SignalBox

Table of Contents

Summary

SignalBox is software for controlling model railway signals, points and other accessories. It doesn't attempt to control the running of trains at all.

It use the same hardware as the EzyBus system: An Arduino Uno with LCD shield, Arduino Nanos to provide outputs (to Servos, LEDs etc) and MCP23017 input expanders to read switches and buttons, all connected by an I^2C bus. Although it uses exactly the same hardware as EzyBus, all the software is completely new.

It is positioned as a half-way house between the EzyBus system, and a CBUS one.

There is no need for a computer, all configuration can be done using the LCD panel. A computer can be connected to save and/or restore the configuration if desired.

Features

Features of the system are (in brief):

- Multiple input types: toggle switches (SPST) and intermittent (non-latching) push buttons.
- Multiple inputs can be operated simultaneously.
- Multiple outputs can be operated by a single input - eg crossovers.
- Multiple inputs can operate the same output(s) - so multiple mimic-panels can control the same or similar outputs.
- Multiple output types, Servos, and various digital IOs including PWM.
- Servos, with configurable sweep and speed, with attached digital IO that switches at the mid-point of the servo travel.
- Signal "bounce" and "stutter" for semaphore signals.
- Variable-intensity LEDS (including fading) using PWM.
- Some other output types, eg flashing (varying speeds) or flickering LEDs.
- Four-aspect signals.
- Three-aspect UK road traffic lights.
- Three-aspect non-UK road traffic lights.
- Random outputs that go on or off at unpredictable intervals.
- Where multiple outputs are controlled by an input, the ability to delay the outputs, so for example crossing gates can operate one after the other.
- Automatic reset of inputs after a time delay (for signals that go red/danger with input from a TOTI but then revert to green sometime later).
- Interlocks that prevent certain outputs operating if other outputs are set incorrectly.

Hardware

The system uses the EzyBus hardware, either the MERG kits (22 & 23) or self-assembled using the PCBs (kits 922 and 923) built exactly as described in the EzyBus manual.

There are two versions of the PCB kits, the older ones use an Arduino Nano, the new ones an ATmega328 chip (the same as found on a Nano) but don’t include a serial interface so programming is via a custom programmer or with a USB-TTL serial cable or adapter and a few external components.

The software must be programmed onto the Uno (via the USB) and the Nanos (which is more difficult with the new kits). With a serial cable you can use either the Arduino IDE or the avrdude program directly.

Note that EzyBus numbers its input and output modules and their pins from 1 to 8 or 1 to 16 (0x10 hexadecimal). This software numbers from zero to 7 or 15 (0xF hexadecimal).

Overview

The system consists of an Arduino Uno controller with LCD shield, up to eight input modules (MCP23017 chips) and up to 32 output modules (Arduino Nanos) all connected by an I^{2}C 2-wire bus.

The inputs are not directly tied to the corresponding output. Each input can be mapped to operate any output. Indeed, each input can operate up to six outputs simultaneously or in sequence.

The outputs generally drive servos with an associated digital output, or a pair of digital outputs (often LEDs). For servos, the range of rotation (or reverse rotaton) and speed of rotation can be specified. The digital outputs use PWM so can arrange to adjust the intensity and/or fade the outputs.

Interlocks can be specified between outputs to prevent certain combinations, for example prevent a signal being cleared if the points are set incorrectly (and vice-versa).

Inputs

Each input module provides 16 input lines which are normally held high. Pulling the input low activates the input.

Each input can be configured to operate as one of four types:

Toggle	An on-off toggle switch (eg SPST). Switched high sets the corresponding output(s) “Hi”, switched low sets the output(s) “Lo”.
On_Off	A momentary push-button. First activation sets the output(s) “Hi”, another activation sets the output(s) “Lo”.
On	A momentary push-button which always sets the corresponding output(s) “Hi”.
Off	A momentary push-button which always sets the corresponding output(s) “Lo”.

Each input can be configured to operate up to 6 outputs. Several different inputs can operate the same (or some of the same) outputs of another input, providing many-many configurations.

Outputs

Each output module has 8 outputs each of which can be configured as one of nine types:

Servo	Output connected to a servo which can rotate between 0-180 degrees. The endpoints can be set, as can the speed of rotation.
Signal	A servo as above which additionally emulates bounce and stutter typical of a semaphore signal.
LED	A LED or other digital output device, typically a (2-aspect) colour signal or other indicator light. The intensity of each output, and the speed with which the output fades from one setting to the other can be configured.
LED_4	A 4-aspect colour signal. Configured using two adjacent outputs, again with intensity and fade configuration options.
RoadUK	A three-aspect UK road traffic light, Red, Red & Amber, Green, Amber and back to Red again in sequence. Also configured using two adjacent outputs. Up to four “RoadUK” outputs can be configured on a single output module, and will operate in turn.
RoadRW	A three-aspect road traffic light as used in the rest of the world (non-UK), Red, Green, Amber and back to Red again in sequence. Also configured using two adjacent outputs. Up to four “RoadRW” outputs can be configured on a single output module, and will operate in turn.
Flash	A pair of LEDs that flash for a while whilst changing state – one ends up on, the other off depending whether it’s being set “Hi” or “Lo”.
Blink	As for Flash above, except that the output always ends with both outputs off.
Random	A pair of LEDs that (pseudo) randomly go on and off at unpredictable intervals.

The output modules have three-pin headers for connection to servos (Signal, Vcc and Ground). There’s an additional digital output (“pad”) which goes high when the servo is “Hi”. The change-of-state of this output occurs as the servo moves through the mid-point of its travel.

When connecting digital outputs (LEDs) the main output is on the servo “Signal” pin, with a complimentary output an the associated “pad” output.

Note that any digitally-operated output (not just LEDs) can be used for the non-servo options. If a simple on-off signal is required then the speed can be set to ‘F’ for an instant (no fade) switch and the intensity to 255 so the PWM signal is “always on”.

Setup

Connect all the devices (Uno, input and output modules) to your I^{2}C bus and suitable power supplies. Switch everything on.

The Uno will display its splash panel on the LCD.

```
SignalBox v3.3.2
Setup     Apr 21
```

After a few seconds, a prompt to calibrate the LCD buttons appears. Press the indicated buttons in turn as prompted.

```
Calibrate button
Press select
```

This step is skipped on subsequent boots. To force re-calibration of the buttons, should it be needed, press and hold any button whilst the Uno is re-booted. When the “Calibrate button” message appears, release the button, then press the indicated buttons when prompted as above.

Now a display indicating the attached input modules is shown. The top line shows the numbers of the input modules detected (numbered 0 to 7) or a dot for missing modules. In the example, input modules 4 & 5 are present.

```
Nodes   ....45..
Input
```

If a ‘#’ character should appear here, it indicates an I^{2}C LCD display was detected and that will be used to duplicate all messages that appear on the regular LCD display.

After a few seconds, the output modules detected (numbered in hexadecimal from 0 to F) are shown. If there are more output modules, they’re identified on the second row as ‘G’ to ‘V’. In the example, output modules 2, 3 and 6 have been detected.

```
..23..6.........
................
```

When complete, the standard panel is displayed showing the time the software has been running which gives confirmation that the software is running OK.

```
SignalBox v3.3.2
00:12     Apr 21
```

This is the normal running state for the software. After this initial setup, subsequent boots of the software will progress directly to this panel. No interaction is required.

