCL lines.

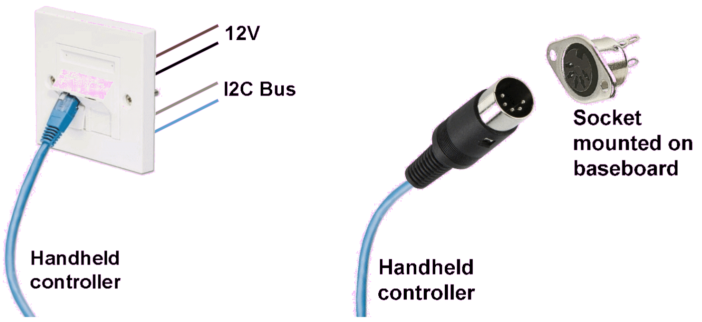
As with all communications lines, it pays to have the best wiring approach, to avoid external interference, capacitive effects and crosstalk between adjacent wires. These can degrade the signals on the bus, particularly on very large layouts.
Network cables (Cat5e) are an ideal choice, as they have four sets of twisted pairs within them. One twisted pair can be used with the SCL line and the 0V sharing the pair. Another pair can consist of the SDA line and either +5V or 0v.

You can run the I²C bus round the layout in any way that suits yourself. You can use soldered connections, screw terminals, tag strips, or whatever.
You can have a single run of the bus, with modules leading off from them, or you can zig-zag the bus, running from module to module.

If possible, avoid running the I²C bus too close to other wiring on your layout, to minimize the possibility of interference.

Making the master pluggable

When setting servos on points that are scattered throughout a large layout, having the master permanently screwed inside the control panel is not advisable.
It would be difficult to observe the movement of the point blades from a distance.
If the master is fitted with a plug (it only needs four connections – power and bus) it can be plugged into various sockets placed round the layout.
When laying the bus wire round the layout, fit sockets within easy viewing distance of the points (e.g. one for the fiddle yard, one for the station throat, etc.).



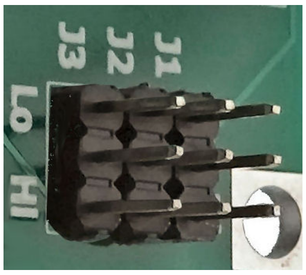
These examples show a couple of possible plug and socket arrangements. The left illustration is of a network type known as RJ45. It has eight connections, although you only need to use four. You can wire it any way you like (as long as you use the same colour scheme for all sockets). The RJ45 connector crimps the wires. If you prefer soldered connections, the other illustration shows a 5-pin DIN plug and socket.

How it works

The input modules use the MCP23017 I/O expander chip, while the output modules use programmed Arduino Nanos. The master controller uses a programmed Arduino Uno.
The master controller regularly checks the status of all the input modules.
The input modules store the current voltage level on all the pins and report their status back to the master controller when requested.
If the controller detects a change from an input module, it knows which module reported the change and which pin has altered its state (i.e. gone from high to low or vice versa). The controller then sends a command to the appropriate output module to rotate a servo, to change the status of one of its digital outputs, or both.
If a pin is configured as a servo, its corresponding digital pin is also switched when the servo is moved.. This is an option for those who have isolated frogs and want frog switching.
All current settings are held in the controller's EEPROM, so the settings remain stored even after the power is switched off.

Setting up modules

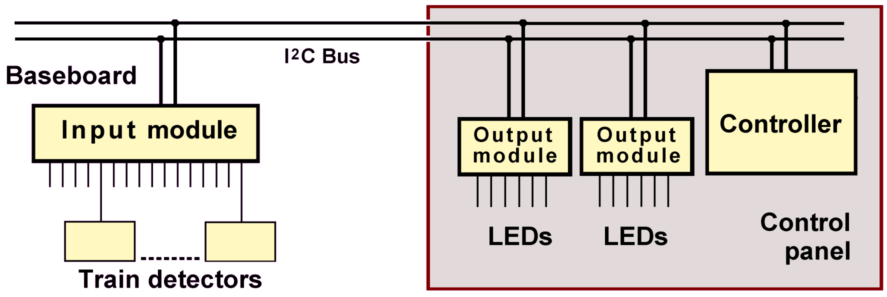
Each input module is allocated a different address (so that the master module can uniquely identify each of them).
Similarly, each output module is allocated a different address (so that the master module can send commands to specific modules).
This is done using jumpers across configuration pins.
Since the input module handles up to 16 inputs and output modules only handle up to 8 outputs, you have to match the switches to the outputs.
Each input module draws under 5ma and each output module draws under 50mA.



