HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

MIDI to CV module made for 900 yen - modular synth self-made

21

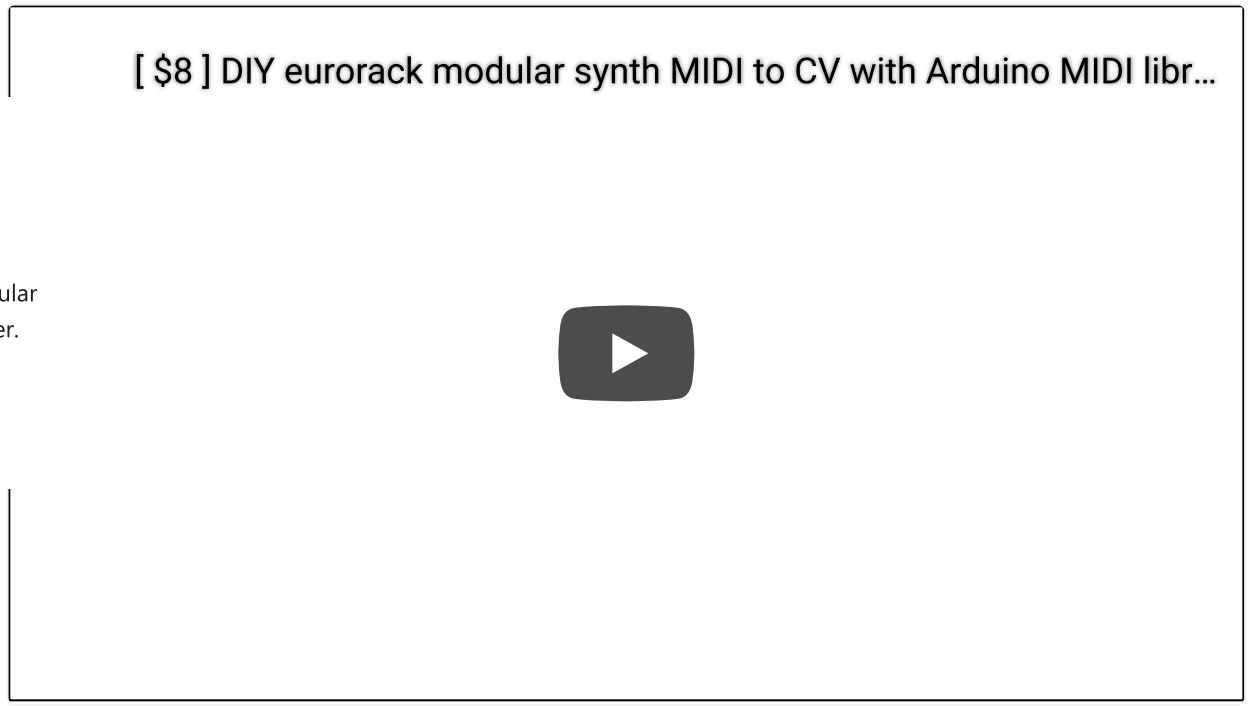
HAGIWO
07:55, 6 March 2021 (UTC)

While challenging Arduino programming, I made my own modular synthesizer MIDI to CV module, so I wrote a memoir of that.

HAGIWO

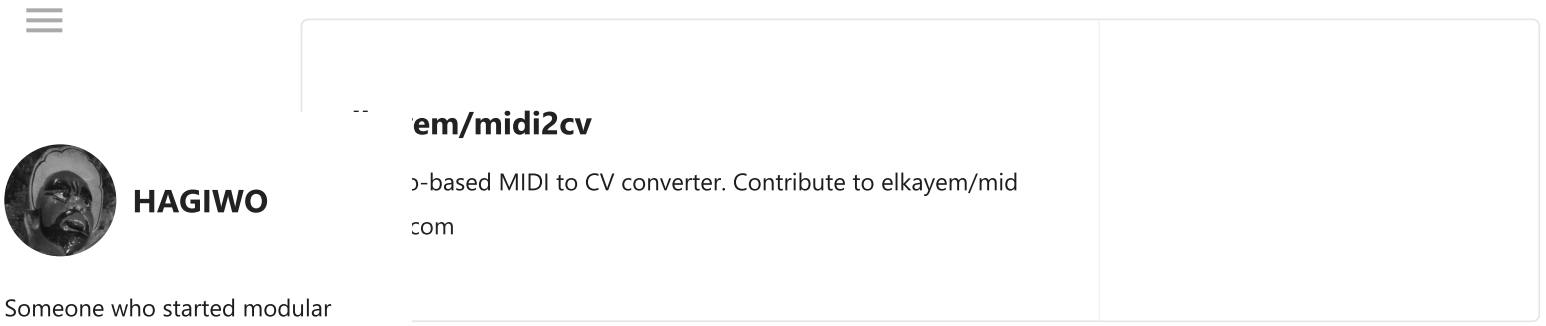
Someone who started modular synths. I'm a legacy engineer.