Configuration

To enter configuration mode, press any of the LCD buttons and the configuration menus are shown.

Menu buttons

The menus are navigated using the LCD buttons.

Up	Moves to the next menu at the current level, or increases the value of the current field.
Down	Moves to the previous menu at the current level, or decreases the value of the current field.
Right	Selects the current item and moves to the sub-menu or next field.
Left	Exits the current item and moves to the next higher field or menu. If changes have been made, then a warning message is shown to confirm that they should be discarded.
Select	Depending on the context, one of three things may happen:

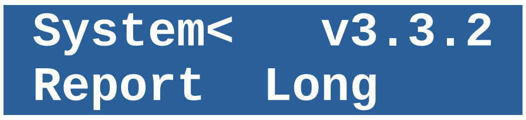
1. The current changes are saved. A warning message is displayed to confirm the changes made. If no changes have been made, operates as the “Left” button.
2. The current item is toggled between it’s normal state and an alternate one (eg output is disabled/enabled).
3. The current output is exercised so you can identify it, or check the changes you’ve made are those desired.

The currently selected menu or item is marked at all times with chevrons – the ‘>’ and ‘<’ characters.

Menu structure

The top-level menu has 6 options.

The example image shows the System menu selected with its current sub-menu (Report) also displayed.

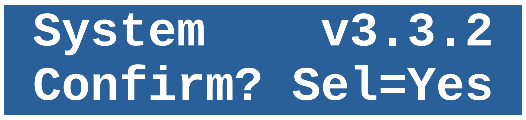


```
System<   v3.3.2
Report  Long
```

System	Configure system-wide options.
Input	Configure the input modules.
Output	Configure the output modules.
Lock	Configure interlocks between outputs.
Export	Export the current configuration.
Import	Import a previously saved configuration.

Use the up/down buttons to select the desired menu and the right-button to enter its sub-menu.

When returning to this menu level (if changes have been made) using the left (to cancel) or select (to confirm) buttons, a warning message is displayed asking for confirmation.



```
System    v3.3.2
Confirm? Sel=Yes
```

When the warning message appears, use the select button to accept, or any other button to refuse, the cancellation or confirmation of the changes as desired.

System menu

The system menu has 4 sub-menus:

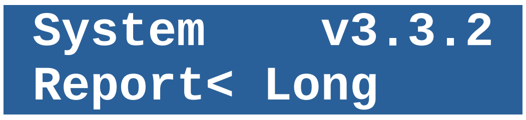
Report	Sets the reporting level.
Nodes	Displays the detected input and output modules' node IDs (as shown during startup).
Ident	Identifies all the outputs in turn by exercising them (move servos or blink LEDs).
Debug	Sets the debugging level.

Use the up/down buttons to select the desired sub-menu, and then right-arrow to enter the sub menu configuration.

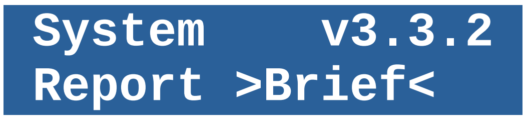
Use the left or select buttons to cancel or confirm any changes and exit.

Report sub-menu

This allows the reporting level to be set. When inputs are triggered and outputs actioned, this level dictates what messages (if any) are shown on the LCD panel.



```
System    v3.3.2
Report< Long
```



```
System    v3.3.2
Report >Brief<
```

Use the right-button and then adjust the level using the up/down buttons.

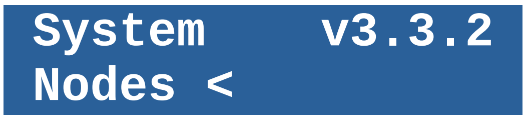
None	No reports are shown.
Brief	Reports are shown for a short period (about 2 seconds).
Long	Reports are shown for a long period (about 4 seconds).
Pause	A report is shown and the operator must acknowledge it before any more processing can continue by pressing one of the LCD buttons.

See the section on Report output to see what the output will look like.

Nodes sub-menu

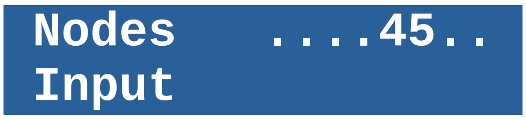
Shows the connected input and output module node numbers.

Use the right-button to show the node Ids. These will be shown exactly as for the start-up.



```
System    v3.3.2
Nodes <
```

As during start-up, first the input nodes present are shown, then the output nodes.



```
Nodes    ....45..
Input
```

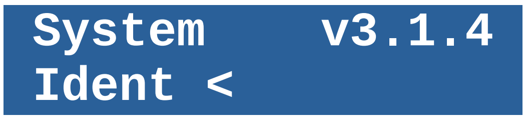
Press any button whilst the nodes are being shown to pause the display until the button is released.

No changes can be made.

Ident sub-menu

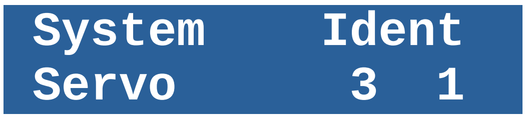
Cycles through all the connected output nodes, exercising each one in turn.

Use the right-button to start the Ident process.



```
System    v3.1.4
Ident <
```

As each one is exercised, it is identified on the LCD screen. The example shows the servo on output module 3, pin 1 is being exercised.

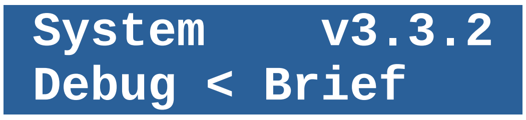


```
System    Ident
Servo       3  1
```

If the process goes too quickly, use the left-button to go back one pin. Conversely, the right-button will move ahead to the next pin. Similarly, down-button will re-identify the previous module, up-button will move ahead to the next module. The select button will abort the ident process.

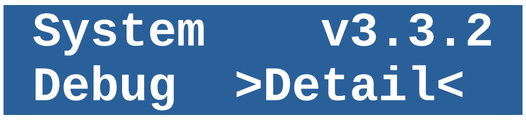
Debug sub-menu

This controls the verbosity of debug messages output to the serial port of both the Uno control module and the output modules.



```
System    v3.3.2
Debug < Brief
```

Use the right-button and then adjust the level using the up/down buttons.



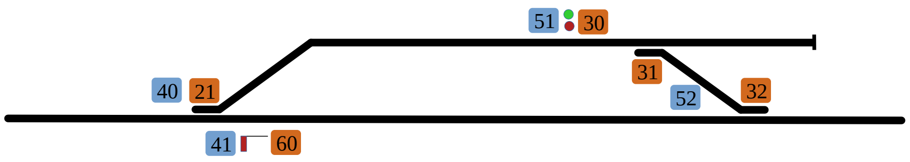
```
System    v3.3.2
Debug  >Detail<
```

None	No debug messages are output.
Errors	Only error messages (when something goes wrong) are output.
Brief	Brief messages describing major events are output.
Detail	More detailed messages are output for more events.
Full	Debug messages are output for all events.

Normally this option should be set to “None” or “Errors” as the serial-IO will severely impact the performance of the system – servos will lurch, LEDs will flash rather than fading etc.

