Page under construction!

Summary

MIDIbox Quad Genesis (MBQG) is a homebrew synthesizer project in progress, designed and built by Sauraen and originally commissioned by Josh Whelchel. It runs up to four sets of YM2612/YM3438 (OPN2) + SN76489/94/96 (PSG) sound chips, one set of which roughly comprised the sound generation hardware in a Sega Mega Drive / Genesis. The synth is powered by a STM32F4 core running MIOS32, and is based on MIDIbox hardware and software throughout.

Useful Links

MIDIbox Quad Genesis forum thread

MIDIbox FM V2.1 (OPL3 synth on STM32F4) forum thread, including some old discussion with yogi about MIDIbox Quad Genesis ideas

Fill out this form if you might be potentially interested in buying MBHP_Genesis boards

Hardware

Hardware components

- CORE_STM32F4 module (embedded CPU, LCD parallel interface, DIO SPI level shifter, SD card, USB MIDI)
- Zero, one, or two MIDI_IO modules
- One, two, or four MBHP_Genesis modules
- One MBHP_Genesis_LS level shifter board to interface 3V MCU to 5V sound chip boards
- 2×40 character LCD (or VFD, or OLED…) (size mandatory)
- MBQG_FP custom front panel board; or alternately, DIN, DOUT, and/or DIO_MATRIX modules as needed to make custom front panel
- Stereo audio output connection of desired type
- Mono audio input connection for each SN76494/96 if desired
- Power supply (see below)

Power Supply

All the modules within MIDIbox Quad Genesis run at 5V, so there is no need for a fancy supply. However, the front panel draws up to a theoretical maximum of 1.2A (if the synth crashes with all the

LEDs on), and noise from the digital components can easily get into the analog audio output.

Because of these considerations, and wanting to have the most versatile power option possible, I powered my synth as follows. I mounted a standard barrel jack (with an insulating mount, so both terminals were still electrically isolated from the metal case) and a subminiature toggle switch on the back panel. Those, and an internal fuse holder with a 1.5A fuse, were wired in series. This connected to the AC terminals of a bridge rectifier I made from 1N4007 diodes, and the DC output went to a 3300uF electrolytic capacitor. This is the beginning of a standard linear power supply, which means that any input from any wall-wart or adapter in the appropriate voltage range (see below), whether AC or DC of either polarity, will work just the same.

This filtered DC was split and sent to two miniature switching regulators, OKI-78SR-5 (Mouser 580-OKI78SR5/1.5W36C), which is a fantastic switching regulator that functions as a drop-in replacement for a 7805 linear regulator. It can push 1.5A output and has an input range of 7-36V, meaning that any kind of wall transformer or switching regulator rated at about 8V-30V, AC or DC, will work on the synth.

The output of one switching regulator was used exclusively for the front panel, while the output of the other was sent to the core and the Genesis boards. A 0.1uF capacitor was placed from input to ground and output to ground on each regulator, and additionally a 10uF capacitor was placed at the output.

This approach worked fine for me. As far as noise, my synth does have audible noise at harmonics of 1kHz, but I don't think this is due to the power supply (the regulators switch at much higher frequencies, 500kHz). You are welcome to adapt this design to your needs or build a completely different power supply, though I wouldn't recommend trying to power the synth from USB due to the front panel.

Synth Architecture

This section was written during early development; while it does provide a roughly-accurate overview of what the synth does, the details haven't been gone over recently and many have changed slightly.

<table>
<tr><td>Dummy</td><td>Dummy</td><td>Dummy</td><td>Dummy</td><td>Dummy!</td></tr>
<tr><td colspan="5">Subscreens</td></tr>
<tr><td colspan="5">Modes</td></tr>
<tr><td colspan="2">VGM Streaming</td><td>VGM from RAM</td><td>VGM Realtime Queue</td><td rowspan="2">Interface</td></tr>
<tr><td>Stream Thread</td><td colspan="3">VGM Player</td></tr>
<tr><td>FAT File System</td><td colspan="3">Chip State Tracking</td><td>Front Panel</td></tr>
<tr><td>MIOS32 SD Card</td><td colspan="3">MBHP_Genesis I/O</td><td>MIOS32 DIO</td></tr>
</table>

Overview

The synth is intended to support two use paradigms. Both may be active simultaneously on different voices.

The first is the case in which the user has the synth connected to their DAW or to an extension to a tracker, and the user wants to have full control of every sound chip parameter via MIDI (MIDI is

supported over USB and UART). This is achieved by setting voices to Tracker Mode and assigning channels to control them. The front panel controls will reflect and edit the current voice state, and a short burst of MIDI messages can be dumped back to the DAW which when played back to the synth will restore the voice to the current state. No polyphony, modulators, or other synth engine features are available in this mode.

The second paradigm encompasses the following use cases:

- Playing Mega Drive/Genesis VGM files on the synth from the SD card.
- “Sampling” VGM files to create custom music–that is, extracting sample, instrument, and control information from VGM files, and then modifying them or playing new music with them.
- Creating intricate, custom sounds with FM synthesis and playing them in polyphony across all four sound chips.

