Computer Documentation

Revision history:
This document applies to Proto 2.0.1 and Rev 1 which are identical
See final page for details of hardware revisions

What's this all about?

I'm developing the Music Thing Workshop System, a new standalone modular synth that fits into a small peli-style case.

The right hand side of the device has classic analogue circuits - 2 x oscillators, 2 x filters, 2 x envelopes, a ring mod and lots of connections to the outside world = stereo inputs, headphone outs, contact mic preamp, stompbox send and return.

On the left hand side is a computer that reads tiny little program cards.

Someone asked me what my intentions were with this device, this is what I told them:

> *I think my intention is to make something that is useful and open.*
>
> > *Open = you can connect it to other things - take audio from phone/laptop/tracker and process it, connect to and feedback pedals.*
> > *Open = connect two devices and improvise together.*
> > *Open = The computer block on the left makes that much bigger - people who write code can make program cards for it - sequencers or Audio effects or sample/delays.*
>
> *Hopefully that openness leads to community and longevity - so in 5 years time people are still extending it, finding more interesting things to do with it. My ideas are the small limited seeds that hopefully grow into something more interesting.*

Wait… a computer?

Every studio needs a computer. This is the vibe I'm going for:

Or maybe the EMS studio in Putney, with a pair of PDP-8 minicomputers:

What might the computer module be used for?

A few ideas are already underway:

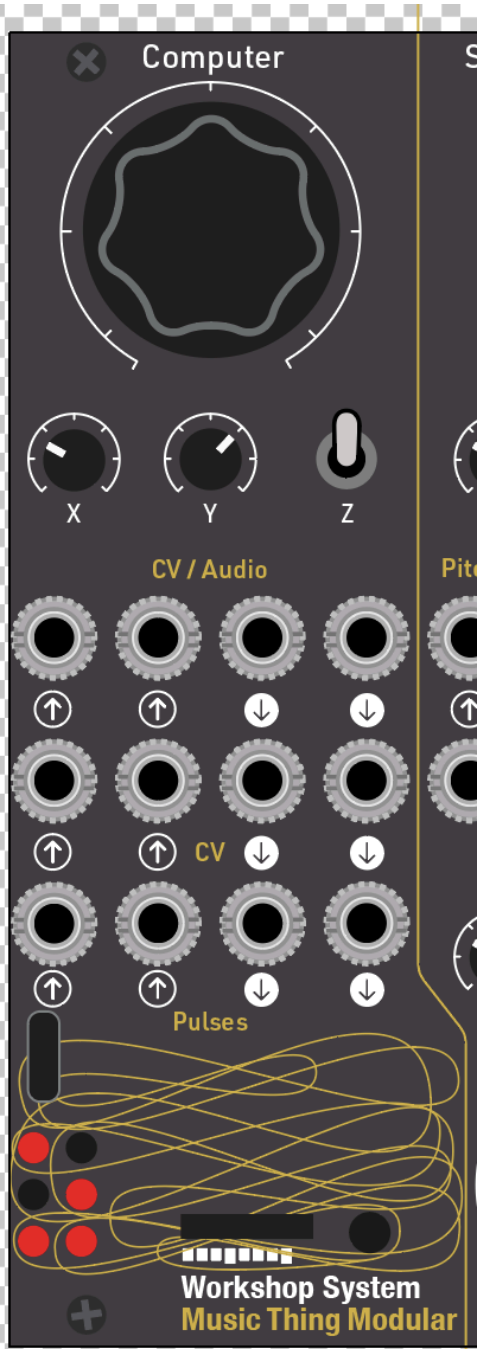
- The uSeq team from Sussex Uni are porting their amazing live coding environment to the Computer
- Sebsongs will be porting a simplified version of his brilliant Sampler module
- I'll do a Turing Machine module or something random-ish
- And Hunter Adams at Cornell has promised me something involving synthetic birdsong

I can imagine a huge range of others:

- Wavetable and additive and FM and other types of oscillators, perhaps quantized to standard or non-standard tunings
- Playback of short samples stored in up to 16mb flash on the card
- Wild computer music experiments, like Xenakis Gendys
- Short mid-fi modulated delays, and all the resonators, phasers, flangers, chorus, pitch shifting that can come from that. In stereo.
- Lofi reverbs
- Lofi delays
- Polyphonic sound generation / chords (there are 2 x nice multimode filters just to the right)
- Midi or OSC input and output
- Ableton link, maybe
- Generative Drum synthesis or pattern generation
- Speech synthesis
- AI audio analysis and response
- Beat detection and matching
- Physics simulations - bouncing balls and dual pendulums
- Chaos simulations
- Weird browser-based interfaces
- Arpeggiators
- Generative sequencers
- Auto tuning for the analog oscillators
- Self-playing albums released as cards, with instructions for how they should be patched to the rest of the device.
- Noise generators
- Data sonification
- Braids and Grids and other Mutable Greatest Hits
- Euclidean and de Bruijn and other rhythm generators
- A USB midi host, if that's possible
- Things that use flash_get_unique_id() which pulls a unique 64 bit ID from the flash memory, so every card is unique.
- A full generative 303 with square/saw osc, sequencer, drum patterns
- Epic super slow sequences and LFOs

