



FM Shield Reference Manual

OPA Shield reference manual

Revision 0.1 - 24/01/2016

Copyright Frédéric Meslin, Thomas Hopper 2015-2016

This reference document covers all the features provided by the OPA Shield and its dedicated computer-side editor. You will find details about the sound engine, the serial communication protocol, the shield pinout, the internal block-diagram and mechanical characteristics.

All commands and data structures to read and write program parameters, global parameters and manage the internal memory are included.

We've done our best to ensure this document is error free. If you find an error, please let us know via email: fred@fredslab.net

This device has been manufactured respecting the European ROHS directives.

fredslab.net - Product team

- Product concept: Frédéric Meslin & Thomas Hopper
- Hardware / software: Frédéric Meslin
- Product-design / Artwork: Thomas Hopper
- Documentation: Frédéric Meslin & Thomas Hopper
- Beta-testing: Serge Joliprès, Pierre Estève, Benoit Ruelle, Florent Touchard

Our website:

www.fredslab.net/opa

Disclaimer

Fredslab.net cannot be liable for any erroneous information contained in this manual.

This manual, its content and the product specifications may be updated at any time without prior notice. You are free to distribute this manual or any portion of it without our consent.

To ensure your personal safety and to make proper use of your electronic product, please respect the following instructions:

- Use the device indoors only, in a dry and cool atmosphere
- Do not leave the device unattended for longer periods of use
- Do not expose the unit to extreme heat / moisture / radiation or vibration sources
- Do not spill water or other liquids on the product circuitry
- Do not try to modify the product circuitry
- Do not expose the unit to heavy electrostatic discharges
- Store the device in an electrostatic bag when not in use
- Always respect the power ratings

This shield, connected to a power amplifier, speakers or headphones, can produce extremely loud sounds that may cause irreparable damage to human hearing. Therefore sound volume should always be moderated.

This product has been designed to generate audio range frequencies, for musical / sound generation applications. Any other use is not covered by the product warranty. Damage due to non-respect of the safety / proper use instructions are also excluded from the warranty.

Connections

The OPA Shield is designed to be mounted on top of original Arduino / Genuino boards having a specific pin-header connector. Other boards compatible with Arduino might be able to host an OPA Shield but mechanical and electrical specifications must be checked carefully.

The OPA shield is powered by the Arduino bus and requires approximately 5V, 100mA

Serial communication information

To generate the sound, the OPA Shield embedded a powerful 16-bit microcontroller with DSP extensions. The Arduino board communicates with it using the serial interface (UART) and two additional lines, CS1 (DIG2) and SWAP (DIG4).

Prior to any transaction, the serial interface must be configured as follow:

- Baudrate: 115200 bauds
- Packet size: 8 bits
- Parity: none
- Stop bits: one

By default, the OPA Shield **will only** receive messages from Arduino when CS1 (DIG2) is set to low. This is meant to prevent OPA receiving unattended messages when the serial interface is used for another purpose (ie: upload of an Arduino sketch / communication with another module).

The SWAP (DIG4) is meant to internally swap the RX (DIG0) and TX (DIG1) pins for direct communication with a computer. When SWAP (DIG4) pin is high, the Arduino can be used as a bridge, relaying data from the Arduino USB port to the OPA Shield. In this mode, a special sketch needs to be uploaded to the Arduino that ensure the RX (DIG0) and TX (DIG1) pins are set to high-impedance.

Multiple shields configuration - optional

OPA Shield provides an additional chip select CS2 (DIG3) to address up to 3 stacked shields, making them a capable 36 voices sound-generator. Shields internal address need to be set using the on-board jumpers AD1 and AD2.

| Jumpers | AD1 (J1) - unconnected | AD1 (J1) - connected |
|------------------------|------------------------|----------------------|
| AD2 (J2) - unconnected | Address = 0 & 2 | Address = 1 |
| AD2 (J2) - connected | Address = 2 | Address = 0 |

OPA Shield internal address is composed of [AD2, AD1]. This address must match the CS [CS2 (DIG3), CS1 (DIG2)] address for the shield to communicate. By default the OPA shield communicates with both address 0 & 2 making the state of CS2 (DIG3) irrelevant.

