Simple Sequencer Hardware

Project Name: Simple Sequencer

Project Manager: Jarmo Verho

Prepared by: Minh Nguyen, Minh Do, Thong Nguyen and Jarmo Verho

Issue Date: 9.5.2021

Version: 1.1

Distribution: Public

Synopsis: Description of the Simple Sequencer hardware.

Version history:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 0.1 | JV | 10.3.2021 |  |  | Initial version |
| 1.0 | JV | 5.4.2021 | JV | 5.4.2021 | First public version |
| 1.1 | JV | 9.5.2021 | JV | 9.5.2021 | Added pin header info |
|  |  |  |  |  |  |

Basic operation and features

Simple Sequencer is an analog sequencer. From electronics point of view, an analog sequencer is a counter, looping through a set of steps, producing user-defined actions for each step. In Simple Sequencer’s case, these actions include producing ***gate*** output pulses and ***control voltage*** variable outputs. The gate signals are digital, that is, they can be used as on/off or trigger style controls for the other instruments. The control voltages are continuously variable and can be used to control things like pitch and volume of the other instruments. For example, it would be possible to use the sequencer to run a short melody loop by using the control voltage output to control an oscillator’s pitch and a gate output to trigger a drum machine.

The sequencer has two gate-compatible inputs, ***clock*** and ***reset***. When the clock input receives a pulse, the sequencer advances to the next step. The reset input terminates the current sequence and returns the sequencer back to the sequence’s first step. Thus, the reset input can be used to limit the sequence’s length. The instrument also has two push-button switches for producing clock and reset inputs manually.

The gate and the control voltage signals use Eurorack standard unipolar levels. The nominal levels for the gate signals are 0V (logical low, “off”) and +5V (logical high, “on”). The control voltage range is from 0V to +8V. When used to convey pitch information, the scale is 1V per octave.

In addition to these incoming and outgoing signals, the instrument has group of mode selection switches and two adjustments. The user can select the sequence’s maximum length (4 or 8 steps), the stepping direction (forwards/backwards), portamento (on/off) and whether the clock signal should come from an external input (like explained above) or from the internal source. The two adjustments include a speed adjustment for the internal clock and the portamento, affecting how quickly the control voltage outputs shift from one step’s output to the next.

Simple sequencer is able to produce two control voltage outputs and 2 + 2 gate outputs; two gate outputs are freely programmable, while the other two are their inverses. The internal clock is adjustable from 40 to 200 beats per minute, while the external clock input allows significantly wider range – the current software allows anything from zero to 7500, although timing inaccuracies are likely to set the practical upper limit much lower.

Implementation

Simple Sequencer hardware can be divided in a group of functional blocks:

* ***General control*** (microcontroller and its immediate support components). This is done with an Arduino Nano microcontroller board, implementing the actual sequencer functionality. In other words, it reacts to the inputs and performs operations in sequence.
* ***Gate switches and the gate signal generation***. This block is driven by Arduino and it allows user to select (with switches) when gate pulses are to be generated. This block also includes some LEDs to indicate the state, like the current step and whether the gate pulse is being produced or not.
* ***Control voltage production and handling.*** This is very similar to the gate signal handling block: Arduino drives the block, which contains potentiometers so that the user can adjust the produced control voltages, step by step. The block also includes some analog signal processing to scale the output voltages to Eurorack standard levels. It also includes a portamento feature.
* ***Mode selection switches and adjustments.*** This is just a group of 3 switches and one potentiometer. Arduino checks the states of the switches and the potentiometer periodically and changes the operating mode (sequence direction, sequence length, clock source and internal clock speed) accordingly. In other words, all the actual functionality is done in software.
* ***Clock and reset input*** block routes the external gate-level signals to Arduino. This includes some protection against overvoltage and reverse voltages. The block also contains push button switches so that the user can produce clock and reset pulses manually.
* ***Power management***. This block includes a connector for Eurorack power, some filtering and also an option for using an external +12V power supply (for non-Eurorack operation). If an external power supply is used, the instrument needs to produce its own -12V power supply, which is done in this block. The block also includes some protection against overvoltage and short circuits. Note that Arduino contains an additional regulator, which produces the +5V supply from the +12V input.