Train detector feedback

The system was initially intended for controlling points, LEDs, relays, etc. located on the baseboard.

However, the I²C bus is bi-directional (it passes data in either direction).



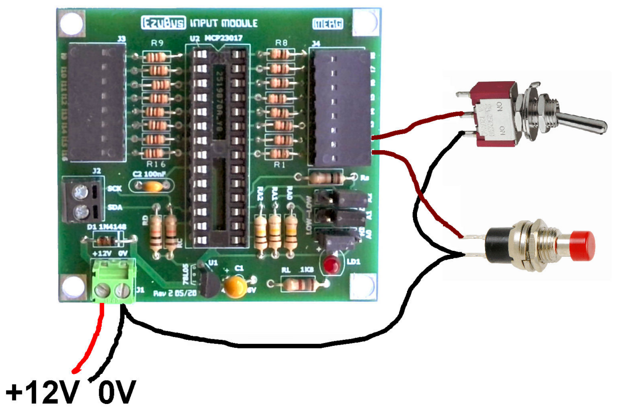
This diagram shows the input modules being located on the baseboard and being activated by TOTIs (train detectors). The output modules are located inside the control panel and illuminate LEDs on the face of the control panel.
This provides a track occupancy system over the two wires.

You can provide both facilities at the same time, sharing the same bus.
You can mix and match modules.

- Place some input units in a control panel to control points, etc.
- Place some output modules on the baseboard to operate the points
- Place other input units under the baseboard to feed back track detection information to the control panel.
- Place other output units in the control panel to illuminate LEDs indicating occupied sections of track.

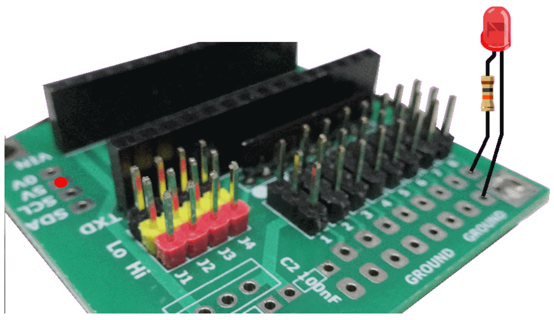
Connecting switches / pushbuttons

While connected to the I²C bus, the input module's input pads are held internally at +5V. An input can be brought down to 0V by a switch (e.g. for points or signals) or by a pushbutton (e.g. for an uncoupler).
This image shows a switch connected between input 10 and 0V, and a pushbutton connected between input 13 and 0V.
Alternative switching inputs include reed switches, microswitches, etc.



Connecting LEDs

There are two rows of pads on the output module, next to the servo connections.
The inner row is used for the digital outputs, while each pad in the outer row is at 0V.
The red dot shows where a +5V supply is available.



A LED plus its dropper resistor can be connected between one of the numbered digital pads and either the 0V or +5V pads (make sure you wire the LED with the correct polarity). If you are connecting one leg of the LED to 0V, you can connect the LED anywhere on the control panel etc., with a single wire from the digital pad (as the other leg of the LED can connect to the 0V line anywhere on the panel or layout.

The choice of connecting to +5V or 0V depends on when you want the LED to illuminate.

A LED lights when the digital pad goes to 0V, if the dropper is connected to +5V.

A LED lights when the digital pad goes to +5V, if the dropper is connected to 0V.

The recommended maximum current available from any digital output is 20mA for continuous usage.

Connecting servos

Servos have three wires, one each for the 0V, +5V and the controlling signal.
The colour of the wires can vary on different models. Usually, they are either black, red, and white, or they are brown, red, and orange/yellow.
When inserting a servo's plug into the output module, the signal wire (e.g. the white or orange wire) should face inwards towards the Nano.