The address 3 is reserved.

Voices allocation system

OPA can play up to 10 notes simultaneously. Notes can be produced by any of the 8 programs loaded in memory. If an additional note is triggered, the oldest allocated voice is reused to produce the new note. This so-called “voice-stealing” mechanism allows the playback of complex pieces of music with limited hardware resources. The indicator (LED2) flashes upon voice-stealing.

In certain cases, stealing voices can produce audible clicks. To avoid these, either:

- Disable stealing using global parameters or
- Reduce the tail (enveloppe release time) of the carrier operators or
- Trigger fewer notes at the same time

Audio output














OPA Shield has a 3.5 mm output jack that produces a stereo line level signal. This output can be directly attached to a mixing desk, an amplifier, a computer soundcard or the input of another instrument or sound processing device. The output is not suitable to drive speakers or headphones.

Using OPA with an Arduino board connected via USB might result in ground-loops or noise from power supply problems. Power supply noise can be reduced using an external power adapter. Issues caused by ground loops can be reduced with proper wiring or connecting the Arduino board through a USB galvanic isolator.

Sound engine

The OPA sound engine is an highly optimised 10 voices polyphonic and multi-timbral synthesizer. Synthesis technology is based on phase modulation of sine-wave oscillators, commonly referred as FM-synthesis. The engine can play notes from 8 programs simultaneously.

A single program consist of 4 operators, connected according to an algorithm, and two general parameters. The engine offers 13 different algorithms which are:

| | | | |
|---|---|---|--|
|  | Algorithm 1 Single carrier 1 All modulators in chain |  | Algorithm 8 Two carriers 1 and 2 Operator 4 and 3 modulates 1 and 2 |
|  | Algorithm 2 Single carrier 1 Operators 3 and 4 modulates 2 Operator 2 modulates 1 |  | Algorithm 9 Two carriers 1 and 2 Operator 4 modulates 1 Operator 3 modulates 2 |
|  | Algorithm 3 Single carrier 1 Operator 3 modulates 2 Operators 4 and 2 modulates 1 |  | Algorithm 10 Three carriers 1, 2 and 3 Operator 4 modulates 1 |
|  | Algorithm 4 Single carrier 1 Operator 4 modulates 2 and 3 Operators 3 and 2 modulates 1 |  | Algorithm 11 Three carriers 1, 2 and 3 Operator 4 modulates 1 and 2 |
|  | Algorithm 5 Two carriers 1 and 2 Operator 4 modulates 3 Operator 3 modulates 2 |  | Algorithm 12 Three carriers 1, 2 and 3 Operator 4 modulates 1, 2 and 3 |
|  | Algorithm 6 Two carriers 1 and 2 Operator 4 modulates 1 and 3 Operator 3 modulates 2 |  | Algorithm 13 Four carriers 1, 2, 3 and 4 |
|  | Algorithm 7 Two carriers 1 and 2 Operator 4 modulates 3 Operator 3 modulates 1 and 2 | | |

The additional mixing related parameters are:

- Program volume
Set the overall volume of the notes produced by the program
- Program panning
Set the stereo panning of the notes produced by the program

Programs computer editor

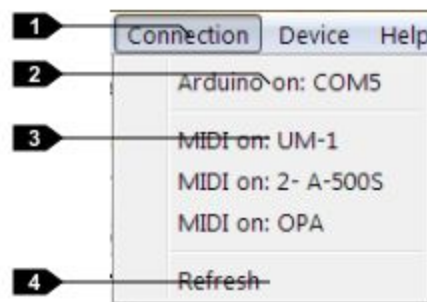
A multi-platform editor has been developed to ease the creation of programs for the OPA. Programs can also be generated by an Arduino sketch. Latest version of the OPA Editor can be downloaded for free from our website:

www.fredslab.net/opa

Editor setup

To use the editor, the Arduino board hosting the OPA Shield must be attached to the computer with an USB cable. Please refer to the Arduino documentation for the complete connection procedure and eventual driver downloads.

