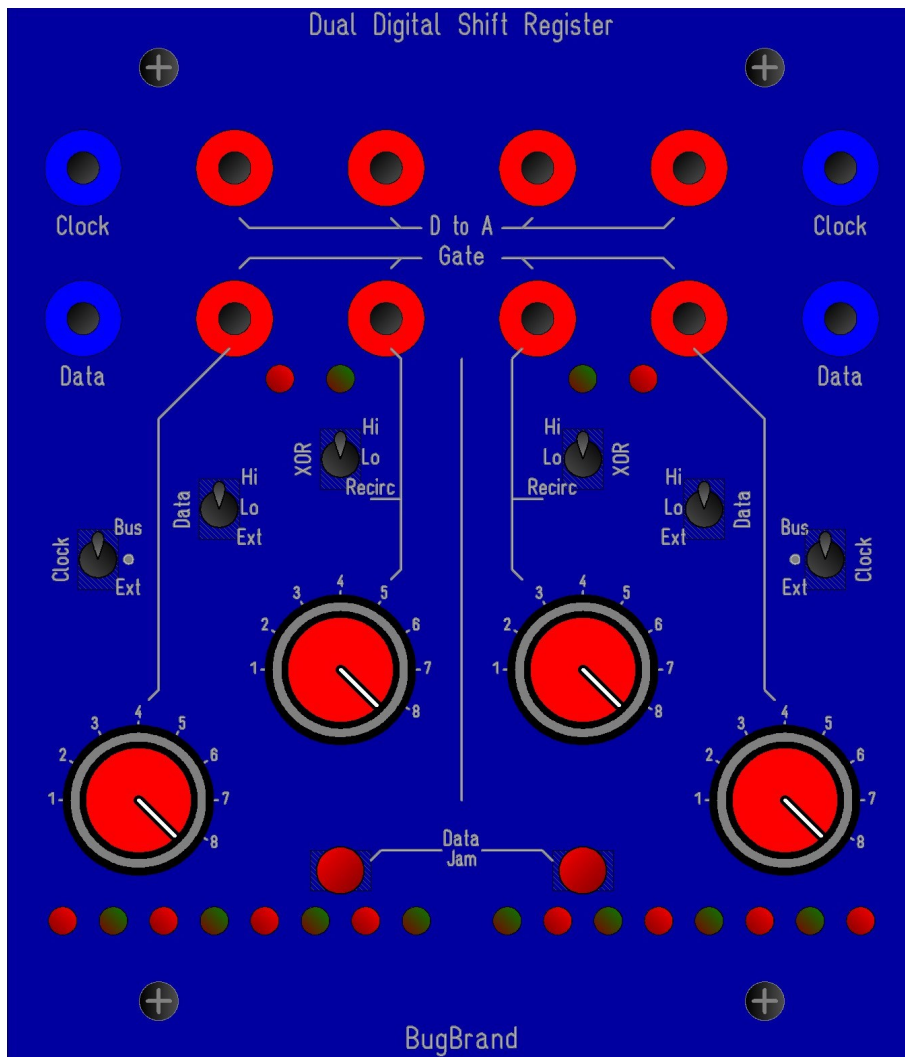


Dual Digital Shift Register (DDSR)

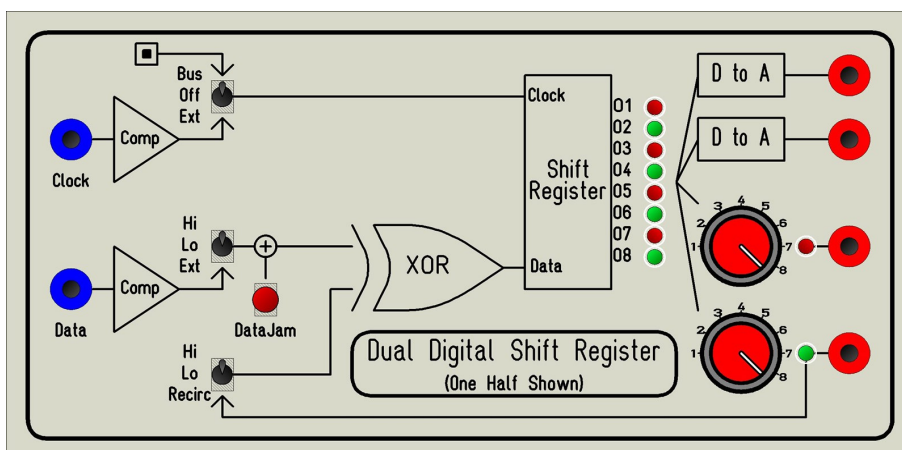


The DDSR features two identical / mirrored Shift-Registers. While its internals are relatively minimal, it is hard to simply convey all the possible behaviours – clock division, delaying, random generation, audio processing, etc. – the best approach is to learn by playing!

Each side is built around an 8 stage logic array, where each stage can be either logic 0 (low/off) or 1 (high/on), as shown on the line of 8 LEDs. Each time a clock event occurs the array shifts along one step, with the first stage being filled with whatever is present on the Data/XOR line.

