# FM Minimal program

```
#include <OPA.h>
OPA opa;
void setup() {
  opa.enable();
}
void loop() {
  opa.noteOn(OPA_PROGRAM_0, 60);
  delay(1000);
  opa.noteOff(OPA_PROGRAM_0, 60);
  delay(1000);
  }
}
```

# Kit Minimal program

```
#include <OPA.h>
OPA opa;
void setup() {
  opa.enable();
}
void loop() {
  for(int i=0; i < 32; i++){
    opa.noteOn(OPA_PROGRAM_DRUMS, 60 + i);
    delay(250);
  }
}
```

# Program parameters

| | |
|---|---|
| 00: OPA_PROGRAM_NAME | Program ASCII name (8 characters) |
| 08: OPA_PROGRAM_ALGORITHM | Carrier modulator routing |
| 09: OPA_PROGRAM_VOLUME | Master level |
| 10: OPA_PROGRAM_PANNING | Position in stereo field |

# Operators parameters

| | |
|---|---|
| 00: OPA_OP_VOLUME | Output or modulation level |
| 01: OPA_OP_COARSE | Coarse pitch (semitones) |
| 02: OPA_OP_FINE | Fine pitch (128th of semi) |
| 03: OPA_OP_ENVATTACK | Attack time |
| 04: OPA_OP_ENVDECAY | Decay time |
| 05: OPA_OP_ENVSUSTAINLEVEL | Sustain level |
| 06: OPA_OP_ENVINITLEVEL | Initial level |
| 07: OPA_OP_ENVRELEASE | Release time |
| 08: OPA_OP_LFOSPEED | Modulation speed |
| 09: OPA_OP_LFOAMOUNT | Modulation intensity |
| 10: OPA_OP_FEEDBACK | Feedback level (only op. 4) |
| 11: OPA_OP_FLAGS | Additionnal flags |

# Kit parameters

| | |
|---|---|
| 00: OPA_SAMPLE_VOLUME | Output level |
| 01: OPA_SAMPLE_PANNING | Stereo panning |
| 02: OPA_SAMPLE_DECAY | Decay time |

OPA Shield Cheatsheet V1.0 21.07.2016 (c) Fred's LAB http://fredslab.net/opa

# Kit Samples by Loopmasters.com

| | |
|---|---|
| 00: AnKick | 16: Ride 1 |
| 01: AnRimshot | 17: Ride 2 |
| 02: AnSnare | 18: GB Perc 1 |
| 03: AnClaps | 19: GB Perc 2 |
| 04: AnCloseHH | 20: Hit 1 |
| 05: AnOpenHH | 21: Hit 2 |
| 06: AnMaracas | 22: Shaker |
| 07: AnCowbell | 23: Agogo |
| 08: AcKick | 24: Bougaraboo High |
| 09: AcRimshot | 25: Bougaraboo Low |
| 10: AcSnare | 26: Tabla High |
| 11: AcSticks | 27: Tabla Low |
| 12: Closed hithat | 28: Conga High |
| 13: Opened hithat | 29: Conga Low |
| 14: Mid tom | 30: Cabassa |
| 15: Low tom | 31: Egg |

# Interface

57600 bauds, 8-bit serial, 1 stop bit, no parity
RX on Digital 0, TX on Digital 1,
CS1 on Digital 2, CS2 on Digital 3
SWAP Digital 4, RESET Digital 7
Compatible with both 5V or 3V3 logic

# Internel links

**Project website:**
http://fredslab.net/opa

**OPA Editor sources:**
https://github.com/Marzac/OPA-Editor

**OPA Arduino library:**
https://github.com/Marzac/OPA-Library

**Arduino IDE**
https://www.arduino.cc/en/Main/Software

OPA Shield Cheatsheet V1.0 21.07.2016 (c) Fred's LAB http://fredslab.net/opa

# How to sequence music from a computer

OPA can be used as an external MIDI compatible synthesizer to play back songs composed on any kind of computer.

The OPA shield itself is not a MIDI instrument so it needs an interface. For PC like platforms (running *Windows*, *Mac Os* or *Linux*) the **OPA Editor** allows in-depth program parameters edition and provides the communication interface.

Installation steps are:

1) Install the **OPA Editor** for your operating system.
2) Install the **Arduino IDE** (software).
3) Insert your **OPA Shield** on your Arduino board and connect it to your computer using a USB cable.
4) Transfer the **EditorBridge** sketch to your Arduino. (Sketch can be found in the **OPA Editor** archive)
5) For *Windows only,* install a virtual MIDI port driver such as **loopMIDI** or **virtualMIDI.** Create a virtual MIDI port and name it OPA.
5) For *MAC only*, go to **Audio MIDI Setup application** and create a MIDI bus for the **OPA Editor**.
5) For *Linux only*, use a suitable **ALSA** configuration tool to create a virtual MIDI port.
6) Launch the **OPA-Editor.**

In the **OPA-Editor > Connection menu**, select the *serial port* corresponding to your Arduino board and the appropriate virtual MIDI port to receive MIDI messages.

Use the virtual MIDI port in your favorite sequencer!

OPA Shield Cheatsheet V1.0 21.07.2016 (c) Fred's LAB http://fredslab.net/opa

# How to use sensors to control parameters

One of the **OPA Shield** strengths is to be controllable by the Arduino board. Program parameters can be modified, in realtime, within your sketches using inputs from various sensors.

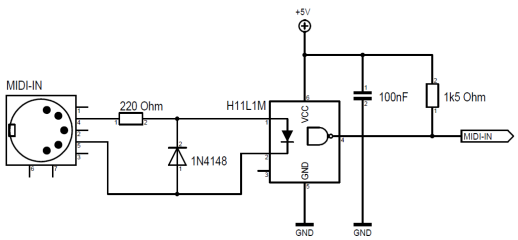
```
#include <OPA.h>
OPA opa;
void setup() {
  opa.enable();
  opa.noteOn(OPA_PROGRAM_0, 60);
}
```

```
void loop() {
  int sensor1 = analogRead(0) >> 2;
  int sensor2 = analogRead(1) >> 2;

  opa.writeOperatorParam(
    OPA_PROGRAM_0, OPA_OPERATOR_0,
    OPA_OP_COARSE, sensor1);
  opa.writeOperatorParam(
    OPA_PROGRAM_0, OPA_OPERATOR_0,
    OPA_OP_VOLUME, sensor2);
}
```

# How to attach a MIDI DIN connector

To use the **OPA Shield** with MIDI controllers or hardware sequencers, you need to build a tiny circuit.




This circuit isolates signals and allows proper connection between MIDI compatible gear and Arduino based projects. The Arduino sketch needs to process MIDI messages to send commands to the **OPA Shield.** Examples can be found in the OPA Arduino Library subfolders.