PM-909 Build Guide

v3.0.0

The PM-909 is an open source, Arduino based, 16 step DIY drum sequencer. It was designed to control the TR-909 or the TR-808 drum modules, however, it can be used to trigger any set of analog drum modules.

Features:

- Open source
- Triggers a maximum of 16 drum voices
- High & Low accent, adjustable for each voice
- Shuffle and Flam, for each voice
- Pattern Play
- Create, edit and play rhythm tracks
  (48 different patterns [1], 224 measures per track)
- Save upto 36 tracks on an EEPROM
- MIDI in [2]
  - Play a rhythm track that was programmed in a DAW
  - Sync the PM-909 to a DAW
- MIDI out [2]
  - Use the PM-909 as a Master
  - Export a rhythm track into a DAW as a MIDI track [3]
- Create random rhythm patterns
- Clock with internal clock or with external LFO / MIDI
- External reset input

[1] When 11 voices are used. If more voices are used, this value will be lower, if less voices are used, this value can be higher (see below).

[2] MIDI functionality only tested with Ableton Live.

[3] It works quite good but not flawless (yet).

A little on how it works:

The PM-909 sequencer is built around an Arduino Nano Every. The Nano Every has the same pinout as a regular Nano v3, but it has more SRAM memory (6KB instead of 2KB).

Two 74HC595 (8 bit shift registers) are used to generate the 5V triggers. One TLC5940 (16 channel PWM LED driver) (U5) is used to drive the 16 LEDs. A second TLC5940 (U4) is used to generate the Control Voltages (CV) for the accents. The PWM signals on the outputs of U4 are low pass filtered and buffered. By using a high PWM frequency (31,4 kHz), the response time of the fluctuating accent CV is short enough to be usable in a drum sequencer.

Using this setup, the PM-909 sequencer could trigger a maximum of 16 drum modules (provided that they accept 5V triggers and 0-5V for the accent CV). By default, the PM-909 circuit and firmware are configured to control 11 drum modules, just like the original TR-909. However, it is quite easy to adjust the circuit as well as the firmware to work with more (upto 16), or less drum modules.

For the latest schematics and firmware versions:
https://github.com/ScharreSoft/The-Poor-Mans-TR909

Using more or less drum modules (voices)

By default, the PM-909 circuit and firmware are configured to control 11 drum modules. If you want to use another number of drum modules (max. 16), some changes should be made on both the hardware and the firmware.

1: Adjusting the hardware:
If more drum voices are needed, op-amp buffers and low pass filters have to be placed at the remaining outputs of U4. Just copy the circuit from the other outputs. If you want to use less than 11 voices, op-amp buffers can be omitted. See the schematic for more information.

2: Configuring the firmware (PM909_config.h)
By use of a simple configuration file, the firmware can be configured to work with more or less drum modules (voices).

MaxVoices	Defines the number of voices that you want to use	(default = 11)
MaxComposition	Defines the length (in measures) of a drum track	(default = 224)
MaxPatterns	Defines the number of different patterns in a track	(default = 48)

WARNING: When changes are made to these values in the configuration file, the rhythm tracks that were previously saved on the EEPROM will become unusable!

If more drum voices are used, more memory will be needed. So when MaxVoices is increased, it is advised to reduce the value of MaxPatterns accordingly. On the other hand, when less drum voices are needed, the number of MaxPatterns can be increased if desired. Using a too high value for MaxPatterns may lead to program instability.

The following values can be used as a guideline:

MaxVoices	MaxPatterns
8	66
9	59
10	53
11	48
12	44
13	41
14	38
15	35
16	33

It is possible to use other names for the drum voices, if needed. See the instructions in the configuration file (PM909_config.h) on how to change the name of drum voices.

PM-909 < - > PM-808

There are actually two schematics, the PM-909 and the PM-808. There are two main differences between the two, one is the CV for the accent, the other is how the Open and Closed Hihat modules work.

Control Voltage

The PM-909 sequencer is designed to work with drum modules that require 0-5V for the accent CV. This is because the TR-909 drum modules require 0-5V CV. If your drum modules need a CV of 0-15V (for instance, if you are using

TR-808 drum modules) you can use the PM-808 schematic. The op-amp buffers of the PM-808 have a gain of 3, resulting in 0-15V CV.
If you are using a mix of drum modules that need both 0-5V and 0-15V CV, you could use the PM-808 schematic and use voltage dividers for the modules that need only 0-5V CV.

