

YM_Mini Synth chip synthesizer

YM2149 usb/midi synthesizer

Getting Started with the software and Testing (Version 1)

Wil Lindsay

www.straytechnologies.com

wil@straytechnologies.com

What is it?

In 1986 Yamaha licensed IC production from General Instruments for a Software-Controllable Sound generator, or 'SSG' Chip. The chip was designed with three Squarewave channels, a noise generator and envelope generator to be controlled from external microprocessors in video games and home computer applications. The chip is most well known for its use in the **Apple II Mockingbird** soundcards and **Atari ST** computers. The YM or the chip from which it was derived (GI's AY-3-8910) also created the music and audio in the **Intellivision**, **Vectrex**, some **MSX** systems, **Sinclair ZX** and **Spectrum** home computers, and many **arcade machines**.

This synthesizer was designed with a few main goals after a lengthy survey of chiptune musicians across the Internet. If I learned anything from this experience it's that a \$5 über-synth that runs on trackers, sequencers, responds to dulcimer tablature, and washes bicycles would be viewed as either too expensive or lacking features by someone. With all of that in mind, I decided to focus on my main goal – Hackable.. meaning any user or builder can make it work the way they want with some effort.

The main goals in design are:

- 1: affordable – and/or user buildable to save on labor cost
- 2: midi based – to maintain wide compatibility with software, trackers, and midi controllers
- 3: flexible & extensible – (yeah, direct midi control will be achieved with very little modification)

How it works:

The system is mostly software based as that was the original intent of the YM chip's designer. As such, control of the chip is created in a modular sense:

musician → software or controller → midi → KIT_MIDI driver → usb to serial cable → microcontroller → YM2149 → stereo jack → mixer or speakers → ears → brain

In order for this chain of modules to work, a certain amount of software setup is required. The system should work roughly the same way on various flavors of WIN or OSX, though there will always be some differences. I'll address the modules above in reverse order as the higher level modules are dependent on everything else working:

Mixer, Speaker, Headphones

Output from the YM_Mini is 1/8" stereo. There is no amplifier circuit, though the chip does a fairly decent job of maintaining the signal. Output could go directly to headphones (no hardware volume control, though), amplified speakers, a mixer board, or computer line-in. Though I don't particularly use them, the signal could likely be patched through hardware effects units as well(mmm. Distortion!).

YM2194 or AY-3-8910

The YM2149 or 2149F is a 40 pin IC that can be plucked from dead devices or still occasionally found NOS (still sitting in a warehouse somewhere). The chip has 3 audio channels which I've routed to stereo in the model of a fairly common AtariST stereo "hack" from back in the day. By design, the YM's sister chip the General Instruments GI's AY-3-8910 is pin compatible, and should work as well with one minor nuance: R10 resistor will not be used.

Microcontroller

The midi signal and synth control is managed by an Atmel Atmega328, well known for it's use in Arduino projects. I've decided to use this controller to make firmware upgrade and changes easily possible for ALL users. As such, the circuit is designed to accept firmware upgrades through the existing USB cable and free Arduino software available at <http://www.arduino.cc>.

USB to Serial Cable and Drivers

The synth receives its command and power directly from a computer's USB port via an FTDI cable or USB-to-Serial FTDI breakout board. This cable maintains the chore of converting modern USB to something the controller can handle, while allowing the option for 6-pin control by other hardware or microcontroller based systems.

Cables or breakout boards are available with the kit or here:

<http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=768-1028-ND>

or here:

<http://www.moderndevice.com>

or many other hobby electronic shops.

These cables require a "virtual com port" driver to be installed on Windows, OSX, or Linux which is available here:

<http://www.ftdichip.com/Drivers/VCP.htm> in MANY flavors.

Users will need to know what the device is referred to on their specific system. On Windows, check your 'Device Manager' for the newly installed Port (COM & LPT) to find the setting: something like "COM3." On OSX and Linux the installed driver should show up as something like "/DEV/TTY/USBVIRTUAL-Ftxxx" in the YM Midi software mentioned below.

KIT_MIDI_V2 Synth DRIVER software

This software will be occasionally updated, with the newest version available at:

<http://www.straytechnologies.com/resources/>

the software is available in WIN, and OS X versions.

This is Java based software, and will need to be run every time the synth is used from a computer.