I think of these cards as fanzines - quick, exciting experiments that can be made in a week or two, then shared. These are fun ideas that probably don't merit an entire module but would be nice to have on a memory card.

About the Computer module



The computer module is an RP2040.

But the Flash memory is on a removable card the size of a micro SD card.

So each card is a completely self-contained system, with all the firmware and local storage.

The Computer has a small 8kbit I2C Eeprom chip, to store calibration data and potentially other information that can be shared between cards.

This means cards can be coded in anything - C/C++ with the RP2040 SDK, Arduino, MicroPython, whatever you want to use. Change the clock speed, whatever works for you. Use clever PIO tricks and the dual core, or just repurpose slabs of old Arduino code.

The Computer module has:

- 3 x Potentiometers
- 1 x Switch - which is (ON)-OFF-ON - momentary push down, normal pull up
- 6 x LEDS, powered from the +12v line to remove noise on the RP2040
- 1 x 8kbit I2C eeprom
- 1 x powered USB data connection - this can give ~150mA power before drawing too much from the onboard voltage regulator
- 1 x standard debug header and a UART connection.
- Normalisation probes to identify if anything is connected to an input channel.
- A reset button on the front panel next to the card slot
- A Boot selection button behind the top pot, so you'll take that off to write .uf2 to a card.

2 x Pulse Inputs, 2 x Pulse Outputs

- Simple digital on/off signals, buffered and scaled with transistors.
- Use them for clocks, pulses, gates. They could also produce unfiltered PWM signals, so could maybe be used for gnarly audio (loud!) or gritty CV.
- They'll often be used to trigger the envelopes, which are Serge-style voltage controlled slopes
- The gates are about 5-6v

2 x CV Inputs, 2 x CV Outputs

- These are bipolar DC-coupled inputs and outputs, both approximately -6v to +6v
- Inputs are read through a 4052 multiplexer into the internal ADC on the RP2040, so are theoretically 12 bit but really more like 8-10 bits. I haven't yet tested the maximum read frequency.
- Outputs are 11 bit PWM signals at 60kHz, filtered with two pole active multifeedback filters, so they should be reasonably clean and fast settling
- They'll be used to control oscillators, filters, envelope speeds, and can also double as extra gates, LFOs or maybe even audio outputs.
- It is also possible to calibrate the outputs to reasonable accuracy using ~15-bit Delta Sigma PWM.

2 x CV/Audio Inputs, 2 x CV/Audio Outputs

- These are also bipolar DC-coupled inputs and outputs, approximately -6v to +6v.
- The inputs are scaled, then go directly to channels 0 and 1 of the ADC. As above, theoretically 12 bits, probably fewer.
- Both DC coupled - like everything, this is up for discussion, let me know if you have a strong opinion.
- The left input is normalled to the right input, so if only one socket is plugged in, both channels see that input.
- Outputs come through an MCP4822 DAC.
- The audio ins and outs might be used for effects, or as oscillators, or as extra CV inputs and outputs. Digital oscillators will be particularly useful for polyphony and pitch precision or microtuning.

A note on voltages: I've used a +/-6-ish volts range to ensure that the Computer outputs can push the analogue elements to their full range. However, I'll be taking feedback and tweaking this (and the analogue sections) as we develop.