Connecting relays

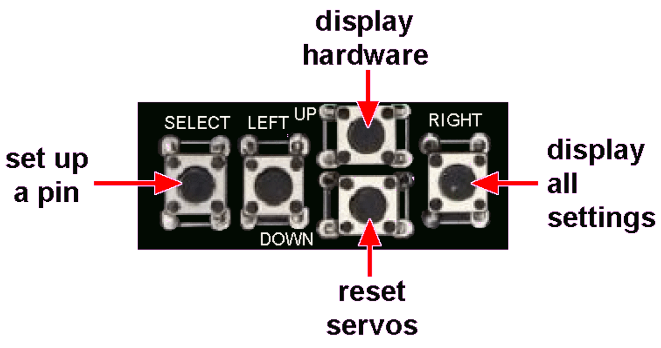
An output module's pins cannot handle high currents or high voltages, so a relay can be used to interface the output module to high current or high voltage devices.
The output module's digital outputs can operate 12V relay modules (the type that use an opto-coupler on their input).
The relays can be used for

- Frog switching
- Controlling heavy current motors
- Operating tortoise motors

Relays allow EzyBus to control Tortoise motors as well as servo motors, if the relay is a DPDT (double-pole double-throw) type.

Configuring outputs

All configuration activities are handled using the buttons on the controller.
The keyboard/LCD shield uses five push buttons.

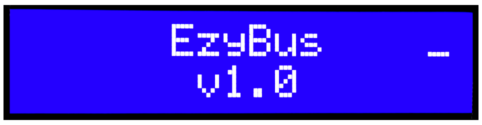


The buttons are used for different tasks at different times.

These are the main (top-level) options:

Up button	Used to display what input modules and output modules are currently connected to your system.
Down button	Used to set all the servos to mid-position.
Select button	Used to configure individual pins to be points/signals/digital, including rotation values and speed.
Right Button	Used to show how all the output modules are currently set up (point, signal or digital output). *This option needs the controller to be connected to your computer's USB port, as the results are displayed on the computer screen.*

This is the default screen:
The system continually checks input modules and controls output modules, until the Select button is pressed. This starts the configuration process.



The configuration process is

1. Select which output module you want to configure.
2. Select which output pin you want to configure.
3. Select whether to make that pin operate:
   - A point (including servos used for uncouplers, gates, doors, animations, etc.)
   - A semaphore signal with bounce.
   - A digital output (e.g. for LEDs, or triggers for sounds, relays, etc.)

If you choose to configure the pin as a point controller, you decide:

- how far the servo arm should rotate in one direction.
- How far the servos should rotate in the other direction
- Having set these 'endpoints', you decide on the speed of the rotation.

A demonstration of this can be found at :
https://youtu.be/z4dnNLFcSbo

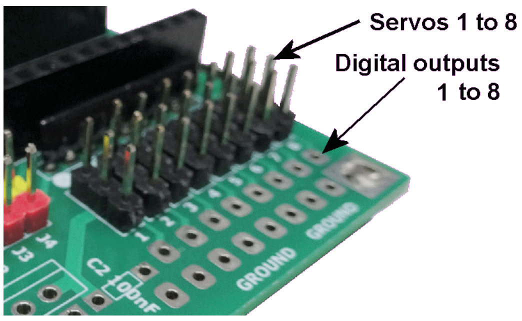
If you choose to configure the pin for a semaphore signal, you set the endpoints and the pin will will automatically add the semaphore 'bounce' movement.

If you choose the I/O option, you are setting up a pin as a digital output.
You are not asked to enter angle or speed settings.

Notes

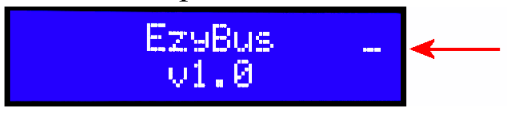
- Choosing the Servo option configures the signals that will appear on the 3-pin servo connections.
- Choosing the I/O option configures a digital pad as a switchable output. The pads line up next to the servo connectors.

If you configure a module's pin 4, for example, as a point then digital pin 4 will change state each time the servo rotates from one position to the other. This output can be used to drive a relay for frog switching if your frog is isolated from the rest of the trackwork.



Diagnostics

It is helpful if the user is provided with system information, either on demand or automatically as the system is used.
If communication has been correctly established with one or more input modules, there should be a constant flicker of an underscore character in the top right digit of the controller's LCD display.