Demonstration layout

For the purposes of the documentation, refer to the following mythical track layout:



Input	40	Point push-button
	41	Semaphore signal push button
Output	21	Point servo
	60	Semaphore signal servo

Input	51	Colour signal push-button
	52	Cross-over push-button
Output	30	2-Aspect colour signal
	31	Cross-over point servo
	32	Cross-over point servo

Outputs not shown	34	4-aspect colour signal
	36	Traffic lights
	22	Flashing lights
	23	Blinking lights
	24	Random lights

Note: Although shown on the same diagram, in reality the input switches (in blue) would be on a control panel, and the outputs (in brown) on the layout itself. Often there would also be inputs on the layout (eg track occupancy detectors etc) and output indicators on the control panel but all this is omitted for clarity.

Output menu

Configure the outputs. The second row shows a summary of the selected output's current configuration. These are its type, the "Lo" and "Hi" settings and the speed, all in hexadecimal. The example shows that output module 2, pin 1 is a servo.

```
Output<    2  1
Servo   4F 3E D
```

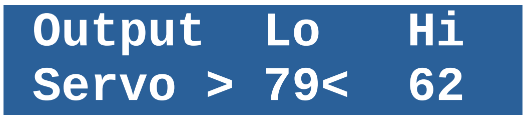
Use right-button and then select the module ID to configure. Use the right-button again and then select the individual pin to configure. Press the select button here to exercise the output to confirm it's the desired one.

```
Output     2 >1<
Servo   4F 3E D
```

Use the right-button again to move to the type field. Use the up/down buttons to select the desired output type.

```
Output     2  1
Servo  <4F 3E D
```

Output Servo sub-menu

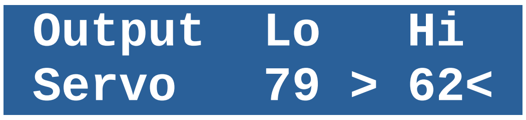


```
Output  Lo   Hi
Servo > 79<  62
```

Use the right-button to configure a servo's movement positions and speed.

The servo's Lo and Hi positions are now shown as decimals between 0 and 180 (degrees).

The servo will move in sympathy to the changes made. To adjust more quickly, press and hold the up/down buttons. Use the select button to test the servo, it will move between the specified Lo and Hi positions.

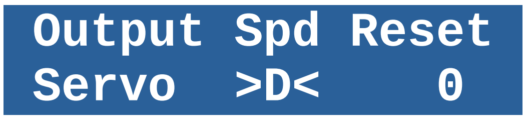


```
Output  Lo   Hi
Servo   79 > 62<
```

Use the right button to move to the "Hi" field. The servo will move to the Hi position at the same time.

Again, the servo will move in sympathy to the changes made. Again the select button will test the servo.

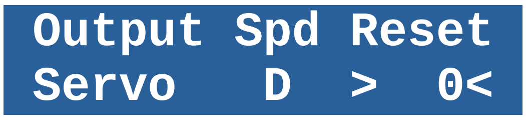
There's no equirement for the "Lo" value to be less than the "Hi" value – the servo will just operate in the opposite direction.



```
Output Spd Reset
Servo  >D<    0
```

Use the right-button to configure the servo's speed.

Speed '0' is the slowest, about 7 seconds, 'F' the fastest (as fast as the servo can go).



```
Output Spd Reset
Servo   D  >  0<
```

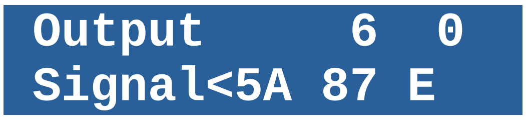
Use the right-button to configure the reset interval. This is the time (in seconds) after which the servo will automatically return to its Lo state. Zero means don't auto-reset. To adjust more quickly, press and hold the up/down buttons.

Servos (and signals) don't often require a 'Reset' value, it's more common for the other output types.

Use the left-button to return through the fields to the "Servo" field.

Use the left or select button to cancel or confirm the changes.

Signal sub-menu



```
Output      6  0
Signal<5A 87 E
```

Signals are configured in exactly the same manner as servos above.

All the same features are available.

Signal outputs perform just like servo outputs, but may (randomly) stutter when going "Hi", and/or bounce when going "Lo" to resemble real semaphore signals.

LED sub menu

A summary of the LED's configuration (in hexadecimal) is shown.

```
Output      3  0
LED   <3F 3E D
```

Use the right-button and the LED's Lo and Hi intensities are now shown as decimals between 0 and 255.

```
Output  Lo   Hi
LED   > 63<  62
```

The LED's intensity will vary in sympathy to the changes made.

Use the right button to move to the "Hi" field. The LED will go out and the complimentary LED will illuminate (at the Hi intensity).

```
Output  Lo   Hi
LED     63 > 62<
```

Again, the LED will move in sympathy to the changes made. Again the select button will test the LED.

Use the right-button to configure the speed at which the LEDs fade in and out. Zero is slowest and takes about 3 seconds, 'F' is fastest, switching immediately.

```
Output Spd Reset
LED    >D<    0
```

Use the right-button to configure the reset interval. This is the time (in seconds) after which the LED will automatically return to its Lo state. Zero means don't auto-reset.

```
Output Spd Reset
LED     D  >  0<
```

Use the left-button to return through the fields to the "LED" field.

Use the left or select button to cancel or confirm the changes.

Notes:

1. If wiring 2-aspect signals, it's normal (though not required) for the red LED to be wired to the "Hi" output (where a servo would be connected) and the green LED to the alternate pad output. Thus "setting" the signal Hi (or "on") will illuminate the red LED, and "releasing" the signal Lo (or "off") will illuminate the green LED.
2. If using the output for non-LED purposes, often the intensity is set to 255 (always on, no PWM) and the speed to 'F' (immediate, no fade).

LED_4 sub menu

```
Output      3  4
LED_4 <3F 3E D 5
```

LED_4 outputs are configured in exactly the same manner as LEDs above.

All the same features are available.

However, LED_4 outputs are tied to the previous (next lower number) pin's output (on the same output module) and operate in tandem.

To operate correctly the previous pin must be configured as a LED with a red LED on the normal (servo) pin, and an amber LED on the alternate pad. The LED_4 must have an amber LED on its normal (servo) pin, and a green LED on its alternate pad.

Setting the LED_4 Hi will illuminate the red LED, with all the others extinguished. Subsequent settings of LED_4 Lo will step through the appropriate combinations: single amber, double amber until finally the green LED alone is illuminated.

If a reset value is specified, then the LEDs will change from red, through the ambers and finish as green as the reset interval expires. In the example, the colours will change after 5 seconds.

RoadUK and RoadRW sub menu

```
Output      3  6
RoadUK<3F 3E D
```

Road outputs are designed to reflect UK (RoadUK) road traffic signals or non-UK (RoadRW – Road for Rest of the World) road traffic signals.

They are configured in exactly the same manner as LED_4s above. All the same features are available.

Like LED_4 outputs, Road outputs are tied to the previous (next lower number) pin's output (on the same output module) and operate in tandem.