follow



This is the eleventh work of programming that I started to break away from system engineers who can not write **background**
code. He was planning to create a stand-alone synth using Yamaha's PCM sound source IC, and needed to understand MIDI for that purpose.
To get started, create a MIDI to CV module to better understand MIDI.

There are already great MIDI to CV modules like MIDI2CV released in open source.



HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

er, copying the code alone would not advance the understanding of MIDI, so it was necessary to reinvent the wheel.
While the functionality was heavily inspired by the MIDI2CV, the coding was created by thinking about it on its own as much as possible.

Specs

Eurorack Standard 3U 6HP Size

Power Supply: 39mA (at 5V) / 37mA (at12V)

Can operate on a single 5V power supply. Or can operate from a single 12V supply.

CV: MIDI scale output at 1V/oct. The range is 0 - 5V.

Output 5V only while GATE:key is on.

MOD: Output modulation wheel. 0 - 5V。

CLK: Output MIDI Clock. The knob can be divided from 24, 12, 6,

nd CLK glow in conjunction with LEDs.

utput has overvoltage, negative voltage, and overcurrent protection

y.

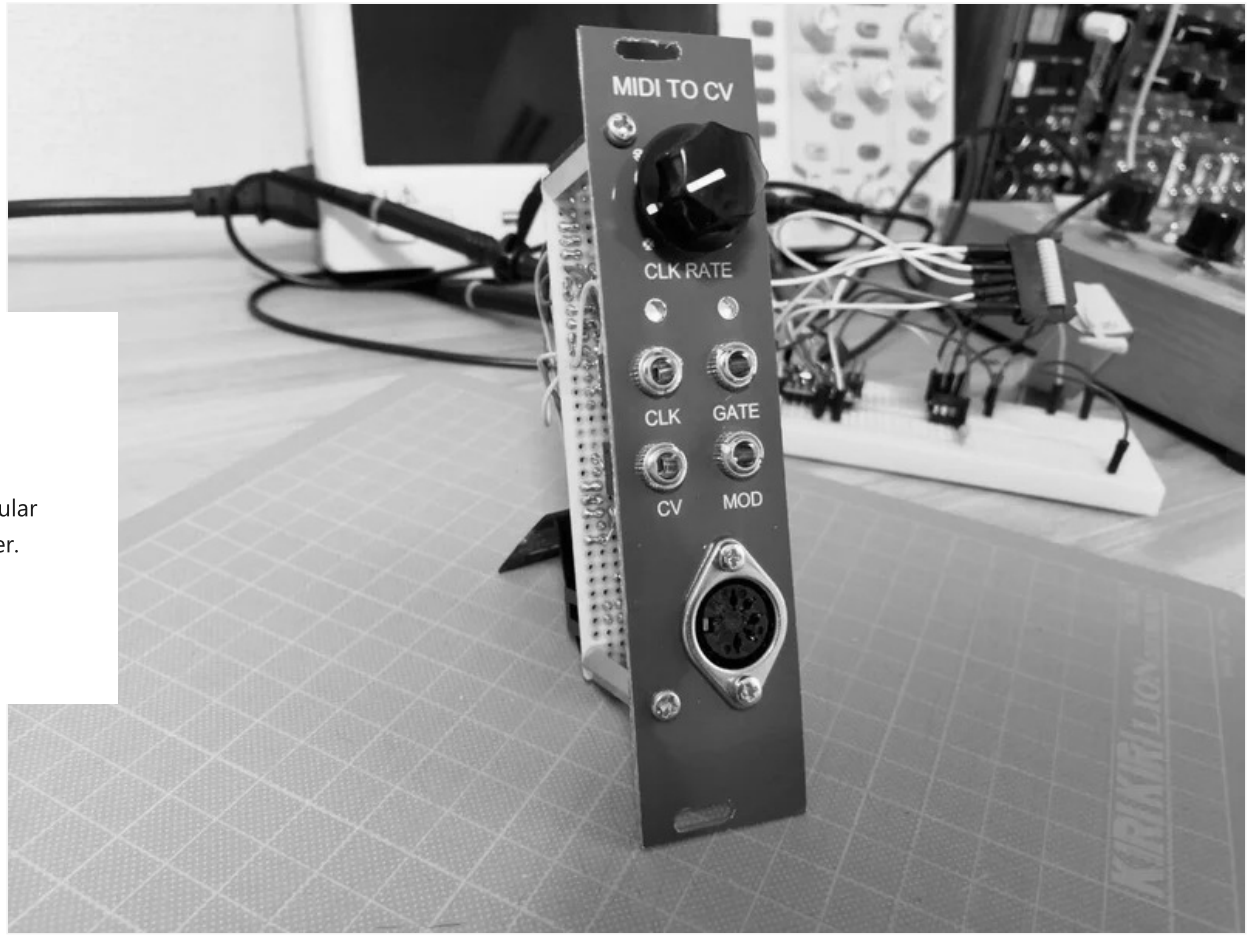
ut accepts only the MIDI signal of CH1, but you can easily change the