General control

The instrument uses Arduino Nano for control. However, the software does not use the Arduino libraries and controls the processor directly. Thus, Arduino behaves like a generic ATmega328P based microcontroller board, running at 16MHz clock. In case the user wants to modify the current software, Microchip’s datasheet [[1]](#footnote-1) can be used as a general reference. An alternative to that would be to replace the current software with a more Arduino-like approach. In that case the user should refer to Arduino’s documentation [[2]](#footnote-2).

Table 1 summarizes how the Arduino pins are connected: what are their signal names in the schematic and what are their actual microcontroller port names (as used in the device firmware). The table also lists how the pin directions should be programmed (digital in, digital out, analog in). Five pins are not connected and can be used for additional features and user modifications. One of these unused pins, IO13, is connected to a LED in Arduino itself. In case the user wants to use these pins, their directions must be defined accordingly in the software. The default software configures them as inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal/function | Dir | AVR | Arduino | Notes |
| IO0, gate switch 1 | Out | PD0 | D0 | IO0…7: 1 out of 8 high |
| IO1, gate switch 2 | Out | PD1 | D1 | IO0…7: 1 out of 8 high |
| IO2, gate switch 3 | Out | PD2 | D2 | IO0…7: 1 out of 8 high |
| IO3, gate switch 4 | Out | PD3 | D3 | IO0…7: 1 out of 8 high |
| IO4, gate switch 5 | Out | PD4 | D4 | IO0…7: 1 out of 8 high |
| IO5, gate switch 6 | Out | PD5 | D5 | IO0…7: 1 out of 8 high |
| IO6, gate switch 7 | Out | PD6 | D6 | IO0…7: 1 out of 8 high |
| IO7, gate switch 8 | Out | PD7 | D7 | IO0…7: 1 out of 8 high |
| IO8, int/ext clock | In | PB0 | D8 | Low: external, high: internal clock |
| IO9 |  | PB1 | D9 | Unused |
| IO10 |  | PB2 | D10 | Unused |
| IO11, direction switch | In | PB3 | D11 | Low: forwards, high: backwards |
| IO12, mode switch | In | PB4 | D12 | Low: 8 step, high: 4 step mode |
| IO13 |  | PB5 | D13 | Unused / Arduino LED |
| IO14, CV multiplexer A0 | Out | PC0 | A0 | IO14...16: 3-bit binary code |
| IO15, CV multiplexer A1 | Out | PC1 | A1 | IO14...16: 3-bit binary code |
| IO16, CV multiplexer A2 | Out | PC2 | A2 | IO14...16: 3-bit binary code |
| AD3/IO17 |  | PC3 | A3 | Unused |
| AD4, Clock input | In | PC4 | A4 | Low-to-high transition: clock pulse |
| AD5, Reset input | In | PC5 | A5 | Low: normal operation, high: reset |
| AD6 |  | ADC6 | A6 | Unused |
| AD7, Clock speed adjust | Analog | ADC7 | A7 | 0..5V: speed adjustment |

**Table 1**. Arduino connections.

Gate switches and gate outputs

The gate switch logic is the largest functional block in the schematic. However, it is functionally very simple: the 8 Arduino outputs are used to provide power to one of the 8 gate switches. This powered switch then re-routes the power either to the normal gate output, or to the inverted gate output. Thus, for each step the sequencer always generates either a normal gate output or an inverted gate output – there is no option for not generating an output.

There are five complications to this basic scheme:

1. There are two groups of 8 gate switches. So, for each step, the instrument produces two independent gate outputs and their inverses.
2. There are 8 LEDs (and their current limiting resistors) for indicating the currently selected step.
3. There are additional 2 x 2 LEDs for indicating the states of the two gate outputs and their inverses.
4. There are 2 x 8 diodes for preventing the switches from affecting each other. Without diodes the switches could short Arduino IO pins, possibly breaking it.
5. The gate output connectors have small resistors (1kΩ) in series with them to protect the Arduino from short circuited gate outputs.