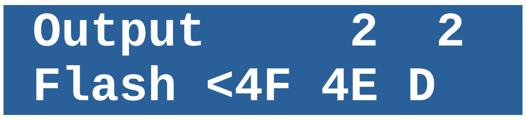
To operate correctly the previous pin must be configured as a LED with a red LED on the normal (servo) pin, and no LED on the alternate pad. The Road must have an amber LED on its normal (servo) pin, and a green LED on its alternate pad.

Setting the Road Hi will illuminate the red LED, with all the others extinguished. Subsequent settings of Road Lo will step through the appropriate combinations: red & amber (UK only), green, amber and back to red.

If a reset value is set, the lights will operate indefinitely through all the states. Further, the reset value of the LED pin (if specified) will set the time that red & amber or just amber are displayed, the Road reset value will determine how long the red and green states persist.

Several Road sets can be configured using adjacent output pins. For a two-way set use (for example) pin0 as a LED, pin 1 as a RoadUK, pin 2 as a LED, pin 3 as a RoadUK, all with suitable reset values. Now the first pair of pins (0 & 1) will cycle through red, red & amber, green, amber and back to red. But next, the second set of pins (2 & 3) will cycle through their colours with the first pair remaining red. After this, the next pair of pins (4 & 5) – if configured as a LED/RoadUK pair – will cycle through their colours. When the last configured pair is finished, the cycling returns to the first pair (pins 0 & 1).

Flash sub menu



```
Output      2  2
Flash <4F 4E D
```

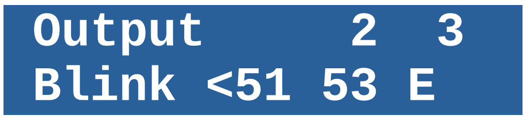
Flash outputs are configured in the same manner as LED outputs.

Two of the parameters have slightly different meanings.

The Speed parameter specifies how fast the output should flash. Speed ‘0’ is the slowest, taking about 4 seconds between flashes. The fastest speed is ‘E’, about a tenth of a second. Speed ‘F’ makes the LED flicker randomly. There is no fade-in or fade-out for Flash outputs.

The Reset parameter specifies how long the output should flash for (in seconds). After this period, the output remains steady at either Lo or Hi as appropriate. Zero means flash indefinitely – only stop when the output is set to Lo.

Blink sub menu

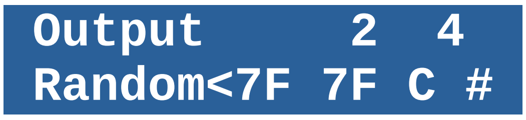


```
Output      2  3
Blink <51 53 E
```

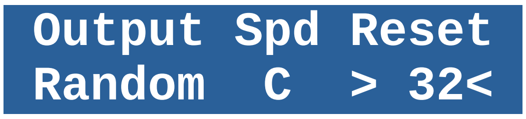
Blink is almost identical to Flash and is configured in the same manner.

The difference is that Blink outputs (after their reset interval) finish with both outputs off . As for “Flash”, the reset value specifies how long the output should flash for (in seconds) with zero again indicating flash indefinitely.

Random sub menu



```
Output      2  4
Random<7F 7F C #
```



```
Output Spd Reset
Random  C  > 32<
```

Random is used to drive a pair of outputs pseodo-randomly.

The intensity of the output is set as for LED outputs, as is the fade speed.

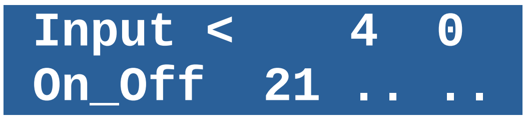
The reset interval dictates the (approximate) interval (in seconds) before the outputs may (or may not) change state. After a random interval between half the reset value (32 seconds in the example) and one-and-a-half times the reset value (ie between 16 and 48 seconds in the example), both the outputs may (or may not) change state. The Hi output (the servo pin) has a 60% chance of being set on, else it will be set off. The Lo output (the alternate pad) has a 40% chance of being set on else it will be set off.

There will then be another interval somewhere between a half and one-and-a-half times the reset value before the LEDs may change again (with the same probabilities as above).

It’s possible for both outputs to be on at the same time, or indeed neither to be on.

Input menu

Configure the inputs. The second row shows a summary of the selected inputs's current configuration. The type of the input, and the output(s) which are operated by this input.

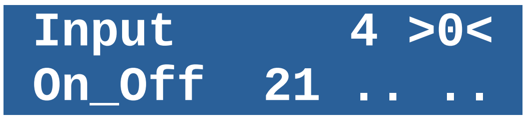


```
Input <    4  0
On_Off  21 .. ..
```

In the example, module 4, pin 0 is an "On_Off" input, operating output module 2, pin 1. The first three (of six) outputs are shown, with the double-dots indicating "disabled".

Refer to the layout diagram in the *Demonstration Layout* section earlier to help with following this description.

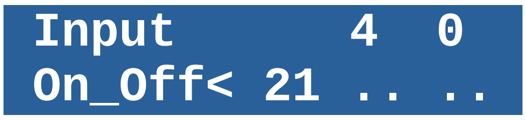
Use the right-button and then select the module ID to configure. Use the right-button again and then select the individual pin to configure.



```
Input      4 >0<
On_Off  21 .. ..
```

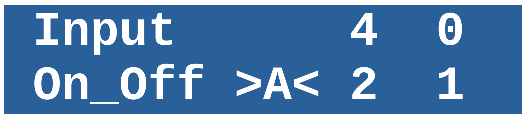
Pressing the select-button here will exercise all the outputs attached to this input for identification/check.

Use the right-button again to move to the type field. Use the up/down buttons to select the desired input type.



```
Input      4  0
On_Off< 21 .. ..
```

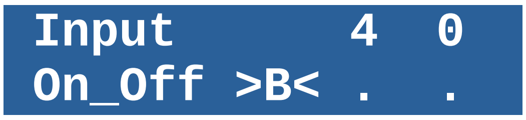
Use the right-button and the first (labelled 'A') output configuration is shown.



```
Input      4  0
On_Off >A< 2  1
```

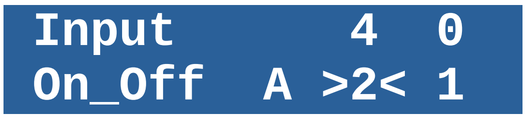
Pressing the select-button here will exercise the attached output to help identify/check you have the desired output.

Use the up/down buttons to choose the output to change, there can be up to six outputs labelled 'A' to 'F'. Disabled outputs are shown with two '.' characters.



```
Input      4  0
On_Off >B< .  .
```

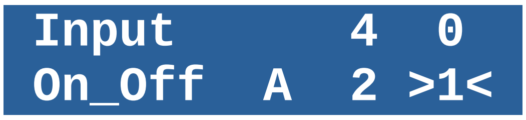
Use the right-button and then the up/down buttons to change the attached output node.



```
Input      4  0
On_Off  A >2< 1
```

Use the select-button to enable/disable the output.

Use the right-button and then the up/down buttons to change the output pin.



```
Input      4  0
On_Off  A  2 >1<
```

Pressing the select-button here will exercise the output for identification/check.

Use the left-button to return through the fields to the type ("On_Off") field.