diting the source code.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow



HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

The total **production cost**

is less than 900 yen.

Arduino compatible board: 220 yen 12bit 2ch DAC: 250 yen Panel: 150 yen

Optocoupler: 20 yen

The DAC and optocoupler are obtained from Akizuki Electronics.

Other electronic components were obtained from Aliexpress and JLCPCB.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

Program

using the MIDI library of Arduino.

We have devised a way to handle NOTE ON/OFF when multiple keys are

input, but the others are very simple.

The output of the CV has values in a table. It uses a Quantizer program made in the past.

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

ecaution, serial communication fights because it uses the RX terminal.

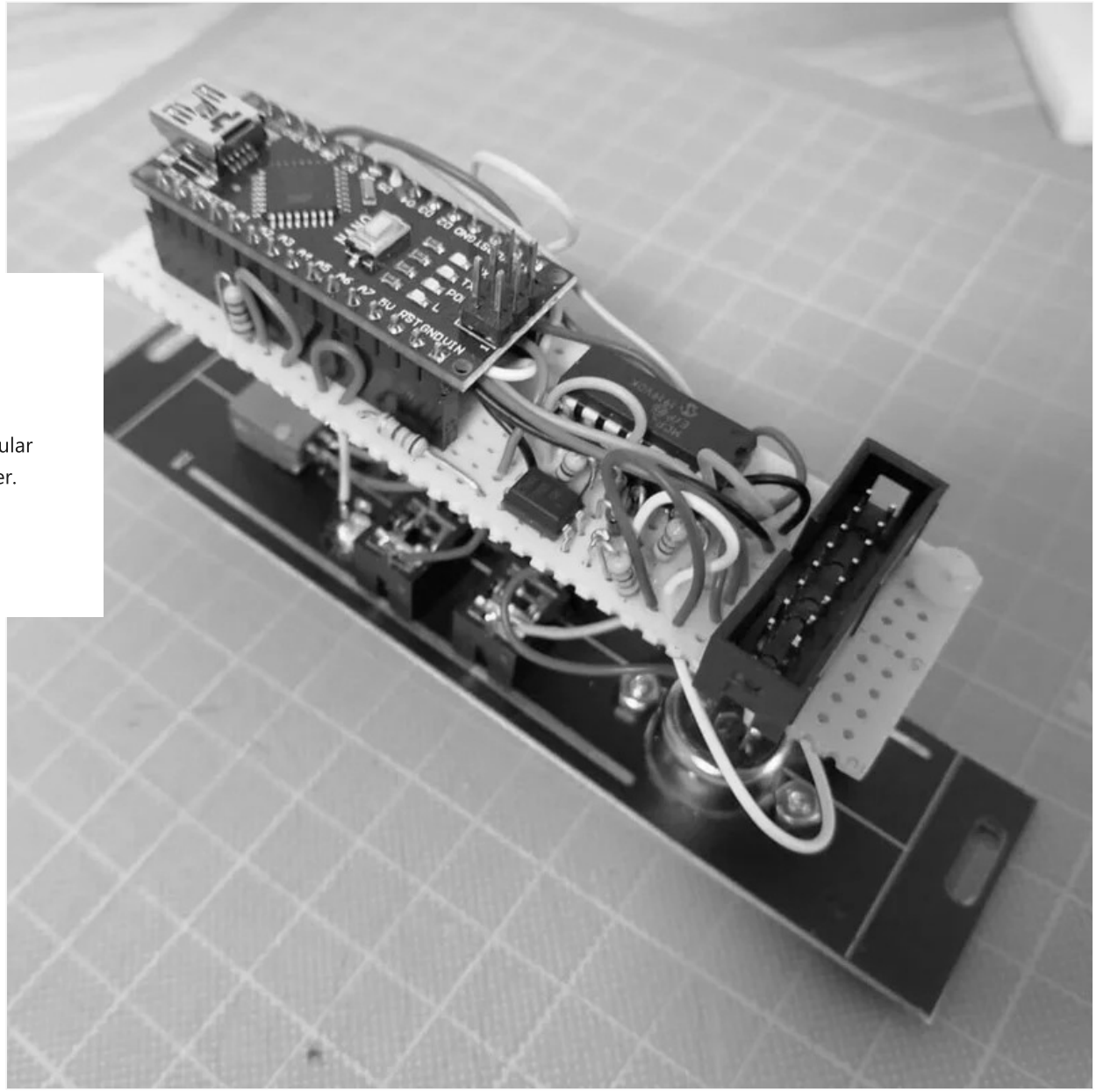
writing a program from a PC to the Arduino, it is necessary to

nect the RX connection on the circuit.

tion, since serial communication with the PC is also not possible, the

nonitor of the Arduino IDE can not be used.

e there's a workaround.)



HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

Since the **hardware**

MIDI input area is standardized, the circuits on the market are copied as it is.

I think there is no problem with the optocoupler using anything. A general optocoupler of about 20 yen is OK.

The output resistance from the DAC was set to 200 Ω. This is to protect the rating of the DAC.