Once the Arduino is attached to the computer, it shows up in the editor connection menu:

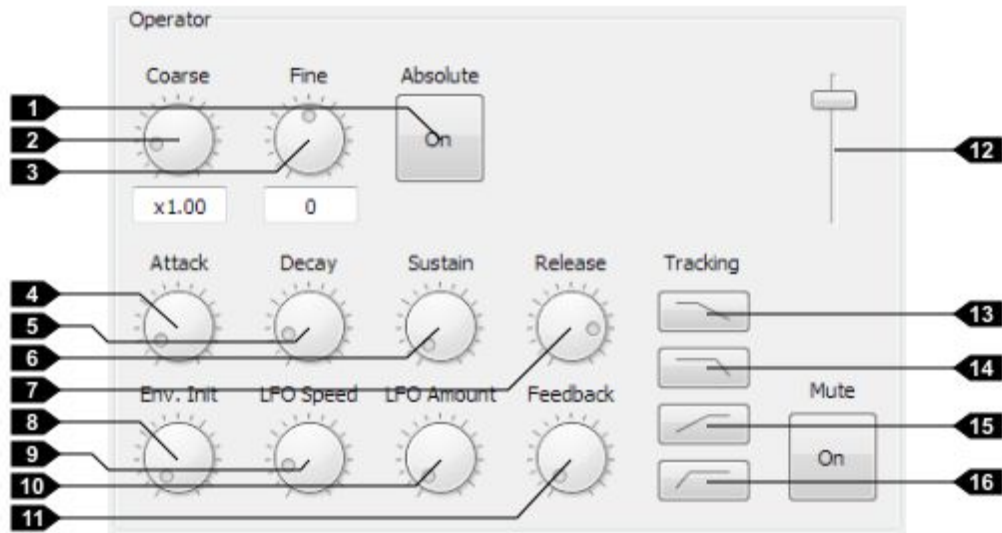


| | | | |
|---|------------------------|----|------------------|
| 1 | Connection menu | 3 | MIDI inputs list |
| 2 | Serial interfaces list | 10 | Refresh action |

To establish the communication between the editor and the OPA Shield, select the serial port that corresponds to your Arduino board. It is also recommended to choose a MIDI input device such as a keyboard, drum pad ... to send note events to OPA.

Configuration of operators

Each operator parameter can be configured using the controls located in operator frames.



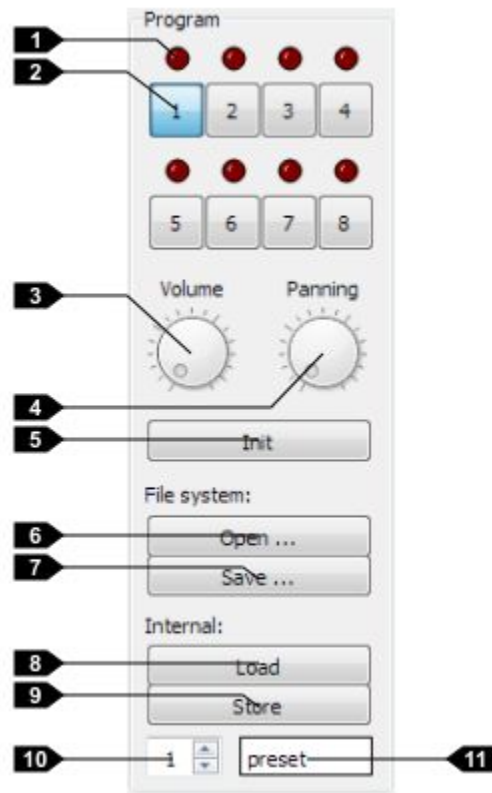
| | | | |
|---|-------------------------------------|----|-----------------------------|
| 1 | Absolute mode - on / off | 9 | LFO speed |
| 2 | Coarse tuning - ratios or semitones | 10 | LFO amount |
| 3 | Fine tuning - 1/128th of a semitone | 11 | Feedback amount |
| 4 | Envelope attack time | 12 | Operator volume |
| 5 | Envelope decay time | 13 | High pitch soft slope curve |
| 6 | Envelope sustain level | 14 | High pitch hard slope curve |
| 7 | Envelope release time | 15 | Low pitch soft slope curve |
| 8 | Envelope initial level | 16 | Low pitch hard slope curve |