Some simple modifications include:

* Replacing the current 2-position switches with 3-position switches (on-off-on). The third position would allow disabling gate output generation entirely for any given step.
* Changing the indicator LED colors and their brightness. In case the user wants to change brightness, the series resistor can be calculated using formula R = (5V – VLED) / ILED, where VLED is the LED forward voltage drop (around 1.8V for most reds, 2.2V for greens and yellows and 3.2V for blues and whites). It is advisable to keep the LED current below 5mA to limit the heating of Arduino’s regulator.

Control voltage production, portamento

While the control voltage section is superficially very different from the gate signal generation section, the basic operational principle is largely the same: the processor selects one of the 8 control voltage adjustment potentiometers, which is then routed to the control voltage output. However, instead of using the processor pins to power potentiometers directly, the design uses an 8-to-1 analog multiplexer to select one of the potentiometers. The processor then controls this multiplexer with 3-bit binary code, via pins IO14…16. Thus, when all the three pins are low, the very first potentiometer is selected. The advantage of using an analog multiplexer (8-to-1 analog switch) instead of diode gating is that the multiplexer passes the potentiometer’s voltage to its output without significant losses, unlike diodes. Like was the case with the gate switches, there are two sets of eight of these potentiometers.

The potentiometers are connected between the ground and the +5V rail, so their output voltages (and thus the outputs of the multiplexers) are in range 0…+5V. However, the Eurorack standard requires the control voltages to be in range 0…8V. This is achieved by amplifiers with a nominal gain of 1.6. The resistors connected to amplifiers U3A and U3B set the gain: with the nominal component values, the gain is for each amplifier is G = 27kΩ/47kΩ + 1 ≈ 1.57.

The instrument also includes a portamento feature, that is, the control voltage output changes gradually from one value to another, instead of changing nearly instantly. This is achieved by a simple R-C filter: a 20kΩ potentiometer, acting as an adjustable resistor, together with a 10µF capacitor has an adjustable time constant from 0 to 200ms. The time constant means the time it takes to reach 63% of the final voltage. After another time constant, the voltage rises 63% of the remaining difference, and so on. Thus, it takes around four time constants to get to within 1% of the final voltage. The circuit also includes a switch for quickly turning the portamento on and off.

There are some complications with this kind of portamento:

* Operational amplifiers (U3A/U3B) generally work badly if they’re connected directly to a large capacitor. Thus, they have 100Ω isolation resistors in their outputs.
* The 20kΩ potentiometer is directly in series with the amplifier output. Due to this, the portamento filter can’t be connected directly to other Eurorack modules – their input loading would cause significant changes in control voltage, depending on portamento setting.
* When switched off, the 10µF filter capacitors will remember the last control voltage for a while. When portamento is switched back on later, there will be a sudden “portamento” from this old voltage to the new value.

To avoid the second problem, the portamento voltage is buffered by two additional amplifiers with a gain of 1. In order to protect them from short circuits in their outputs, they have 1kΩ protection resistors at their outputs. While this protection resistor causes similar problems as the 20kΩ portamento potentiometer, at least the gain error is much smaller and constant.

Simple modifications to this section could include:

* Portamento potentiometer and/or capacitors of different value, to change the adjustment range.
* Large value resistor (100kΩ, for example) from portamento capacitors to corresponding U3 outputs so that the portamento will slowly follow the control voltage even when it’s turned off, thus reducing or eliminating the sudden change in control voltage when it is switched on.
* Different gain-setting resistors for U3 to get different output voltage range.

Mode switches and clock adjustment

The device has also three switches for controlling the operation mode. These switches are the sequence direction selection switch (on IO11), the sequence length selection switch (on IO12) and the clock source selection switch (on IO8). These are simply toggle switches, pulling the indicated Arduino pins either high (+5V) or low (0V). The software then implements the actual functionality. The default software uses the direction switch to select whether the sequence runs forwards or backwards, the length switch to choose between 4 and 8 step sequence and the clock source switch to choose between the externally supplied and the internally (by software) generated clock.

It should be noted that as mechanical switches, these are prone to contact bounce. In other words: when a switch is flipped, the Arduino is unlikely to see a clean singular low-to-high or high-to-low transition. This is due to mechanical vibrations in the switch itself. Another complication is that when the switch is moving from one position to another, the Arduino input pin is not connected to anywhere and thus its state is badly defined.

