megamega2612 FM SYNTHESIZER

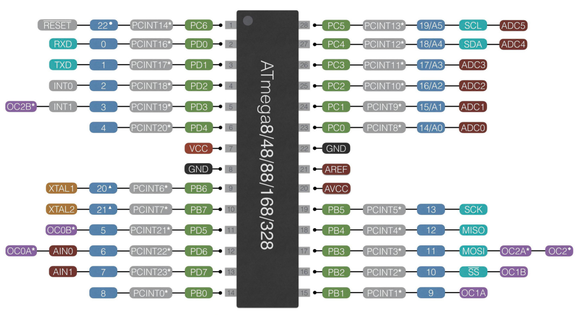
BY BONEY CIRCUITRY (ISAAK THOMPSON)

OVERVIEW

Boney Circuitry's **megamega2612** is a fully programmable **4-operator FM synthesizer** based around Yamaha's infamous **YM2612** IC.
The YM2612 is programmed by one peripheral **ATmega328p** IC, which is controlled by a separate controller ATmega328p via **SPI communication.**
Notes are played via **MIDI in**, which is routed through a **6N138 optocoupler** IC, and received by the controller ATmega's **USART RX** pin.
The synthesizer can be programmed using a minimal and intuitive UI consisting solely of a rotary encoder with a momentary push button, and one separate push button.
The system is programmed in **C**, and uses exclusively **interrupt-driven programming,** with no polling whatsoever.
A **16x2 HD44780 LCD** displays all pertinent information, which includes individual patch parameters as well as **preset patches** and additional options.
Stereo audio is routed through a **dual potentiometer** and a prefabricated preamplifier module based around the **TDA1308 class-AB stereo headphone amplifier**.

ATMEGA328P

The megamega2612 incorporates two ATmega328p ICs. One is the main controller and is connected to the MIDI in and user interface, and to the YM2612 controller via SPI. Many of its functions are utilized, including timers for the YM2612's clock and for firing timed interrupts, the robust RS232 and SPI communication protocols, and its many I/O pins. Ceramic resonators are used for the clock of both ATmega328p ICs.



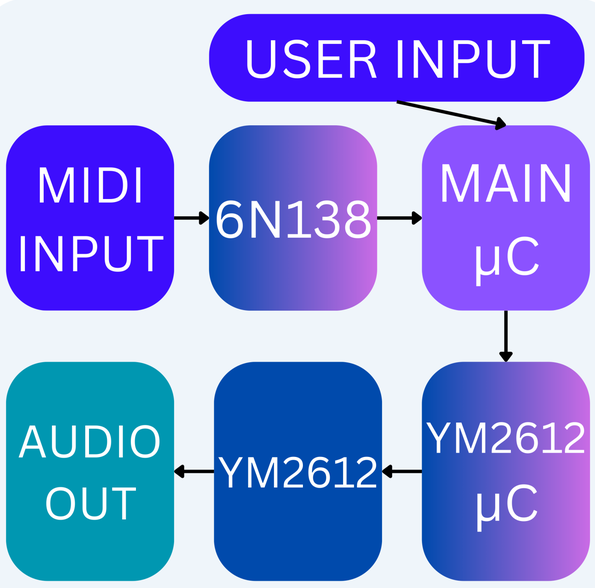
ATmega328p pinout

INTERRUPTS

The system is run entirely by interrupts, making it quite efficient. When MIDI data is sent to the main controller, a USART RX interrupt is fired, and the data is routed appropriately. If a note on or off message is triggered, the note value is loaded into an array, and a flag is raised to signify that a note should be turned on or off. When timer 1 overflows, an interrupt is fired that handles any notes scheduled to be turned on or off. If a button is pressed or the encoder is turned, a pin change interrupt for pins D0-D7 is fired and the system discerns which pin change caused the interrupt. If SPI data is sent from the main controller to the YM2612 controller, an SPI interrupt is fired and the appropriate registers are set in the YM2612.

CONTROL FLOW

The system relies on a complex cascade of data which starts at the user side and ends in audio output. MIDI messages are decoded and translated into register addresses and data for the YM2612, and sent to its respective controller via SPI. User input is received and values are updated and sent to the YM2612. Audio is output through a TDA1308 headphone amplifier (not included in the schematic).