Remarks:

- Envelope times (attack, decay and release) ranges from 1ms to 16s
- LFO speeds ranges from 16s to 16Hz
- Mute control is an editor function and not associated with an operator parameter
- Feedback amount is only available for operator 4

Configuration of programs

Programs can be configured using the controls located in the right-sided program frame.



| | | | |
|---|------------------------------------|----|-------------------------------------|
| 1 | Program activity indicators | 7 | Save a program to the hard-drive |
| 2 | Currently edited program buttons | 8 | Load a program from internal memory |
| 3 | Program volume | 9 | Store a program in internal memory |
| 4 | Program stereo panning | 10 | Internal memory program number |
| 5 | Initialize program button | 11 | Internal memory program name |
| 6 | Open a program from the hard-drive | | |

Available commands

0 : Retrieve Version

Retrieve the version information of the OPA Shield.

| Byte | Value | Description |
|------|-------|----------------------|
| 0 | 00 | Retrieve status code |

This command will generate a reply message using the following format:

OPA rV.vv yyyy.mm.dd\r\n

With:

- V: major revision number
- v: minor revision number
- yyyy: release year
- mm: release month
- dd: release day

Remark: the replied string is terminated with *Carriage return* (\r) and *Line feed* (\n) characters.

1 : Note On

Plays a note by the specified program.

Note numbers correspond to an extended MIDI scale that ranges from 0 to 255.

Note 0 is C-5 / 8.1758 Hz - Note 255 is G# / 23.679 kHz (with Global tuning set to 0).

| Byte | Value | Description |
|------|----------|--------------------|
| 0 | 01 | Note on code |
| 1 | 0 to 7 | Program |
| 2 | 0 to 255 | Pitch in semitones |
| 3 | - | Reserved |

2 : Note Off

Release a note played by the specified program.
Note numbers correspond to standard MIDI notes scale.

| Byte | Value | Description |
|------|----------|--------------------|
| 0 | 02 | Note off code |
| 1 | 0 to 7 | Program |
| 2 | 0 to 255 | Pitch in semitones |
| 3 | - | Reserved |

3 : All Notes Off

Release all notes played by the specified program.

| Byte | Value | Description |
|------|-----------------|---------------------------------|
| 0 | 03 | All notes off code |
| 1 | 0 to 7 / 255 | Program or 255 for all-programs |

4 : All Sounds Off

Mute immediately all notes played.

| Byte | Value | Description |
|------|-------|----------------|
| 0 | 04 | All sounds off |

5 : Parameter Write

This message is used to configure the sound engine, it serves the following purposes:

- Write a specific parameter to one of the 8 programs given its number.