The latter problem is reduced by Arduino’s internal pull-up resistors, which are enabled on all the switch pins. Thus, when a switch is not driving a corresponding input high or low, the pull-up resistor will pull it high. However, this does not help much with the contact bounce problem. The software removes the contact bounce by taking samples of the switch state once per millisecond and requires 4 subsequent detections of the same level in order to act by it. An alternative or additional way to filter the switch inputs would be to add filter capacitors to the switch pins.

In addition to the three switches, this block also includes the internal oscillator’s speed adjustment. This is just a potentiometer connected between the 0V and +5V rail. Arduino reads the potentiometer’s voltage and adjusts the software-based oscillator’s speed according to the measurement. If the internal clock option is selected, this oscillator’s output drives the sequencer.

Clock and reset inputs

The device has two gate-level inputs for externally supplied clock and reset pulses. There are also push buttons so that the user can produce these pulses manually. In principle, these inputs aren’t much different from the mode switch inputs and due to the use of manual push buttons, they can also have problems with contact bounce. Like with the mode switches, the actual functionality is implemented in software. These inputs also use similar software filtering as the mode switches.

There are two major differences to mode switch handling, though. The first difference is the diode gating used to combine the external signals with the push-button generated signals. Without diodes, the switches could short the incoming signals, potentially damaging other instruments. The second difference the Arduino input protection: the diodes and resistors limit the negative and positive overvoltages seen by the Arduino input pins. This is needed, because the Eurorack instruments can produce negative output voltages and also outputs in excess of +5V. Another source for overvoltages is static electricity.

Like with the mode control switches, it might be desirable to add input filtering capacitors. Also, since these signals are routed to Arduino inputs capable of measuring analog voltages (ADC inputs), it might make sense to replace the current input filtering code with something more sophisticated. Another simple modification would be the addition of LEDs (with suitable current limiting resistors) to show the current state of the clock and reset inputs.

Power

The instrument has two main options for power input: the power can be supplied via the standard 10-pin Eurorack cable or it can be supplied from a more conventional external power supply. The first option is obviously more desirable when the instrument is mounted to an Eurorack, while the second is much more convenient for stand-alone use.

When using the Eurorack power option, the Eurorack power bus supplies the instrument with +12V, ‑12V and 0V. These are routed directly to the control voltage amplifier/portamento block, as the amplifiers need +10…11V to be able to produce the +8V maximum output and -2…-3V to be able to produce the 0V output.

The stand-alone power supply section takes just single +12V input, as single output power supplies are significantly cheaper and easier to get than dual supplies. The negative supply is then generated using a charge pump ICL7662. This section also includes a fuse and a protection diode against wrong polarity and overvoltage. The two diodes between the Eurorack power connector and the external power input prevent the charge pump from competing with the Eurorack power input. The resistor in charge pump’s output forms a filter with the other power supply capacitors, thus reducing charge pump’s noise.

Most of the instrument’s electronics runs from +5V supply. This produced by Arduino’s internal regulator, taking +12V input. Since the maximum power dissipation of the Arduino regulator is rather limited, it is recommended to keep the power drain from +5V low. That said, Arduino should be able to provide at least 40mA from +12V input and 20mA from +15V input. Most of this current goes to LEDs: the default software always keeps 3 LEDs on, which consume 10mA or so in total. Short-circuited gate outputs can consume additional 5mA.

The charge pump used to produce the -12V supply is quite weak, being limited to -20mA or so. The amplifiers consume about half of that. Shorted control voltage outputs won’t markedly increase the power consumption from the negative rail, but accidentally connecting them to gate or control voltage outputs of the other Eurorack modules can increase the current drain up to 8mA.

Connections for additions

In order to make it easier to connect additional hardware to Simple Sequencer, be it LEDs, switches, connectors or even circuitry, the device has several 8-pin headers arranged on board perimeter. These include, for example, J1, J2 and J69, which bring out the most important Arduino pins. Similar headers are available for the most control voltage potentiometers (J10, J12, J13, J14), which also include power and ground pins.

1. ATmega328P datasheet: <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf> [↑](#footnote-ref-1)
2. Arduino documentation: <https://www.arduino.cc/en/main/docs> [↑](#footnote-ref-2)