HAGIWO

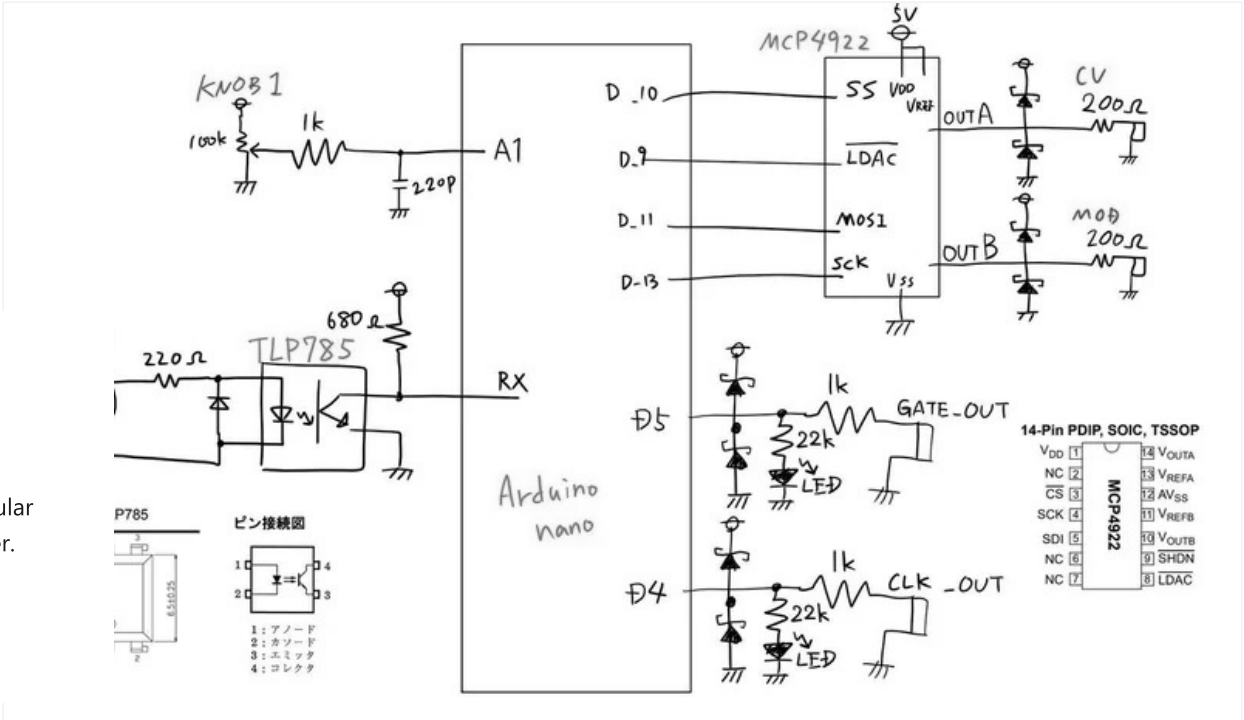
Someone who started modular synths. I'm a legacy engineer.

follow

the output resistance is better at 0 ohms. This is because the pitch is shifted due to partial pressure.

of the modules on the market have no output resistance. Since there is edance on the receiving side, it will not fail unless it is used badly, but if you make a strange connection, it will fail.
As long as it is released as open source, it is not possible to destroy other people's modules, so we try to install protective parts.

ALTHOUGH EURORACK IS STANDARDIZED, THE OUTPUT IMPEDANCE AND INPUT IMPEDANCE ARE NOT UNIFORM. In the case of GATE, the output is often around 1 kohm and the input is around 100 kohm, but the CV is different.



HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

The source code

is poor, but it will be made public. If there is a bad point, it will be a learning experience if you can tell me.

```
#include <MIDI.h>
#include  <SPI.h>//DAC通信用
MIDI_CREATE_DEFAULT_INSTANCE(); //MIDIライブラリを有効

const int LDAC = 9;//SPI trans setting
int note_no = 0;//noteNo=21(A0)～60(A5) total 61,マイナスの値を取るのでint

int bend_range = 0;
int bend_msb = 0;
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
int bend_lsb = 0;
long after_bend_pitch = 0;

byte note_on_count = 0;//複数のノートがONかつ、いずれかのノートがOFFしたときに、最後のノートON
unsigned long trigTimer = 0;//for gate ratch

byte clock_count = 0;
byte clock_max = 24;//clock_max change by knob setting
byte clock_on_time = 0;
    clock_rate = 0;//knob CVin

   V/OCT LSB for DAC
   st long cv[61] = {
     68, 137,  205,  273,  341,  410,  478,  546,  614,  683,  751,
   9,  887,  956,  1024, 1092, 1161, 1229, 1297, 1365, 1434, 1502, 1570,
   38, 1707, 1775, 1843, 1911, 1980, 2048, 2116, 2185, 2253, 2321, 2389,
   58, 2526, 2594, 2662, 2731, 2799, 2867, 2935, 3004, 3072, 3140, 3209,
   77, 3345, 3413, 3482, 3550, 3618, 3686, 3755, 3823, 3891, 3959, 4028, 4095

};

void setup() {
 pinMode(LDAC, OUTPUT) ;//DAC trans
 pinMode(SS, OUTPUT) ;//DAC trans
 pinMode(4, OUTPUT) ;//CLK_OUT
 pinMode(5, OUTPUT) ;//GATE_OUT

 MIDI.begin(1);          // MIDI CH1をlisten

 SPI.begin();
 SPI.setBitOrder(MSBFIRST) ;          // bit order
 SPI.setClockDivider(SPI_CLOCK_DIV4) ;// クロック(CLK)をシステムクロックの1/4で使用(16MHz/4)
 SPI.setDataMode(SPI_MODE0) ;         // クロック極性０(LOW)  クロック位相０
 delay(50);
}

void loop() {

 //-----------------------------clock_rate setting---------------------------
 clock_rate = analogRead(1);//read knob voltage
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
if (clock_rate < 256) {
  clock_max = 24;//slow
}
else if (clock_rate < 512 && clock_rate >= 256) {
  clock_max = 12;
}
else if (clock_rate < 768 && clock_rate >= 512) {
  clock_max = 6;
}
  se if (clock_rate >= 768) {
  clock_max = 3;//fast


  ----------------------------gate ratch----------------------------
   (note_on_count != 0) {
  if ((millis() - trigTimer <= 20) && (millis() - trigTimer > 10)) {
    digitalWrite(5, LOW);
  }