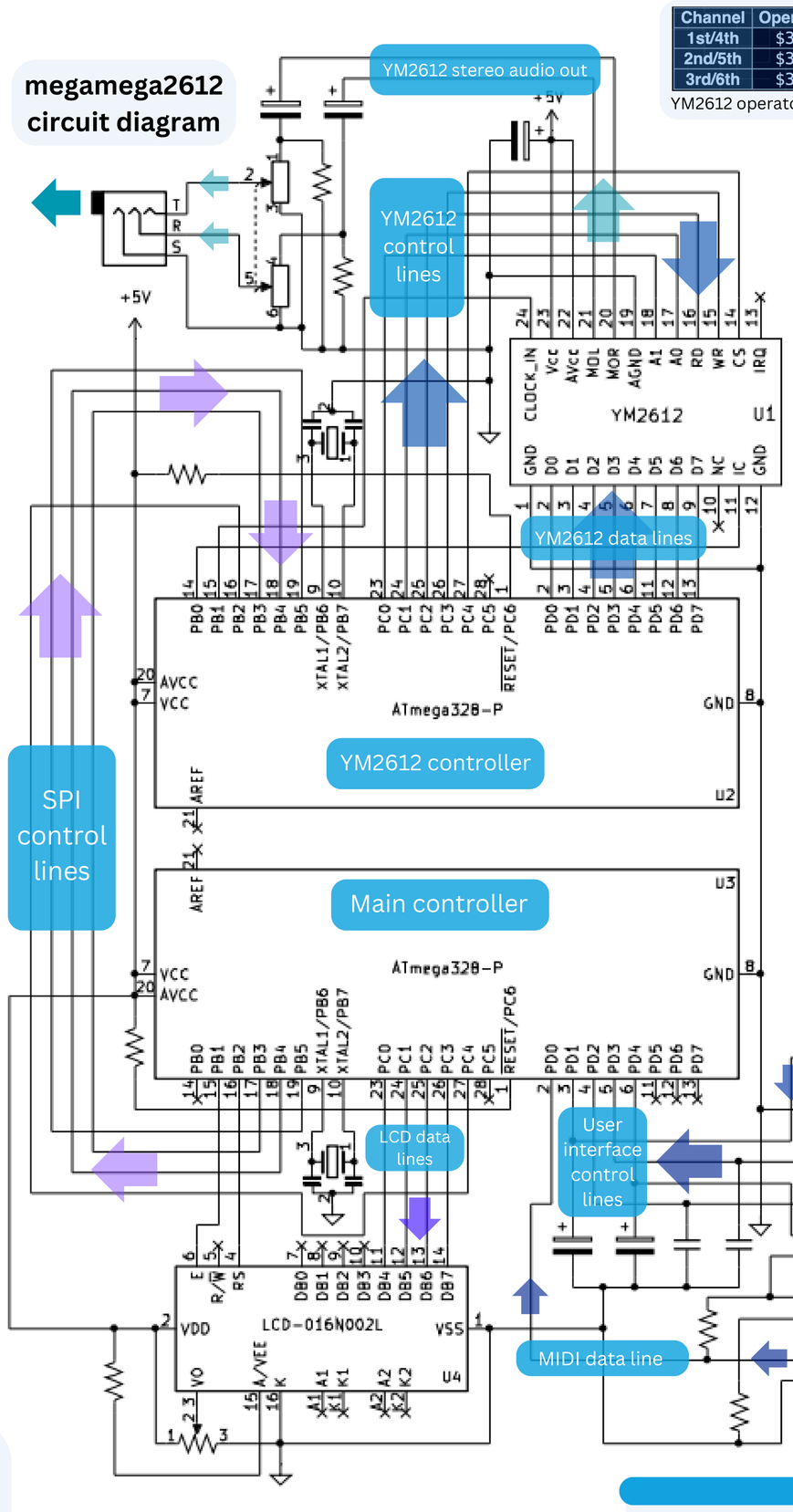
YM2612 frequency register A4+ (high)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	BLK:2	BLK:1	BLK:0	FREQ:10	FREQ:9	FREQ:8

YM2612 frequency register A0+ (low)

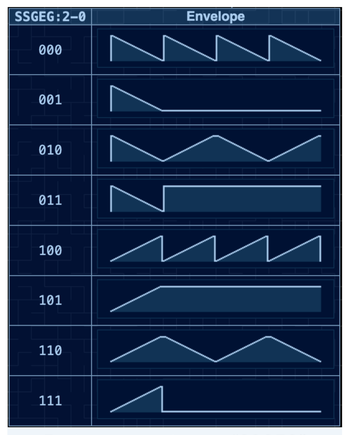
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FREQ:7	FREQ:6	FREQ:5	FREQ:4	FREQ:3	FREQ:2	FREQ:1	FREQ:0

megamega2612 circuit diagram

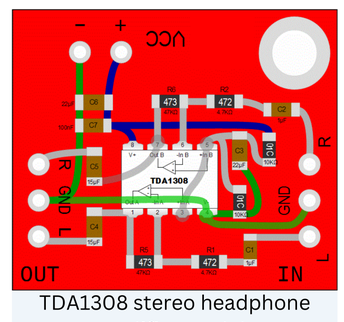


Channel	Oper S1	Oper S2	Oper S3	Oper S4
1st/4th	$30	$38	$34	$3C
2nd/5th	$31	$39	$35	$3D
3rd/6th	$32	$3A	$36	$3E

YM2612 operator and channel address structure



Each operator has an "SSGEG" option, which causes its amplitude envelope to repeat or behave in specific ways, as shown above



TDA1308 stereo headphone amplifier module

YM2612

The megamega2612 uses Yamaha's YM2612 IC, which was responsible for the infamous sound of the Sega Genesis and Megadrive video game systems from the 1990s. Its internal architecture consists of a series of registers, accessed by sending first the address of the register, followed by the data, while toggling appropriate pins.
It has 6 channels, allowing for 6-voice polyphony, with 4 "operators" (sine waves) per channel.
The operators can be arranged in various frequency modulation algorithms, envelope configurations, and other parameters for an enormous variety of sounds.