- Write a specific parameter to one of the global parameters using the program 254.
- Write a specific parameter to all programs at the same time using the program 255.

| Byte | Value | Description |
|------|--------------------------------------|--|
| 0 | 05 | Parameter write code |
| 1 | 0 to 7 / 254 / 255 | Program, 254 for global parameters or 255 for all programs |
| 2 | 0 to 67 (program) 0 to 7 (global) | Parameter index |
| 3 | 0 to 255 | Parameter value |
| 4 | - | Reserved |

6 : Parameter Read

This message is used to read the configuration of the sound engine, it serves the following purposes:

- Read a specific parameter from one of the 8 programs given its number.
- Read a specific parameter from the global parameters using the program 254.

| Byte | Value | Description |
|------|--------------|--------------------------------------|
| 0 | 06 | Parameter read code |
| 1 | 0 to 7 / 254 | Program or 254 for global parameters |
| 2 | | Parameter index |

This command will generate a reply message identical to the respective *Parameter Write* message.

7 : Program Write

Write a complete program in main memory given its number.

| Byte | Value | Description |
|------|--------|--------------------|
| 0 | 07 | Program write code |
| 1 | 0 to 7 | Program |

| | | |
|----|----------|----------------------|
| 2 | 68 | Program length |
| 3 | - | Reserved |
| 4 | 0 to 255 | Program data byte 0 |
| 5 | 0 to 255 | Program data byte 1 |
| | | ... |
| 72 | | Program data byte 67 |

8 : Program Read

Read a complete program from main memory given its number.

| Byte | Value | Description |
|------|-------|-------------------|
| 0 | 08 | Program read code |
| 1 | | Program length |

This command will generate a reply message identical to the respective *Program Write* message.

9 : Internal Memory Program Store

Burn a single program from main memory to internal user memory.

Memory protection in *Global parameters* must be disabled prior to program storing.

Remark: Storing a program in internal memory will destroy any previously stored content.

| Byte | Value | Description |
|------|---------|------------------------------------|
| 0 | 09 | Internal memory program store code |
| 1 | 0 to 7 | Program |
| 2 | 0 to 89 | Internal memory program |

10 : Internal Memory Program Load

Recall a single program from internal user memory to main memory.

| Byte | Value | Description |
|------|---------|-----------------------------------|
| 0 | 10 | Internal memory program load code |
| 1 | 0 to 5 | Program |
| 2 | 0 to 89 | Internal memory program |

11 : Internal Memory Program Write

Write a program directly to internal user memory.

| Byte | Value | Description |
|------|----------|-----------------------------------|
| 0 | 11 | Internal memory program read code |
| 1 | 0 to 89 | Internal memory program |
| 2 | 68 | Program length |
| 3 | - | Reserved |
| 4 | 0 to 255 | Program data byte 0 |
| 5 | 0 to 255 | Program data byte 1 |
| | | ... |
| 72 | | Program data byte 67 |

This command will generate a reply message identical to the respective *Program Write* message.

12 : Internal Memory Program Read

Read a program from internal user memory.

| Byte | Value | Description |
|------|---------|-----------------------------------|
| 0 | 12 | Internal memory program read code |
| 1 | 0 to 5 | Program |
| 2 | 0 to 89 | Internal memory program |

This command will generate a reply message identical to the respective *Program Write In Internal Memory* message.

13 : Pitch-bend

Modulate the pitch of a note being played.

| Byte | Value | Description |
|------|--------------|---|
| 0 | 13 | Pitch bend code |
| 1 | 0 to 7 / 255 | Program or 255 for all programs |
| 2 | -128 to 128 | Pitch bend coarse, in semitones |
| 3 | -128 to 128 | Pitch bend fine, in 1/128th of semitone |