RP2040 to Computer Pinout

GPIO0	UART0_TX	From unpopulated headers next to LEDs
GPIO1	UART0_RX	From unpopulated headers next to LEDs
GPIO2	PULSE_1_INPUT	Inverted Digital input: Low input = High reading. For example, use a falling edge to track the start of a pulse. NB: Input pin must have the pullup enabled, this powers the transistor.
GPIO3	PULSE_2_INPUT	As above
GPIO4	Normalisation Probe	Connected to the switch inputs on all the inputs via a BAT54S protection diode. Toggle this pin to identify which sockets have plugs in them.
GPIO5/6/7	Board identification	GPIO Pins 5, 6, 7 = binary bits 0 0 0 = Proto1.2 (all pins floating) 1 0 0 = Proto 2.0, 2.0.1, Rev1
GPIO8	PULSE_1_RAW_OUT	Inverted digital output: 1/true = low, 0/false = high. Scaled via a transistor. Pin should be input, no pullup.
GPIO9	PULSE_2_RAW_OUT	As above
GPIO10	LED_1	Leds are driven through a NPN transistor array from the +12v rail. 1/true = LED is illuminated.
GPIO11	LED_2	As above
GPIO12	LED_3	As above
GPIO13	LED_4	As above
GPIO14	LED_5	As above
GPIO15	LED_6	As above
GPIO16	EEPROM_SDA	Connects to Zetta ZD24C08A EEPROM, clone of 24C0* chips. Contains 8K bits (1024 X 8). SDA & SCL lines have 2.2k pullups. Data is stored in 8 x 1024 bit pages addresses 0x50 to 0x5B
GPIO17	EEPROM_SCL	As above
GPIO18	DAC_SCK / SCK	MCP4822 Control
GPIO19	DAC_SDI / MOSI	MCP4822 Control
GPIO20	Not connected	
GPIO21	DAC_CS / CS	MCP4822 Control
GPIO22	CV_2_PWM	Inverted PWM output. Two pole active filtered. Use 11 bit PWM at 60khz. 2047 = -6v 1024 = 0v

		0 = +6v Requires firmware calibration for precise values
GPIO23	CV_1_PWM	As above
GPIO24	MUX_LOGIC_A	This is a 4052 Multiplexer with 2 x 4 channels. Truth table is below
GPIO25	MUX_LOGIC_B	As above
GPIO26	AUDIO_L_IN_1	Inverted bipolar analog input,into 12(?) bit internal ADC +6v = 0 0v = 2048 -6v = 4095 DC Coupled, requires calibration for precise readings
GPIO27	AUDIO_R_IN_1	As above
GPIO28	MUX_IO_1	Analog Input 2, from the Multiplexer
GPIO29	MUX_IO_2	Analog Input 3, from the Multiplexer

4052 Multiplexer Truth Table

Logic A	Logic B	ADC Channel 2 GPIO 28	ADC Channel 3 GPIO 29
0	0	Main Knob	CV 1
0	1	X Knob	CV 2
1	0	Y Knob	CV 1
1	1	Z Switch	CV 2

LED Numbers

🔴LED1 = GPIO10	🔴LED2 = GPIO11
🔴LED3 = GPIO12	🔴LED4 = GPIO13
🔴LED4 = GPIO14	🔴LED5 = GPIO15

Eeprom Memory Map

PAGE 0 0x50 memory map for 2 x precision PWM voltage **outputs** = Channels 0 and 1		
Offset	Bytes	Contents
0	2	Magic number = 2001 - if number is present, eeprom has been initialised

2	1	Version number 0-255
3	1	Padding
4	1	Channel 0 - Number of entries 0-9
5	40	10 x 4 byte blocks: 1 x 4 bit voltage + 4 bits space | 1 x 24 bit setting = 32 bits = 4 bytes
45	1	Channel 1 - Number of entries 0-9
46	40	10 x 4 byte blocks: 1 x 4 bit voltage + 4 bits space | 1 x 24 bit setting = 32 bits = 4 bytes
86	2	CRC Check over previous data
88		END

FAQ

What language does this support?

- Because there is no firmware in the Computer module, you can use any language you like. So far, people are developing in Arduino Pico, Pico SDK (C/C++), Circuit Python and Micro Python.

How do I get started?

- There is a friendly and active Discord community and a collection of Work-in-Progress program cards which can be treated as sample code.
- Over time, we'll develop easier to use 'Hello World' sketches and code templates, but we're not quite there yet.

How does this work financially?

- I'm not sure, but I imagine something like this…
- The cards are simple and cheap to make - I'm hoping less than $1 each, for 2mb cards, 16mb might be more, but component prices can change.
- Thonk will sell blank cards individually or in bulk
- The card design will be open source - take the gerbers or kicad files and get them made wherever you like - particularly if you want to change the design cosmetically or otherwise.
- You can sell cards however you like - I'd imagine Bandcamp or instagram DMs or Shopify or whatever works for you.
- Or you can sell or give away the .UF2 files, so users can burn their own cards over USB. I love the idea of physical cards folded into fanzine-style paper instruction sheets, but a quick download might also work.
- To be clear - neither Music Thing Modular nor Thonk seek to make money from card sales. I just want to see lots of cool cards in the world.

What about IP protection?