No.	Pin Name	I/O	Function
1	GND	–	Ground pin.
2	D_0	I/O	8-bit bidirectional data bus. Communicates data with the processor.
3	D_1		
4	D_2		
5	D_3		
6	D_4		
7	D_5		
8	D_6		
9	D_7		
10	TEST	I/O	Pin to test this LSI. Do not connect.
11	$\overline{IC}$	I	Initializes the internal register.
12	GND	–	Ground pin.
13	$\overline{IRQ}$	O	Interrupt signal issued from the two timers. When the time programmed into the timer has elapsed, this goes low. Output with open drain.
14	$\overline{CS}$	I	Control the D0 – D7 data bus. (CS / RD / WR / A1 / A0 / Details) 0 1 0 0 0 Writes register addresses of timers, etc. · 0 1 0 0 1 Writes register data of timers, etc. · 0 1 0 1 0 Writes register addresses of channels 4-6. · 0 1 0 1 1 Writes register data of channels 4-6. · 0 0 1 0 0 Reads statuses. · 1 X X X X D0 – D7 are set to high-impedance.
15	$\overline{WR}$		
16	$\overline{RD}$		
17	A_0		
18	A_1		
19	A GND	–	Ground pin.
20	MOR	O	Two-channel analog outputs. These are output with a source follower.
21	MOL		
22	A V_{CC}	–	+5V power supply pins.
23			
24	ϕ M V_{CC}	I	Master clock input.

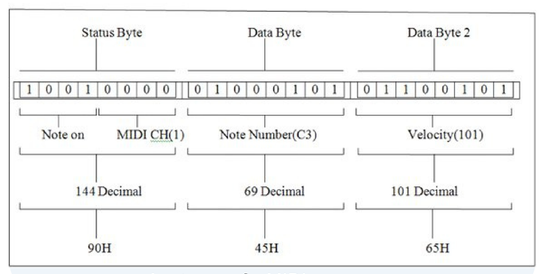
Description of YM2612 pin functions

Algorithm #	Layout	Suggested uses
0		Distortion guitar, "high hat chopper" (?) bass
1		Harp, PSG (programmable sound generator) sound
2		Bass, electric guitar, brass, piano, woods
3		Strings, folk guitar, chimes
4		Flute, bells, chorus, bass drum, snare drum, tom-tom
5		Brass, organ
6		Xylophone, tom-tom, organ, vibraphone, snare drum, base drum
7		Pipe organ

YM2612 FM algorithms

MIDI IN

MIDI messages are routed through a 6N138 optocoupler IC in order to protect the microcontroller. Messages consist of 2 or 3 bytes, and include note on or off data (note, velocity), pitch bend, modulation wheel, aftertouch, etc.



Anatomy of a MIDI message

INTERFACE

The user interface of the megamega2612 consists of a 16x2 LCD screen, a rotary encoder with a built in push button, and a second push button. Upon startup, the megamega2612 displays its name, followed by the name of the default preset patch. The user can select different patches, or use the buttons to navigate to different groups of parameters. If the user presses the encoder button while turning, a different parameter within the group is selected. If the user presses the button while turning the encoder, a different operator is selected. If the user turns the encoder without pressing a button, the value of the currently selected parameter changes

CONCLUSION

The YM2612 is a cult classic in the video game music world, as well as the DIY synthesis world. There are several similar projects to this one, some of which are even commercially available, such as the DAFM synth:
https://www.tindie.com/products/kassersynths/dafm-synth-genesis-ym2612-ym3438-diy-kit/
and the RYM2612 plugin:
https://www.inphonik.com/products/rym2612-iconic-fm-synthesizer/
Many before me laid the path that made this project possible, and to them I am grateful. In the future, I will continue to develop the sonic capabilities of the megamega2612, possibly incorporating analog circuits such as a voltage controlled filter, expanding on the modulation possibilities of MIDI, or including the option to save and recall presets within the program.

ACKNOWLEDGMENTS

Most YM2612 knowledge and figures were derived from here:
https://www.plutiedev.com/ym2612-registers
and here:
https://www.smspower.org/maxim/Documents/YM2612
Inspiration for the YM2612 control code came from here:
https://github.com/AidanHockey5/MegaMIDI/
and here:
https://github.com/Ryan-Marchildon/ym2612-arduino-test
Inspiration for the MIDI code came from here:
https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html
and here:
https://benryves.com/projects
The LCD library was written by Joerg Wunsch and adapted for ECE:3360 by Chaz WIlmot and Tristan Pawlenty:
https://github.com/chazwilmot/LCD_library
And of course, many thanks to Professor Kruger!