Hi Hat module

The TR-909 Hihat module has one combined trigger pin for the Open and Closed Hihat. The trigger-out pins on the PM-909 sequencer for Open and Closed Hihat are tied together using diodes and connected to this pin. A second pin on the 909 Hihat module is called HIHATSELECT. When it is high, Closed Hihat is selected, when it is low, Open Hihat is selected.

The TR-808 Hihat module does not have a HIHATSELECT pin. Instead, the Open Hihat and the Closed Hihat each have their own trigger pin. So the diodes are not necessary here.

In both the PM-909 and the PM-808, the accent outputs are tied together using diodes. Note that the orientation of the diodes is different in the PM-909 and the PM-808. It will work with 1N4148 diodes, but it is best to use schottky diodes.

The PM-909 firmware works with both the PM-909 and PM-808 schematics. Be sure to make the necessary changes to the configuration file (PM909_config.h);

If you are using the TR-909 Hihat module:
#define TR_909_HIHAT 1
#define TR_808_HIHAT 0

If you are using the TR-808 Hihat module:
#define TR_909_HIHAT 0
#define TR_808_HIHAT 1

If none of the TR-808 or TR-909 hihat modules are used:
#define TR_909_HIHAT 0
#define TR_808_HIHAT 0
In this case, the diodes on the accent pins and the diodes on the trigger pins should be omitted.

EEPROM

By default, the firmware is configured to work with a 1025Kb EEPROM. Smaller EEPROMS can be used (256 Kb or 512 Kb) but you will be able to store less drum tracks, and you will have to adjust the configuration file (PM909_config.h).

Using the default values (11 voices and 48 patterns) the numbers of drum tracks that can be saved are as follows:

EEPROM	Drum tracks
1025 Kbit	36
512 Kbit	18
256 Kbit	9

When less voices or less patterns are used, the number of stored drum tracks can be higher. When more voices/patterns are used, the number will be lower. Use the PM909 Calculation Sheet to calculate the maximal number of drum tracks that can be saved on the EEPROM with the settings of your choise.

NB: the EEPROM has to be I2C compatible.

Compiling the sketch

The sketch makes use of two libraries that should be installed in your Arduino IDE:

a) LiquidCrystal_I2C.h (https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library)
b) Tlc5940_AVR0.h (https://github.com/ScharreSoft/Tlc5940_AVR0)

Two adjustments should be made in the Tlc5940_config.h file. This file can be found in the Documents\Arduino\ libraries\Tlc5940_AVR0\ folder. Open the file and make these two adjustments:

1: We are using 2 TLC5940s in our circuit: set the number of daisy chained TLCs to 2:

```
#define NUM_TLCS 2
```

2: We want to get a high PWM frequency: set the Grayscale Resolution to 511:

```
#define GRAYSCALE_RESOLUTION 511
```

NB: because we are using the Tlc5940_AVR0 library, Arduino pin D12 can not be used as a GPIO.

Optional: Remove the warning from the LiquidCrystal_I2C library

When compiling the LiquidCrystal_I2C library by fdebrabander you will get a warning. When you compile the sketch again, the warning will disappear and it will work just fine. However, if you want to get rid of the warning proceed as follows:

1) Go to the directory of the LiquidCrystal_I2C library. You can find it in the ..\documents\Arduino\libraries folder

2) Open the file LiquidCrystal_I2C.h

3) Locate this section:

```
#define En B00000100  // Enable bit
#define Rw B00000010  // Read/Write bit
#define Rs B00000001  // Register select bit
```

4) Replace it with this:

```
#define En 0b00000100  // Enable bit
#define Rw 0b00000010  // Read/Write bit
#define Rs 0b00000001  // Register select bit
```

5) Save the file.

Now the warning should disappear.

A1		Arduino Nano Every
U1, U2, U3	TL074	Quad Opamp
U4, U5	TLC5940s	16 channel PWM LED driver
U6, U7	74HC595	8 Bit Shift register
U8	24C256	Serial EEPROM 1025 Kb (I2C compatible) (256 Kb or 512 Kb will also work)
U9	6N137	Opto coupler
U10	TL072	Dual Opamp
U11	LM7805	Voltage regulator (5 volt, 1 A)
U12	16x2 LCD I2C	Standard 16x2 LCD screen with an I2C interface

IMPORTANT!!

It is highly advised to use an appropriate heat sink for the LM7805. Depending on the current that your LEDs draw, it can become quite hot.