- It's up to you
- I'd love to see all the code shared so the community of developers can improve as fast as possible

- But if you don't want to do that, it's fine
- I don't know how easy it is to extract the binaries from a flash chip, I'd be surprised if it was very difficult, and maybe that could then be decompiled.

Will the computer module be available as a single Eurorack module?

- Almost certainly, but not until after the Workshop System is released.
- Both the Workshop System and the Computer will be sold first as kits, then as built units, all via Thonk.

How do I duplicate a load of cards?

- Not sure yet, you can probably do them pretty fast using the module with a script and the debug header, or you might be able to get them written at the PCB fabricator.

How will people learn about my card?

- Thonk and Music Thing Modular will be very motivated to talk about and share and promote cards, because they make the Workshop System more useful
- I'm hoping to develop a strong community around the Workshop System, probably based on email bulletins and Discord.
- We'll probably develop some kind of listing or catalogue, not sure how that will work yet - let me know if you've seen anything similar that works well

How do I get a Computer dev kit?

- At the moment, complete Workshop Systems are very rare - I'm building them myself. However, Computer is a separate PCB, so I will be able to start shipping them out soon-ish.
- This isn't perfect - the most interesting cards might be the ones that interact closely with the rest of the Workshop System, but hopefully is better than nothing
- Please fill in this form to request a dev kit. It might take me a while but I want to get as many of these into people's hands as possible. https://forms.gle/cpiYJzhsmtwwMBes6
- You can also join the Discord here: https://discord.gg/j79Dk88Dms

Timeline

- Late Feb 2024: Spec and a small number of prototype boards released
- Mid April 2024: Dyski Workshop in Falmouth, first time Workshop Systems are used by musicians
- Summer 2024: Beta testing.
- October 2024: Public launch
- November/December 2024: Workshop Systems on Sale
- As people start to get Workshop Systems, the audience for cards will expand, so there's not a huge rush. A nice idea in March 2025 will be more valuable than one in October 2024.

Computer Board Version History

Check the back of your Computer PCB to see what revision you have.

Proto 2.0.2 / Rev 1.0.0 This is the September 2024 version, avaliable with the Proto 2 systems used at Machina Bristronica and shipped in first batch of systems

- Fixed LED issues from Proto 2.0
- Added 8k i2c EEPROM on GPIO16 (SDA) and GPIO17 (SCL)
    - Connects to Zetta ZD24C08A EEPROM, clone of 24C0* chips. Contains 8K bits (1024 X 8).
    - SDA & SCL lines have 2.2k pullups.
    - Data is stored in 8 x 1024 bit pages addresses 0x50 to 0x5B
- Cosmetic silkscreen changes between Proto 2.0.1 and Rev 1.0.0

Proto 2.0 This is the July 2024 version, available in very limited numbers due to wonky LEDs

- Cleaner power for RP2040 ADC: 80hz low pass RC filter on the ADC power pin (200Ω + 10uF to ground)
- Filtering on Audio inputs to ADC: 16kHz RC filters (1KΩ + 10nF to ground)
- Normalisation probe: GPIO4 is attached to all socket switched inputs via 100K resistors and BAT54S protection diodes.
- Board identidfication bits: GPIO Pins 7, 6, 5 = binary bits connected to +3.3 or GND
    - 0 0 0 = Previous boards (pins floating)
    - 1 0 0 = Proto 2.0 this board
- Improved PWM filtering: changed from Sallen Key to Multifeedback for reduced 60kHz ripple
    - NB: PWM CV outputs are now inverted
- SMD LEDs: 0603 red LEDs on the board (together with thru-hole footprints) so this layout could be used with light pipes vs 3mm leds.
- LED Buffers: Incorrect use of CD4050BPWR, so LEDs don't work at all on this board

Proto 1.2

- This is the May 2024 version shipped out as a Developer Kit to more people
- DAC Issues are corrected: For this version, Gain should be set to 1x like this:
    - `#define DAC_config_chan_A_nogain 0b0011000000000000`
    - And the output will be -6v to +6v. If you use 2x gain, the output will swing from +6 to -11v
- Other changes

    - This version has GPIO Pins 0 and 1 - the default UART pins - pulled out to two pins by the LEDs
    - Other changes were non-functional layout changes -- I'd misunderstood the JLCPCB design rules so the components in Proto 1 were a bit tight

Proto 1

- This is the version in all the Dyski Workshop systems, and the version that the very earliest developers have
- DAC Issues: In this version the MCP4822 gain structure is incorrect, so the gain needed to be set to 'double' and the output was not centered properly.
- For example `#define DAC_config_chan_A_gain 0b0001000000000000`