Two 'Data Tap' Gate Outputs are taken from the register, along with two Digital-to-Analogue sums which provide stepped waveforms.

The block diagram below shows the layout for each Shift-Register:

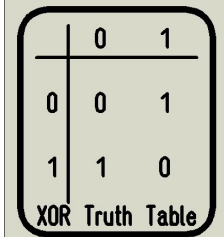


Clock Input & Switch: Each side can be clocked either by an external signal (switch set to Ext) or from an internal Clock-Bus signal (eg. From Clock section of the ClockBox Frame) – clocking can also be switched off simply with the switch set to its middle position.

The Bus lines, internal to the frame, carry two clock signals (eg. From each side of the master Clock module) and rear jumpers allow you to pre-configure which clock is used on each side of the DDSR.

The External input passes through a comparator (Threshold c.+1V) so that any waveform can be used, with the data array being shifted on the rising edge of any clock event. Clocks do not have to be in any way periodic/regular and can happily run up through the full audio spectrum.

Data Input & Switch, XOR & Switch: A two input Exclusive OR logic gate feeds the Shift Register's Data line. The Truth Table shows how it can be thought of as a logic controlled inverter – if one input is low, the other input passes through normally, but if the first input is taken high then the other input will be inverted. By allowing control of each input via the switches, we can manipulate the Data line in a number of ways.



	0	1
0	0	1
1	1	0

XOR Truth Table

The first XOR input comes from the Data Input, Data Switch and Data Jam button:

- With the Switch set to Ext, an external signal (via Comparator with threshold c.+1V) determines the logic state. You can input waveforms, noise, gates etc. - remembering, of course, that it is only the state when a clock event occurs that really matters.
- With the Switch set to Lo, whatever is happening on the other XOR input determines the output. And, conversely, with the Switch set to Hi, whatever happens on the other XOR input is logically Inverted.
- The Data Jam button can insert a High state, but only on a Low state – so it will have no effect if, for example, the Switch is already set to Hi.

The second XOR input can again be set to logic Lo or Hi, with corresponding behaviour, but it can also take one of the Data Taps to allow Recirculation (data looping).

Data Taps: Each side has two 8-way rotary switches which 'tap off' the register's data array. The outer one is independent, while the inner one I tend to see as the Recirculation Tap – though, of course, it does not always have to be recirculating/looping. For each Tap, the Gate output changes between 0V and +10V dependent on whether the step is Lo or Hi – the state remains constant until the next clock event, when the array shifts along.

If the register is looping (recirculating) then adjusting the independent tap allows you to effectively move/offset the position of its Gate pattern.

D to A Outputs: The Digital to Analogue Outputs use a method of weighted summing (R-2R) to produce a waveforms from the 8 data stages. The two outputs on each side each use different stage combinations so that they behave differently – the outer ones sum 4 stages, while the inner ones sum all 8 stages.

The outputs all vary between 0V and +10V and are very much tied to what is in the data array. As such, if you have a recirculating pattern in the array then you will get looping patterns from the D-to-A outputs.

Patch Ideas:

- **Inverting Pattern** – Set a pattern to loop with, for example, Recirc set to stage 8. With Data set to Lo, you get an 8 step loop. When Data is set to Hi, each recirculation inverts the data, so you get a 16 step loop.
- **Clock Divisions** – Starting with the array empty (eg. clock with Data and XOR switches set to Lo), set the Recirculation tap to 2 and input one data bit – this should result in an alternating pattern in the array. Now try switching to longer recirculation settings and changing the Data switch between Hi and Lo. This works well into audio rates to create sub-octaves, though at audio rates it works best to keep the Data switch set to Hi. You can listen to either to the D-to-A outputs or the Gates – the Gate output will naturally sound harsher as it is a pure square wave.
- **Pulser Waves** – Continuing with the audio rate clock divisions, clock from an audio rate oscillator and listen to the output from the D-to-A (or Gate). But now set the Data switch to Ext and feed it with a sub-audio signal – eg. LFO or gate pattern from the 2nd half of the DDSR. Each time an event occurs on the Data line, the audio pattern is 're-seeded', altering the tonality.
- **Digital Noise Pattern** – Digital Noise Generators typically use a pre-patched arrangement of Shift-Registers (eg. 24 to 32 stages) and XOR(s), the inputs of which are chosen such that resulting pattern only repeats after many thousands of clock steps, resulting in what appears to be random patterns. A similar-but-shorter version can be patched up by connecting the independent tap back into the Data input (switch set to Ext). The taps need to be different – trying 5 or 7 on one and 8 on the other is a good starting point. This can be run either at low frequencies for quasi-random clocking, or try it clocked by an audio oscillator to hear the noise-ish results (you can hear the periodicity).
- **Longer Patterns with Two Registers** – In a somewhat similar fashion, longer patterns can be set up by mixing Data sources with Recirculation. Set up a regular pattern on one register (eg. 8 steps) and take one tap over to the other register's Data input. Set the second register to recirculate with a different length (eg. 7 steps). Further Data & Recirc pattern combinations will become apparent when using external clocking modules such as dividers and logic gates – as shall be found in the ClockBox Frame.