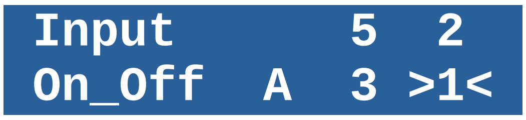
Use the left or select button to cancel or confirm the changes.

Input delay

To configure a delay between the operation of an input's outputs, put a special "delay" output between the two outputs concerned.

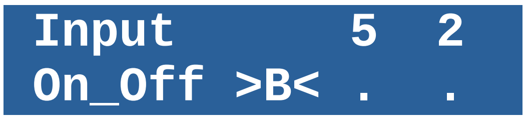
For example to put a delay between the outputs module 3, pin 1 and module 3, pin 2 for input module 5 pin 2:

Use the right-button to get to the output label field, and configure output "A".



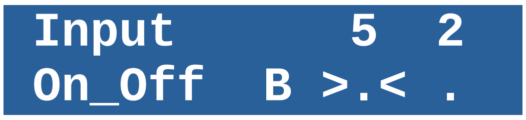
```
Input      5  2
On_Off  A  3 >1<
```

Return to the label field and use the up/down buttons to select output 'B'.



```
Input      5  2
On_Off >B< .  .
```

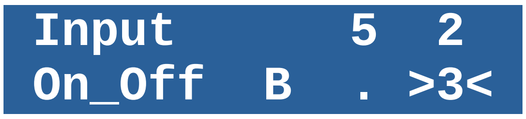
Move to the node field and use the select button if necessary to ensure the output is disabled.



```
Input      5  2
On_Off  B >.< .
```

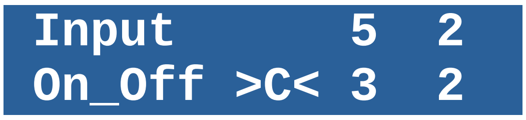
Use the right-button and then the up/down buttons to specify the delay required, from 1 to 7 seconds.

To remove a delay, set it to zero (a '.' will once more be shown).



```
Input      5  2
On_Off  B  . >3<
```

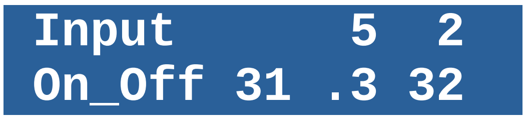
Proceed to output 'C' to configure the output to be actioned after the delay.



```
Input      5  2
On_Off >C< 3  2
```

Return to the type ("On/Off") field and a summary of the setting is shown.

Input module 5, pin 2, will operate output module 3, pin 1, pause 3 seconds and then operate output module 3, pin 2.



```
Input      5  2
On_Off 31 .3 32
```

Note: With multiple outputs, the outputs are operated in reverse order when being set Lo, so first output module 3, pin 2, then wait 3 seconds, then output module 3, pin 1.

The output types Flash & Blink don't honour any configured delays – they always start or stop immediately. This is to ensure that when used on control panels to indicate "operation in progress" they operate whilst the operation is actually in progress, and not before it starts or after it finishes.

Lock menu

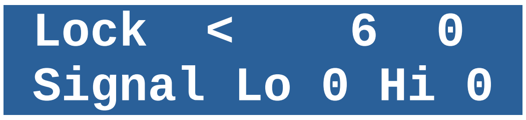
Interlocks can be configured between any pair of outputs. Each output can have up to eight locks, four against its Lo state, four against its Hi state.

Normally locks are created in pairs, one entry in each of the outputs.

Locks specify a state that is forbidden. For example a signal may not be Lo (Off) when a servo (point) is set against it, Hi for example.

The definition for the signal will be specified in its Lo set, against the point being Hi. The definition for the point will be in its Hi set, against the signal being Lo (Off).

Refer to the layout diagram in the *Demonstration Layout* section earlier to help with following this description. The example here sets an interlock between the point (output module 2, pin1) and the semaphore signal (output module 6, pin 0).

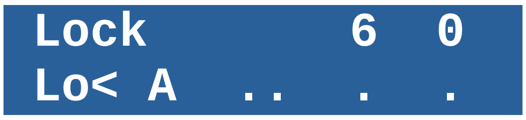


```
Lock  <    6  0
Signal Lo 0 Hi 0
```

In the top-level menu, use the up/down buttons to select the Lock menu.

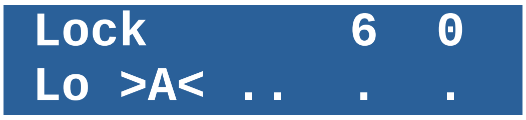
Use the right-button to move through the fields, selecting the desired node and pin with the up/down buttons as you go, just as for Output definitions.

The example shows module 6, pin 0 (a signal) has no locks set in either of its sets, Lo or Hi.



```
Lock       6  0
Lo< A  ..  .  .
```

Use the right-button in the pin field and the bottom row shows the first (labelled ‘A’) Lo lock. The dots indicate “disabled”.

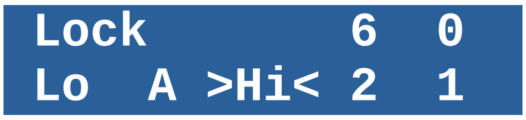


```
Lock       6  0
Lo >A< ..  .  .
```

Use the up/down buttons to select the output’s Lo or Hi set, then use the right-button to configure a lock.

Pressing the select-button here will exercise the complimentary output (if set) for indication/check.

Use the up/down buttons to select the desired entry, there are four labelled ‘A’ to ‘D’. There is no significance to the order in which they are defined, all the entries will be utilised when testing for locks.

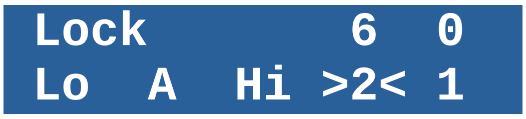


```
Lock       6  0
Lo  A >Hi< 2  1
```

Use the right-button and choose the Lo or Hi state for the complimentary output.

Pressing the select-button here will enable/disable the lock entry.

In the example, output 6, pin 0 (a signal) has its first (‘A’) Lo lock set against output 2, pin 1 being Hi.

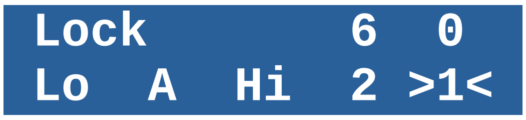


```
Lock       6  0
Lo  A  Hi >2< 1
```

Use the right-button and then the up/down buttons to specify the output node to lock against. Locks can be set against any module, any pin.

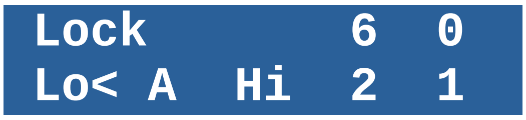
Pressing the select-button here will exercise the complimentary output for identification/check.

Use the right-button and then the up/down buttons to adjust the output pin in the same manner. Again, pressing the select-button here will exercise the output.



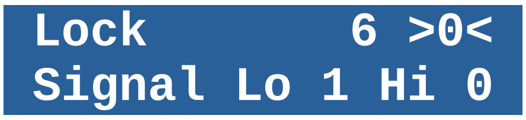
```
Lock       6  0
Lo  A  Hi  2 >1<
```

Return to the first field (“Lo”) and use the left or select button to cancel or confirm the changes.