Program parameters list

| Operator structure | | | | Size = 16 bytes |
|------------------------|-------|--|----------------------|-----------------|
| Parameter | Index | | Parameter | Index |
| Program algorithm | 0 | | | |
| Program volume | 2 | | | |
| Program stereo panning | 3 | | | |
| | | | | |
| OP1 volume | 4 | | OP2 volume | 20 |
| OP1 coarse | 5 | | OP2 coarse | 21 |
| OP1 fine | 6 | | OP2 fine | 22 |
| OP1 envelope attack | 7 | | OP2 envelope attack | 23 |
| OP1 envelope decay | 8 | | OP2 envelope decay | 24 |
| OP1 envelope sustain | 9 | | OP2 envelope sustain | 25 |
| OP1 envelope init | 10 | | OP2 envelope init | 26 |
| OP1 envelope release | 11 | | OP2 envelope release | 27 |
| OP1 LFO speed | 12 | | OP2 LFO speed | 28 |
| OP1 LFO amount | 13 | | OP2 LFO amount | 29 |
| OP1 feedback | 14 | | OP2 feedback | 30 |
| OP1 flags | 15 | | OP2 flags | 31 |
| | | | | |
| OP3 volume | 36 | | OP4 volume | 52 |
| OP3 coarse | 37 | | OP4 coarse | 53 |
| OP3 fine | 38 | | OP4 fine | 54 |
| OP3 envelope attack | 39 | | OP4 envelope attack | 55 |

| | | | | |
|----------------------|----|--|----------------------|----|
| OP3 envelope decay | 40 | | OP4 envelope decay | 56 |
| OP3 envelope sustain | 41 | | OP4 envelope sustain | 57 |
| OP3 envelope init | 42 | | OP4 envelope init | 58 |
| OP3 envelope release | 43 | | OP4 envelope release | 59 |
| OP3 LFO speed | 44 | | OP4 LFO speed | 60 |
| OP3 LFO amount | 45 | | OP4 LFO amount | 61 |
| OP3 feedback | 46 | | OP4 feedback | 62 |
| OP3 flags | 47 | | OP4 flags | 63 |

Global parameters (binary structure)

| Global structure | | | Size = 8 bytes |
|------------------|---------------|-------------|--|
| | Byte position | Range | Description |
| volume | 0 | 0 to 255 | Master shield volume |
| coarse | 1 | -128 to 127 | Master transpose, in semitones |
| fine | 2 | -128 to 127 | Master fine tune, in 1/128th of semitone |
| flags | 3 | 0 to 255 | Global flags, refer to dedicated table |
| reserved 1 | 4 | - | Unused |
| reserved 2 | 5 | - | Unused |
| reserved 3 | 6 | - | Unused |
| reserved 4 | 7 | - | Unused |

| Global flags (OR combinaison) | | |
|-------------------------------|----------|--|
| | Bit mask | Description |
| stealing mode | 1 | Voice stealing mode 0 disabled / 1 enabled |
| memory protection | 2 | Internal memory protection 0 disabled / 1 enabled |
| reserved 1 | 4 | Unused |
| reserved 2 | 8 | Unused |
| reserved 3 | 16 | Unused |
| reserved 4 | 32 | Unused |
| reserved 5 | 64 | Unused |

| | | |
|------------|-----|--------|
| reserved 6 | 128 | Unused |
|------------|-----|--------|

Program parameters (binary structure)

| Program structure | | | Size = 68 bytes |
|-------------------|---------------|----------|--|
| | Byte position | Range | Description |
| algorithm | 0 | 0 to 13 | Program algorithm / operators configuration |
| reserved | 1 | | |
| volume | 2 | 0 to 255 | Program volume |
| panning | 3 | 0 to 255 | Program stereo panning 0 is left, 128 is center and 255 right |
| operator 1 | 4 | - | Operator 1 configuration Refer to dedicated table |
| operator 2 | 20 | - | Operator 2 configuration Refer to dedicated table |
| operator 3 | 36 | - | Operator 3 configuration Refer to dedicated table |
| operator 4 | 52 | - | Operator 4 configuration Refer to dedicated table |