  if ((trigTimer > 0) && (millis() - trigTimer > 20)) {
    digitalWrite(5, HIGH);
  }
}

//----------------------------midi operation----------------------------
if (MIDI.read()) {                // チャンネル1に信号が入ってきたら
  MIDI.setInputChannel(1);
  switch (MIDI.getType()) {

    case midi::NoteOn://NoteOnしたら
      note_on_count ++;
      trigTimer = millis();
      note_no = MIDI.getData1() - 21 ;//note number

      if (note_no < 0) {
        note_no = 0;
      }
      else if (note_no >= 61) {
        note_no = 60;
      }

      digitalWrite(5, HIGH); //GateをHIGH
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
  OUT_CV(cv[note_no]);//V/OCT LSB for DACを参照
  break;


case midi::NoteOff://NoteOffしたら
  note_on_count --;
  if (note_on_count == 0) {
    digitalWrite(5, LOW); //GateをLOW
  }
  break;


case midi::ControlChange:
  OUT_MOD( MIDI.getData2() << 5); //0-4095
  break;


case midi::Clock:
  clock_count ++;

  if (clock_count >= clock_max) {
    clock_count = 0;
  }

  if (clock_count == 1) {
    digitalWrite(4, HIGH);
  }
  else if (clock_count != 1) {
    digitalWrite(4, LOW);
  }
  break;


case midi::Stop:
  clock_count = 0;
  digitalWrite(5, LOW); //GateをLOW
  break;


case midi::PitchBend:
  bend_lsb = MIDI.getData1();//LSB
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow

```
      bend_msb = MIDI.getData2();//MSB
      bend_range = bend_msb; //0 to 127

      if (bend_range > 64) {
        after_bend_pitch = cv[note_no] + cv[note_no] * (bend_range - 64) * 4 / 10000;
        OUT_CV(after_bend_pitch);
      }

      else if (bend_range < 64) {
        after_bend_pitch = cv[note_no] - cv[note_no] * (64 - bend_range) * 4 / 10000;
        OUT_CV(after_bend_pitch);
      }
      break;


  }
```

```
//DAC_CV output
void OUT_CV(int cv) {
 digitalWrite(LDAC, HIGH) ;
 digitalWrite(SS, LOW) ;
 SPI.transfer((cv >> 8) | 0x30) ;   // H0x30=OUTA/1x
 SPI.transfer(cv & 0xff) ;
 digitalWrite(SS, HIGH) ;
 digitalWrite(LDAC, LOW) ;
}

//DAC_MOD output
void OUT_MOD(int mod) {
 digitalWrite(LDAC, HIGH) ;
 digitalWrite(SS, LOW) ;
 SPI.transfer((mod >> 8) | 0xB0) ;   // H0xB0=OUTB/1x
 SPI.transfer(mod & 0xff) ;
 digitalWrite(SS, HIGH) ;
 digitalWrite(LDAC, LOW) ;
}
```

HAGIWO

Someone who started modular synths. I'm a legacy engineer.

follow