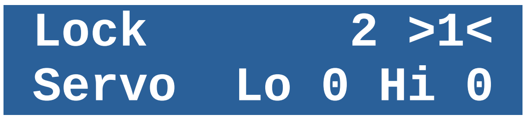
```
Lock       6  0
Lo< A  Hi  2  1
```

The panel now shows there’s one Lo lock defined for this output.



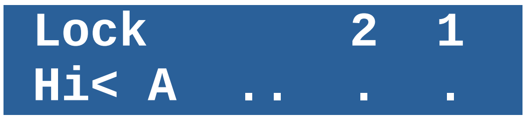
```
Lock       6 >0<
Signal Lo 1 Hi 0
```

To set the complimentary lock, select the desired node and pin as before. As usual, pressing select on the pin field will exercise the output for identification/check.



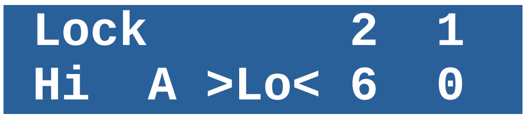
```
Lock       2 >1<
Servo  Lo 0 Hi 0
```

Use the right-button and then the up/down buttons to select the Hi set for this output.



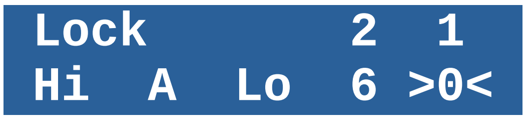
```
Lock       2  1
Hi< A  ..  .  .
```

Use the right-button to move to the state field, and the up/down buttons to select Lo state.



```
Lock       2  1
Hi  A >Lo< 6  0
```

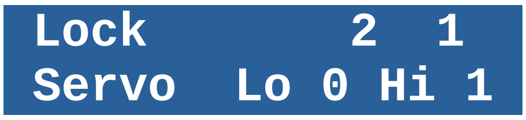
Use the right-button and then the up/down buttons to select the complimentary output.



```
Lock       2  1
Hi  A  Lo  6 >0<
```

The example shows output 2, pin 1 (a servo) locked when Hi against output 6, pin 0 (a signal) being Lo.

Use the left or select button to cancel or confirm the changes.



```
Lock       2  1
Servo  Lo 0 Hi 1
```

The panel now shows there’s one Hi lock defined for this output.

Notes.

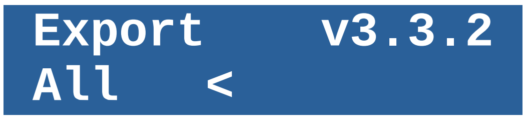
1. It’s possible to specify locks against a pair of outputs where both outputs are in the same state (Hi or Lo) – this is perfectly reasonable.
2. It’s possible to specify a lock against the same output module and pin. This isn’t useful.

Export menu

The configuration can be exported via the Uno’s serial IO using a USB cable connected to a computer.

Some software capable of reading from the USB socket is required on the computer. The Arduino IDE is ideal for this.

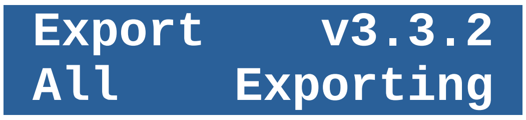
In the top-level menu, select the Export menu and then use the right-button to move to the export sub-menu.



```
Export    v3.3.2
All   <
```

Use the up/down buttons to select the definitions to export (the system, the outputs, the inputs, the locks) or select “All” to export everything.

Use the right-button to perform the export. The panel will show that it’s exporting for a brief period.



```
Export    v3.3.2
All    Exporting
```

The output can be saved for use in the import function.

The export output is human readable, and can even be edited though that is not recommended.

There will be one line for the system configuration, 16 lines for each input module, 8 lines for each output module, 8 lines for each output module’s locks plus some comment lines (which start with a ‘#’ character), something like this (many lines have been omitted for brevity):

```
#System Version Report  Debug
System  v3.1.4  Long    Errors

#Input  Node    Pin     Type    OutA    OutB    OutC    OutD    OutE    OutF
Input   4       0       On_Off  2 1     . .     . .     . .     . .     . .
Input   4       1       On_Off  6 0     . .     . .     . .     . .     . .

Input   5       0       On_Off  . .     . .     . .     . .     . .     . .
Input   5       1       On_Off  3 0     . .     . .     . .     . .     . .
Input   5       2       On_Off  3 1     . 3     3 2     . .     . .     . .

#Output Node    Pin     Type    Lo      Hi      Spd     Reset
Output  2       0       None    00      FF      00      00
Output  2       1       Servo   B4      00      0D      00
Output  2       2       Flash   4F      4E      0D      00
Output  2       3       Blink   51      53      0E      00
Output  2       4       Random  7F      7F      0C      20

Output  3       0       LED     27      27      0C      00
Output  3       1       Servo   3F      3F      0D      00
Output  3       2       Servo   3F      3F      0E      00
Output  3       3       LED     3F      3E      0D      00
Output  3       4       LED_4   3F      3E      0D      05
Output  3       5       LED     3F      3E      0D      01
Output  3       6       RoadUK  3F      3E      0D      08

Output  6       0       Signal  B4      00      0D      00

#Lock   Node    Pin     LockLoA LockLoB LockLoC LockLoD LockHiA LockHiB LockHiC LockHiD
Lock    2       1       .       .       .       .       Lo 6 0  .       .       .

Lock    6       0       Hi 4 1  .       .       .       .       .       .       .
```

Import menu

A previously saved export can be imported to restore the state if it’s been lost or damaged using a USB cable connected to a computer.

As for export, some software capable of writing to the USB socket is required on the computer. The Arduino IDE can be used for this.

In the top-level menu, select the Import menu and then use the right-button to move to the import mode.

```
Import
Waiting
```

Using the Arduino IDE or other suitable comms software, send all or some of the data from a previously exported system to the Arduino via the USB cable.

As each line is processed, a message indicating what was processed is displayed.

```
Import    System
```

An input, module 4, pin 0 (an On_Off input).

```
Import    Input
On_Off     4  0
```

An output, module 2, pin 1 (a servo).

```
Import    Output
Servo      2  1
```

A lock for output module 6, pin 0 (a signal).

```
Import    Lock
Signal     6  0
```

When the importing is finished, the waiting message re-appears. Use any button to return to the top-level menu.

```
Import
Waiting
```

Advanced features

This section covers two advanced features.

Module node ID

The output modules normally set their module-ID using the jumpers on the PCB. However, the modules can also have their ID set using software. This will be necessary to set the module IDs greater than 'F' as there are only 4 jumpers.

If using software-assigned IDs, it's usual to leave all the jumpers off – the module will initially set its ID to 0. As no two modules can have the same number, it's necessary to add the modules to the system one at a time. Also, there should be no module 0 in the system already.

In the output menu, press the select button when on the node number field.

```
Output     0  0
New node# >0<
```

Use the up and down buttons to choose a new node number for this module. You won't be able to choose numbers that are currently in use.