Operator parameters (binary structure)

| Operator structure | | | Size = 16 bytes |
|--------------------|---------------|-------------|--|
| | Byte position | Range | Description |
| volume | 0 | 0 to 255 | Operator volume |
| coarse | 1 | 0 to 255 | Relative mode: Operator pitch ratio Absolute mode: Operator absolute pitch (semitones in MIDI scale) |
| fine | 2 | -128 to 127 | Operator fine tune, in 1/128th of semitone |
| envelope attack | 3 | 0 to 255 | Volume envelope attack time |
| envelope decay | 4 | 0 to 255 | Volume envelope decay time |
| envelope sustain | 5 | 0 to 255 | Volume envelope sustain level |
| envelope init | 6 | 0 to 255 | Volume envelope initial level |
| envelope release | 7 | 0 to 255 | Volume envelope release time |
| LFO speed | 8 | 0 to 255 | LFO frequency |
| LFO amount | 9 | -128 to 127 | LFO modulation amount |
| feedback | 10 | 0 to 255 | Operator feedback amount |
| flags | 11 | - | Operator configuration flags Refer to dedicated table |
| reserved 1 | 12 | - | Unused |
| reserved 2 | 13 | - | Unused |
| reserved 3 | 14 | - | Unused |
| reserved 4 | 15 | - | Unused |

| Operator flags (OR combinaison) | | |
|---------------------------------|----------|---|
| | Bit mask | Description |
| absolute | 1 | Absolute mode Operator does not track pitch |
| tracking soft low | 2 | Pitch to volume tracking Soft slope curve is applied on low pitches |
| tracking hard low | 4 | Pitch to volume tracking Hard slope curve is applied on low pitches |
| tracking soft high | 8 | Pitch to volume tracking Hard slope curve is applied on high pitches |
| tracking hard high | 16 | Pitch to volume tracking Hard slope curve is applied on high pitches |
| reserved 1 | 32 | Unused |
| reserved 2 | 64 | Unused |
| reserved 3 | 128 | Unused |

Technical specifications

OPA is FM-synthesis musical shield for Arduino / Genuino boards.

Synthesis part

- 4-operators (sine wave) structure
- 13 different algorithms
- 10 voices polyphony
- 8 simultaneous programs
- Absolute or pitch tracking operators
- 1 ADSR envelope & 1 triangle LFO per operator
- Special operator 4 with feedback and noise features

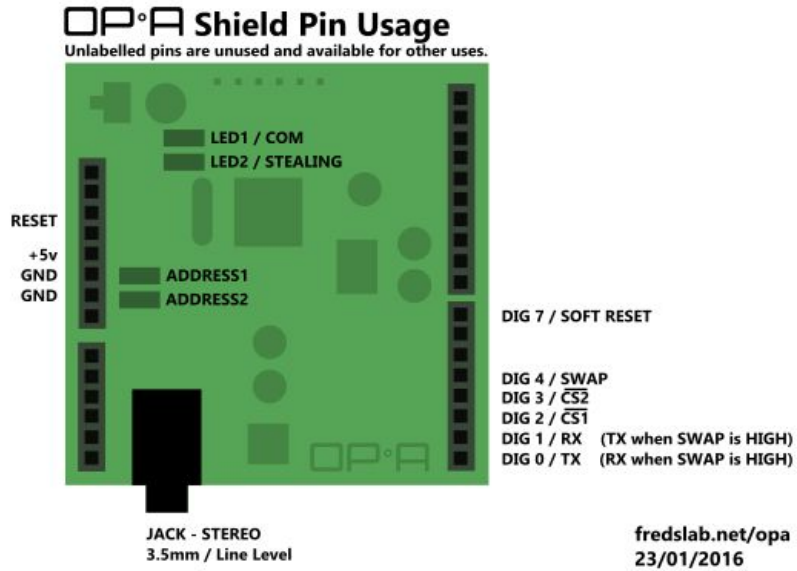
Electrical characteristics

- 16-bit high-quality stereo codec
- 3.5mm line level jack output (1kOhm impedance)
- Powered only with +5V / 100mA from Arduino
- Compatible with 0V / 3.3V and 0V / 5.0V logic

Mechanical characteristics

- Dimensions 53 x 53 x 20 mm (including headers)
- Weight: unknown yet

Pinout diagram



Block diagram