On opening the software (yeah..it starts slowly) the user will need to click the check box for the Serial Port supplied by the FTDI driver mentioned above. The user then needs to select what virtual midi "port" or "virtual cable" the synth will be connected to. After a few seconds of internal software chugging, the driver will display "Ports Connected" and can be minimized (but needs to be left running) for the duration of your session. If you're running multiple YM kits, each synth will need its own running copy of the KIT_MIDI driver.

I've had great success with this system, though Windows users will have a smoother experience if the Serial Port is always selected before the Midi port. I've also found that plugging-unplugging hardware controllers such as the M-Audio Key Stations (or the YM_Mini, for that matter) causes some panic in the software, and will likely require a restart of the software.

Midi

Software use of the synthesizer will require some sort of Virtual Midi Cable system to be installed on your system before using the KIT_MIDI Driver and the YM_Mini synth. If you don't already use midi software, setup is somewhat different on various operating systems, and is quickly outlined below.

OS X based systems:

On OS X there is a built in system called IAC (Inter Application Communication) which can be found in Applications > Utilities > Audio Midi setup. Click on IAC Driver in the Midi Window, and verify that the "Device is online" box is checked. Under "More Information," you can add additional midi busses or ports as needed. In KIT_MIDI and other software these should show up as "IAC DRIVER MIDI 1" or whatever number your port is.

Folks running pre- Mac OS X 10.6 will discover that Apple and Java do not always play together well. In this case you may need to install "mmj" midi for Java, as Apple evaded implementing midi in some java updates. Info and the files are at : <http://www.humatic.de/htools/mmj.htm>

WIN based systems:

In Windows you'll need a "Midi loopback device" installed on your system. There are several versions such as "LoopBe1" and "Hubi's Midi loopback system", found on the Internet with their respective installation instructions.

I can't speak highly enough of Jamie O'Connell's "Midi Yoke" at:

<http://www.midiox.com/index.htm> this program has been around since the AOL/ CompuServe days and is still maintained and works great! With either solution, the virtual midi device is installed like real hardware and appears to be an actual midi port to the OS and software. It will show up in the KIT_MIDI software under it's respective name in the right hand column, once correctly installed.

Linux based systems:

This is a WIP, and Linux users are probably already used to doing things on their own. So, I will simply point you to the ALSA library's virtual midi card "VirMidi" and hopefully fire all of this up on my midi box for detailed installation instructions in the near future. There are also Python options for those ready to explore the nuts and bolts.

Controlling the YM_Mini

Your new best friends will be the Midi Implementation section below and the YM2149 datasheet. In short, the system has been tested and works happily with EnergyXT, PureData, Max/MSP, MidiOx, Protrekkr, ztracker, Bliptonome (Arduinome/Monome) hardware and the Oxygen8 hardware midi controller.

I firmly believe that the implementation is hardy enough to handle control from any software that can connect to a virtual midi cable. I've also tested the system on several OS's and various machines including a sluggish Dell netbook, and find that it's fairly stable. That said, remember that you're simultaneously running a LOT of software, and your Aunt's Pentium-I may not be up for the challenge. Faster systems will mean less midi timing glitches and Java crashes. Running 5 VST's, FruityLoops, browsing the Interweb, and doing your taxes at the same time will mean more problems.

Midi Implementation

The current midi implementation will likely take on some additions as new firmware is released and users start fussing with nuance. The main highlights are here with categorized midi implementation below for the detail obsessed. The most important thing to remember is that this is a low level sound generator from the 1980's. Squarewaves. Noise Channel. No filters. Awesome.

General Usage

Mapping the western midi scale to the chip has proven to be a chore, though my current implementation uses a fairly accurate look-up table, as apposed to any quirky, non-scalable formula magic. That said, the chip can't handle low notes below Midi note 24 (C-1), so they just don't exist. If you want to go lower than that, or between western notes, the answer is to pitchbend your way there. Yes. I've implemented Pitchbend: It is limited to the resolution of the chip, so low notes bend smoothly, and high notes bend like an Intellivision in a blender.