As up to 32 output nodes can be configured, they use the "numbers" '0'-'9' and 'A'-'V'.

```
Output     0  0
New node# >M<
```

Use the select button to confirm the choice (or left-button to cancel), and the normal confirmation dialog is shown

```
Output     0  0
Confirm? Sel=Yes
```

The display now shows the input nodes being re-programmed.

```
Renumber   0  0
Input     ....45..
```

And then the outputs with the new node number in place.

```
..23..6.........
......M.........
```

And then the normal output menu is shown once more. All the pins of the new module will be of type "None" initially.

```
Output     M  0
None
```

This method can be used to re-number an existing node. All the node's definitions and the corresponding links to input modules' definitions (specifying which output the inputs operate) and lock definitions (specifying interlocks between outputs) will be adjusted to reflect the renumbering.

To cancel a module's software node ID (and revert to its jumper settings), use the right-button when on the "New node#" field and a '.' will be shown. Now press select to confirm the change as normal – the module will be reset to use its jumper settings as its node ID.

Input pin scanning

When configuring inputs, it might be easier to just press the input in question rather than using the up and down buttons to select an input node and pin.

```
Input      >4< 0
On/Off   21 .. ..
```

In the input menu, on the node number field, press the select button. The display shows it's waiting for an input to be pressed/switched.

```
Input       4  0
Scanning inputs
```

Now press the desired input button or switch.

The appropriate input configuration is selected. In the example, it's module 5, pin 2 operating output module 3, pin 1, and output module 3 pin 2 with a 3-second delay between.

```
Input       5  2
On_Off   31 .3 32
```

Press the select button, and the desired input is selected. Now proceed with configuration as normal.

```
Input      >5< 2
On_Off   31 .3 32
```

If it's the wrong input, either press the correct input (a summary of the new input will be displayed), or press any button other than the select button, and input-selection is cancelled.

Operation

Booting

In normal operation, just power-up and wait for the initialisation process to run through (about 10-15 seconds).

Whilst booting, the Uno and the Nanos will flash their software version number using their built-in LEDs. For example, if the version number is 3.3.2, there'll be 3 short flashes, a pause, 3 short flashes, a pause and then 2 short flashes. Zeros are shown as a single long flash.

The built-in LED is pin 13, and this is also available on the LCD shield ICSP header to drive an external LED if desired.

The Nanos (but not the Uno) then also report their module ID. This is normally as set on their jumper pins, a short flash for Hi, a long flash for Lo. There are four jumpers which can specify a module ID from 0 to 'F' (hexadecimal). So for example, if jumpers 0 & 1 are low, and 2 & 3 are high (specifying module number 'C' hexadecimal, 0011 binary), then there'll be a long flash for jumper 0, a long flash for jumper 1, a short flash for jumper 2, a short flash for jumper 3.

If a software module number has been set (see the advanced configuration section above), then the number set in software is reported rather than the physical jumper settings. Software IDs can be greater than 'F' so there's a virtual fifth jumper that's not present on the PCB. If the module number is greater than 'F' then the virtual jumper is deemed set and the first four jumpers select between module 'G' and 'V'. There will now be five short or long flashes reflecting the value of the software module ID virtual jumpers.

Configuration

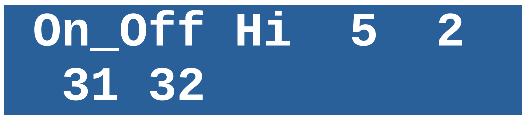
To re-configure the setup, press any button on the LCD display and the configuration menu is shown (see above).

When in configuration mode, no inputs are actioned as this might confuse the configuration process. When configuration mode is exited, the inputs are once more honoured.

Report output

If report output is enabled (see Report sub-menu configuration section above), then when inputs are pressed, a report is generated on the display (if so configured) for a few seconds.

Typical output will look like this.



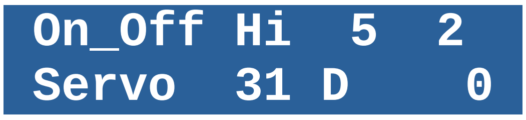
```
On_Off Hi  5  2
 31 32
```

This example shows an On_Off input (going Hi) on input module 5, pin2.

Two outputs have been actioned, module 3 pin 1 and module 3 pin 2.

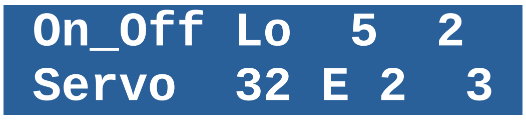
These messages will persist for a few seconds – around 2 seconds if the reporting level is “Brief” and around 4 seconds if the level is “Long”.

If the reporting level is “Pause”, then when these messages appear, more detail is provided, and the operator must press a button to continue.



```
On_Off Hi  5  2
Servo  31 D    0
```

The messages will describe each action in turn in a long format. The example shows input module 5, pin 2 has been pressed, and output module 3, pin 1 (a Servo) is about to be actioned. The speed (D) and reset value (if any, blank in this example) of the output are also shown. Finally, the delay before actioning this output is shown (0 in this case).



```
On_Off Lo  5  2
Servo  32 E 2  3
```

Press the select button and the output is actioned and the next in line is displayed. In the example module 3, pin 2 (another Servo) which has a speed ‘E’, a 2-second reset interval (after which it will reset to Lo) and also an accumulated delay of 3 seconds. So it will delay 3 seconds before operating, and then revert to Lo after a further 2 seconds.

Instead of pressing the select button, the other buttons can be used with each button having a separate effect.

Select	Acknowledge the message, action the output and move to the next output.
Left	Set the reporting level to “None”, and continue. There will be no more reports.
Down	Set the reporting level to “Brief”, and continue.
Up	Set the reporting level to “Long”, and continue.
Right	Enter configuration mode to enable adjustment of the input or output configuration.

Lock Messages

When interlocks prevent an output from being actioned, then a suitable message is shown (if the reporting level is set to “Short” or “Long”).

The example shows On_Off button on module 4, pin 1 was pressed, but output module 6, pin 0 can’t go Lo as that would conflict with output module 2, pin 1 being Hi. The ‘v’ and ‘^’ symbols indicate “Lo” and “Hi” locks respectively.

```
On_Off Lo  4  1
Lock  v60 vs ^21
```

Here the complimentary action is being attempted. On_Off button on module 4, pin 0 was pressed, but output module 2, pin 1 can’t go Hi as that would conflict with output module 6, pin 0 being Lo.

```
On_Off Hi  4  0
Lock  ^21 vs v60
```

Note that when interlocks prevent an action, as well as the messages on the LCD display, pin 13 on the Uno (also available on the shield ICSP header) is held high for a short while. This could be used to light a warning LED for example.

A warning sound is available on pin 12 of the Uno (also available on the shield ICSP header). This pin has a (two-tone) PWM signal generated on it that could be used to drive a buzzer for an audible warning that a lock has been encountered..

Serial commands

It’s possible to send commands to the Uno over the USB serial link from suitable software on a computer, again the Arduino IDE can be used.

The commands understood by the software are all exactly three characters long, the first character indicates the command, the second the node and the third the pin.

The command character can be upper-or lower-case and one of:

I i	Operate the input identified by the node and pin.
L l	Operate the output identified by the node and pin to its “Lo” state.
H h	Operate the output identified by the node and pin to its “Hi” state.
O o	Operate the output identified by the node and pin changing its state from whichever state it’s in to its alternate state.