The core of the synth engine is a module that plays back many (often over 30) VGM files from RAM in parallel on the eight sound chips. The VGM files being played can be edited down from ones saved on the SD card, and then set up so their commands will be played on different voices based on a dynamic-assignment polyphony engine. In addition, one VGM file at a time may be streamed from the SD card for auditioning or editing. A typical use would be loading and previewing a VGM file from your favorite Mega Drive/Genesis game, using the front panel controls to extract the configuration of a single instrument as a smaller VGM file, creating an instrument out of this new VGM file, and assigning this instrument to a channel to be played on all 24 FM voices in polyphony.

Voices

Each voice (any sound chip) can be in one of two modes: Tracker or Free.

Tracker Mode

A voice in tracker mode has little-to-no synth engine and is simply controlled in realtime by MIDI commands from a DAW or tracker. It is configured to respond to commands from one MIDI port on one channel, according to the GENMDM specification. I have been in contact with Aly James, the creator of FMDrive, the celebrated OPN2 VST plugin, and we are working to ensure the MIDI map of his software and my hardware are as identical as possible.

Free Mode

A voice in free mode is available to be assigned in real-time by the synth engine to any program. This sounds simple enough, but this requires tremendous complexity on the inside, for two reasons:

1. A VGM file can use any combination of voices on one pair of chips, and the voices are not homogenous (e.g. FM voices 3 and 6 can do things which 1, 2, 4, and 5 can't; and PSG voice 3 can do things which 1 and 2 can't; but the other way around does work correctly, you can play anything which was intended for voice 1 on voice 3).
2. Certain OPN2 features (the LFO and the test bits) are global to the chip, so VGM files which use these features have to be kept on the same chip or carefully assigned across multiple chips. Moreover, for certain of these features (the LFO), other voices which ignore the features can

share the chip, but for certain ones (the Ugly bit) the feature affects all voices.

The synth engine keeps track of what chip resources each VGM file and program use (its “usage”). (Even computing this isn't always trivial, since a VGM file can be streamed from the SD card!) When playing a note on a program, it determines where to assign all of its used voices based on many factors; and of course, when actually playing the commands within the file, it has to reroute them all as well as modifying their pitch if applicable, all at 44.1 kHz!

Interface Modes

Voice Mode

Select a voice using the Genesis buttons. The voice controls display the current state of the voice. If the voice is in tracker mode, the voice controls are functional to set the current state of the voice; this state of course may be overridden by subsequent MIDI messages. View the current Genesis's audio state on the VU meters, and select whether the right columns display the current voice's operator states or the PSG channel states.

Planned features:

- Capture button: create a VGM file which represents the state of the current voice. Display a menu of channels so you can choose where this will be stored. Upon selecting a channel, or pressing the Capture button again which will use the last edited channel, a new program will be set up on this channel with that VGM file as its init and appropriate keyon and keyoff files, so you can play this sound on the keyboard as usual.

Channel Mode

Press Chan. Use the screen to select a MIDI channel. (Please note that the number of channels in the synth are 16 x the number of ports, which is by default 4 but can be increased.) Set whether the channel is assigned to a voice in tracker mode or whether it should play instruments on voices in free mode. If the channel is in free mode, load, save, delete, or create a new program.

Program Mode

Edits the program selected in Chan mode.

A program consists of 3 VGM files: one to init the voice(s), one played at key on, and one played at key off. Program mode allows the loading, saving, deleting, or creating new template-based files for any of these functions. Additionally, this screen lets the user select the root note of the program, i.e. which keyboard note corresponds to no pitch change from the given VGM files.

Planned features:

- VGM file tempo adjustment, including the option to scale tempo with pitch (like changing analog speed).

- Drum type programs, which instead of having three VGM files, have 16, one for key-on for each of a definable split region. Key off is fixed to just key-off or silence the used voices, and the key-on VGM has to contain any set-up commands.

VGM Mode

Full-featured editor for VGM files loaded into RAM, and partial editor for VGM files streamed from the SD card.

Both modes support real-time preview of the VGM file, muting and soloing voices (applies even to copies playing from key presses, not just the preview copy).

For VGM files loaded into RAM, supports editing, inserting, and deleting individual VGM commands in Cmds mode. In State mode, shows the current state of the voice on the edit controls, and allows editing of this state directly (by changing the last command before the present which affected that parameter).

Planned features:

- For either type of VGM, support defining soft start and end points (“mark”) within file. Then, press Crop to make the VGM be only this section, and also only the voices which are soloed or not muted. If the VGM file is a stream and the result of the Crop operation would fit in RAM, load the result into RAM for further editing.
- For either type of VGM, Capture the current state of a voice as in Voice mode.
- For RAM only, move, duplicate, or copy-and-paste the individual commands or the marked section to a different position.

From:
https://www.midibox.org/dokuwiki/ - **MIDIbox**

Permanent link:
https://www.midibox.org/dokuwiki/doku.php?id=midibox_quad_genesis&rev=1486621023

Last update: **2017/02/09 06:17**