The synthesizer is set up with the following midi channel scheme:

CH1: Centered mono Squarewave A
CH2: Right Channel only Squarewave C
CH3: Left Channel only Squarewave B
CH10: Noise Channel

There is only one noise channel, but it can be sent to various channels using Midi Continuous Controller (CC) 28 as follows

value 0-31 = Channel A (Mono)
value 32-63 = Channel C (Right)
value 64-95 = Channel B (Left)
value 96-127 = Channels C&B (Stereo)

All channels have VELOCITY and PITCHBEND, but no aftertouch/key pressure. VELOCITY is the best way to handle discreet channel volume in your overall mix.

Volume Envelopes

The volume envelope can work on any channel or all channels, but there is only one. Using the envelope excludes VELOCITY control on that channel. Volume envelope shape and duration are controlled with Midi CCs, that will be read from any active channel. Control is as follows:

- CC 22 CH1 Envelope ON/OFF (at or above value of 63 is ON, otherwise OFF)
- CC 23 CH2 (RIGHT) Envelope ON/OFF (at or above value of 63 is ON, otherwise OFF)
- CC 24 CH3 (LEFT) Envelope ON/OFF (at or above value of 63 is ON, otherwise OFF)
- CC 25 ROUGH envelope duration (frequency) control
- CC 26 FINE envelope duration (frequency) control
- CC 27 Envelope SHAPE – in 10 increments of different shapes as per the datasheet

Caveats (or “Features”)

NoteOn commands between 0 and 23 are ignored.

The synth can't handle it, see the note above and the datasheet if you really need your music to hang out below that. For western scale, C-1 is your lowest note.

Noise and squarewaves are sharing channels.

This means the sound is “mixed” in the chip and will produce all sorts of odd anomalies. I've gotten some beautiful distortionary squawks out of this beast by intentionally playing with this fact. Those with lighter constitutions may wish to ping-pong channels and keep things clean, (which is in its own way very interesting).

Envelopes effect the whole channel.

If you have noise and tone playing centered, left or right, then turn on the envelope to effect your tone on the same channel... your noise will follow the envelope as well. (meaning one distortionary tone) Ping-ponging is the only resolution for this as well.

Noise channel stereo sweeps.

It's possible to sweep the noise from one channel to another by using the stereo setting on Midi CC 28, and setting the envelope to only effect one or the other channel (CH2 or CH3) with an attack shape. Effectively, this will attack in one channel and sustain in the other.

Vibrato modulation exists.

Setting the envelope frequency fairly fast with a saw shape envelope will give it a vibrato effect. This works on all sounds and channels.

Sample playback??

If continuous wave envelopes are used with a rough frequency of zero, and the fine frequency modulating to the sample, some resemblance to “playback” is quite possible. At very least some very interesting “formant-esque” effects are at hand. I've also seen this referred to on the ST as a “Buzz filter”.

Current Midi Implementation

Command		Data1	Data2	Respective YM2149 Registers
NoteOff				
CH1	0x80,	Note	Velocity	R7,R8
CH2	0x81	Note	Velocity	R7,RA
CH3	0x82	Note	Velocity	R7,R9
CH10	0x89	Note	Velocity	R7,R8
NoteOn				
CH1	0x90,	Note	Velocity	R1,R0 R7,R8
CH2	0x91	Note	Velocity	R5,R4 R7,RA
CH3	0x92	Note	Velocity	R3,R2 R7,R9
CH10	0x99	Note	Velocity	R6 R7,R8
Pitchbend				
CH1	0xE0	LSB	MSB	R1,R0
CH2	0xE1	LSB	MSB	R5,R4
CH3	0xE2	LSB	MSB	R3,R2
CH10	0xE9	LSB	MSB	R6
Midi Ccs (any channel c)				
Envelope Generator				
CC22	0xBc	0x16	VALUE<>0x3F	R8 Mode OFF/ON
CC23	0xBc	0x17	VALUE<>0x3F	RA Mode OFF/ON
CC24	0xBc	0x18	VALUE<>0x3F	R9 Mode OFF/ON
CC25	0xBc	0x19	VALUE	RC
CC26	0xBc	0x1A	VALUE	RB
CC27	0xBc	0x1B	VALUE	RD
Noise Channel				
CC28	0xBc	0x1C	VALUE	R7 >> 5 0=CHA 1=CHC 2=CHB 4=B&C

Let me know how it goes. Feedback means better firmware. Awesome tunes means inspiration to keep working. Drop me a tune or a line, and let me know how it goes.

-Wil

wil@straytechnologies.com