The second character (identifying the node to operate) must be between ‘0’ and ‘7’ for input commands, and between ‘0’ and ‘9’ or ‘A’ and ‘V’ for output commands.

The third character (identifying the pin to operate) must be between ‘0’ and ‘9’ or ‘A’ and ‘F’ for input commands, and between ‘0’ and ‘7’ for the output commands.

For input commands, the Uno will operate exactly as though the specified input was actioned. For output commands it will just operate the identified output.

CMRI interface

It's possible to communicate with SignalBox over the USB serial link from a computer which is using the CMRI protocol (eg with JMRI).

SignalBox emulates a CMRI "USIC_SUSIC" board with up to 384 inputs and 384 outputs. Any of the SignalBox inputs or outputs can be operated or sensed by CMRI.

Configuration

Configure the Uno as an "USIC_SUSIC" board with 32-bit cards. It should have up to 12 input cards, and up to 12 output cards. Any node address (UA in CMRI terminology) can be selected as SignalBox doesn't attempt to filter by node address.

The first 4 input cards will reflect the state of the (up to 8) SignalBox input nodes (128 inputs) and the other 8 input cards the state of the (up to 32) SignalBox output nodes (256 outputs). The CMRI interface relects the state of both the input and output SignalBox nodes. The input nodes reflect whether they're active (low) or not (high), the output nodes indicate if they're in their "Lo" or "Hi" state.

The output cards allow any of the SignalBox Inputs or Outputs to be actioned. Actioning an input node (on the first 4 cards) will operate just as if the corresponding input in SignalBox was actioned. Actioning an output node (the other output cards) will operate the corresponding SignalBox output node.

Mapping

CMRI numbers bits on the cards making up a node from one, SignalBox numbers both its nodes and pins from zero, which makes the mapping slightly awkward as all the numbers are adjusted by one.

The CMRI node configured above can have up to 384 sensors numbered 1 to 384. The first 128 map to SignalBox input nodes 0 to 7. The remaining 256 sensors map to SignalBox output nodes 0 to 31 (or 'V').

To convert a SignalBox input node and pin to the corresponding CMRI sensor use the formula:

node * 16 + pin + 1	eg: node 5, pin 3 is 5 * 16 + 3 + 1 = CMRI sensor 84.

For output nodes, use the formula:

node * 8 + pin + 129	eg: node 2, pin 1 is 2 * 8 + 1 + 129 = CMRI sensor 146.

Converting the other way is slightly more involved. For input nodes:

Subtract 1	eg: Sensor 84 subtract 1 is 83.
Divide by 16 for the node	83 divided by 16 is node 5.
Use the remainder for the pin	with remainder giving pin 3.

For output nodes:

Subtract 1	eg: Sensor 146 subtract 129 is 17.
Divide by 8 for the node	17 divided by 8 is node 2.
Use the remainder for the pin	with remainder giving pin 1.

CMRI outputs (eg turnouts) are mapped in exactly the same fashion.

Table showing some of the mappings:

CMRI	**SignalBox input**		**CMRI**	**SignalBox Output**	
Sensor	**Node**	**Pin**	**Sensor**	**Node**	**Pin**
1	0	0	129	0	0
2	0	1	130	0	1
…					
7	0	6	135	0	6
8	0	7	136	0	7
9	0	8	137	1	0
10	0	9	138	1	1
…					
15	0	14 (hex E)	143	1	6
16	0	15 (hex F)	144	1	7
17	1	0	145	2	0
18	1	1	146	2	1
…					
127	7	14 (hex E)	255	7	6
128	7	15 (hex F)	256	7	7
			257	8	0
			258	8	1
			…		
			271	9	6
			272	9	7
			273	A	0
			274	A	1
			…		
			383	V	6
			384	V	7

Turnouts

Create turnouts using “1 bit” and “Pulsed Output”. Map them to the actual SignalBox turnout, or to its associated input (if it has one).

Use feedback mode “OneSensor” connected to the SignalBox output. If turnouts have position sensors, use those for feedback using “OneSensor” or “TwoSensor” as appropriate.

Sensors

Create sensors mapped to the appropriate SignalBox inputs, except where the state of an output is sufficient (eg position of a turnout which has no position sensor), in which case map the sensor to the SignalBox output concerned.

Examples

Refer to the demonstration layout above.

To make turnout 21 driven directly by SignalBox output 21:

Create turnout number 146 (2 * 8 + 1 + 129).

Create a feedback sensor also number 146 (2 * 8 + 1 + 129).

Alternatively, to operate the turnout by its SignalBox input (40):

Create turnout number 65 (4 * 16 + 0 + 1).

Create a feedback sensor as above (number 146).

To make turnouts 31 & 32 operate (using SignalBox input 52):

Create a turnout number 83 (5 * 16 + 2 + 1) to operate both turnouts by simulating input 52.

Create a sensor number 154 (3 * 8 + 1 + 129) for turnout 31.

Create a sensor number 155 (3 * 8 + 2 + 129) for turnout 32.

Note that operating turnouts via the SignalBox inputs will preserve all the functionality specified for the input in SignalBox (eg operate a second turnout after a short delay, enforce interlocks etc). Operating turnouts directly via their SignalBox outputs will operate the turnouts immediately and any interlocks between outputs will be ignored.

Extra buttons

If desired, the LCD buttons can be augmented with push-buttons or switches on the following pins of the Uno, and this will be necessary if running without an LCD shield:

Analog pin 1	Select button.
Analog pin 2	Left button.
Analog pin 3	Down button.
Digital pin 2	Up button.
Digital pin 3	Right button.

Wire the pins through a suitable momentary-contact switch to ground.

Extra LCD

A second LCD can be added to the system. This must be of the I^{2}C type and have 4 rows of 20 characters. Such devices normally have an I^{2}C address of 0x27 or 0x3F but can often be configured to other addresses.

The software will scan addresses 0x3F down to 0x27 for such a device. If one is found, then all the messages will be output to this alternative panel.

If an I^{2}C LCD is attached, it’s assumed there’s no LCD shield. If both are attached, put a jumper on pin 11 (available on the LCD shield) to ground. This will force the SignalBox software to drive both displays.

If using just an I^{2}C LCD, then the alternative buttons (above) must be used.

Uno pins.

Three Uno pins have special functions, and these three pins are available either directly on the Uno, or on the LCD shield ICSP header.

Digital pin 11	Indicates the presence (or otherwise) of an LCD shield. If an LCD shield and an I^{2}C LCD display are attached, this pin should be jumpered to ground.
Digital pin 12	Indicates an interlock has been encountered. It is intended to be connected to a buzzer. A PWM two-tone signal is used.
Digital pin 13	This is the built-in LED on a Uno. It will flash the software version number at start-up. An external LED could be attached to this pin. This pin is also illuminated for a short while when an interlock has been encountered.

For completeness, the other Uno pins are described here:

Digital pins 0 & 1	Used to provide serial comms, Rx and Tx.
Digital pins 2 & 3	Used as alternative LCD buttons – Up and Right.
Digital pins 4-10	Used to drive the LCD shield if present.
Analog pin 0	Used by the LCD shield to provide its buttons.
Analog pins 1-3	Used as alternative LCD buttons – Select, Left and Down.
Analog pins 4-5	I^{2}C protocol bus pins.