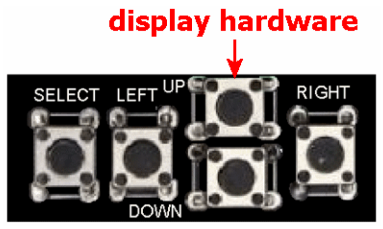


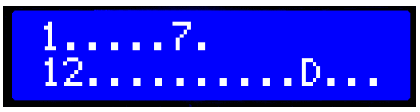
```
EzyBus    _
 v1.0
```

That way, you can tell that if the bus is passing messages between the controller and the input modules. To test a particular input module, have it as the only one on the bus and check whether you see the test flicker.

Reporting on attached modules

If you press the *'Up'* button on the controller, a map of all attached modules is displayed.
The top row shows what input modules are connected and the bottom row shows all output modules on the bus.



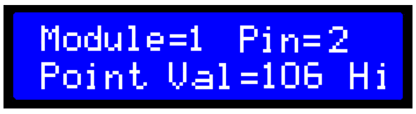


```
1.....7.
12..........D...
```

In this example, input modules with jumpers set to 1 and 7 are attached to the bus.
Output modules 1, 2 and 13 are also shown as being attached to the bus.
Since there are only 10 numeric characters available on the LCD screen, the display uses 'hex' characters for numbers from 11 to 16. *'A'* means 10, *'B'* means 11 and so on. In the example, *'D'* shows that output module 13 is attached to the bus. Dots indicate modules that are allocated numbers but are not attached.

Reporting on controller activities

When you throw a switch, push a button, etc., the resultant action taken by the master controller is displayed on its screen for a short time.

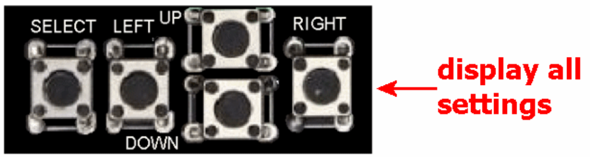


```
Module=1  Pin=2
Point Val=106 Hi
```

In this example, the second servo on output module 1 has been set to a rotation value of 106 and its associated digital output pad has been set to +5V.
This screen output provides positive feedback to the user.
It can also be used to detect faulty input modules, modules set to the wrong addresses, etc.
The module/pin info should match the switch that was thrown on a particular input module.
For example, lowering the voltage on the second pad of input module 1 should display *"module 1 pin 2"*.
Lowering the voltage on the eleventh pin on input module 1 should display *"module 2 pin 3"* .... and so on.

Documenting settings

Every time you configure a module/pin, the LCD displays the settings you have just made.
These settings can be noted and entered into a list for future reference.



Another option, if you have a computer, is to press the 'Right' button to display all the controllers settings.

Here is an example:

```
 Module 1
   Pin 1   Point   Left:  75  Right:  95   Speed:33
   Pin 2   Signal  Up:    93  Down:   86
   Pin 3   Point   Left:  90  Right: 102   Speed:65
   Pin 4   Digital output
   Pin 5   Point   Left:  71  Right:  90   Speed:65
   Pin 6   Point   Left: 143  Right:  42   Speed:13
   Pin 7   Signal  Up:   101  Down:   90
   Pin 8   Digital output
----------------------------------------------------------------
```

This will be repeated for each output module you have connected.

Multiple activities

You can make one change of input on a single result in two or more activities on the layout.

- If you connect a switch to two input pins, a throw of a single switch could result in moving two servos (useful for crossovers or passing loops).
- If you connect one of pins of an output module to the trigger input of a sequencer module (sold by MERG as PMP26), throwing a single switch can initiate a sequence of activities such as animations, lights and sound effects.
- If you connect the output of a train detector to one of the pins of an input module, you can have a train arriving at a station initiate a useful sequence (e.g. station master comes out of office, station lights go on, station announcements, etc.). The train detector informs the master controller of the train's arrival and the controller changes the output of one of the pin's on an output module, which triggers the sequencer.

Reliability

All input modules are identical to each other; the only difference is the jumper settings you used to provide it with its unique address. That means that if one ever breaks down, it can be replaced with another one, with no need to reprogram anything; just make sure its jumpers are set to the same address as the faulty one.
All output modules are also identical to each other and can be replaced in the same way.
For clubs exhibiting at shows, having a spare input module and a spare output module means that a layout can swiftly be brought back into full operation in the event of a module failure.

Notes

- The system is designed for small to medium sized layouts.
- It does not support computer automation using applications such as JMRI or RocRail
- It operates at 100kbps compared to 125kbps for CBUS.