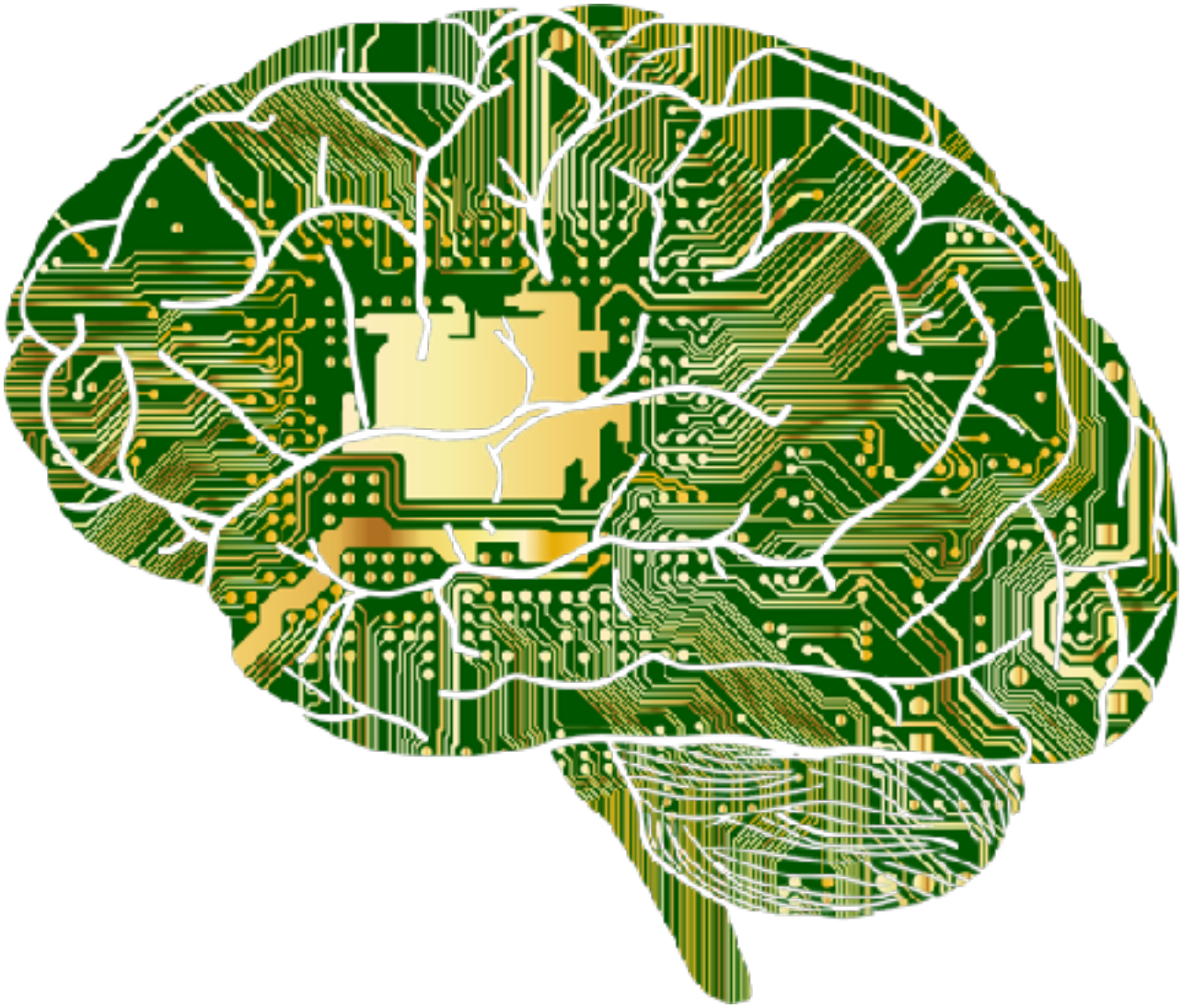
gigatron

The TTL based microcomputer system



ASSEMBLY MANUAL / USER MANUAL

(includes Pluggy McPlugface)

Thanks to openclipart.org for most of the clipart in this manual.

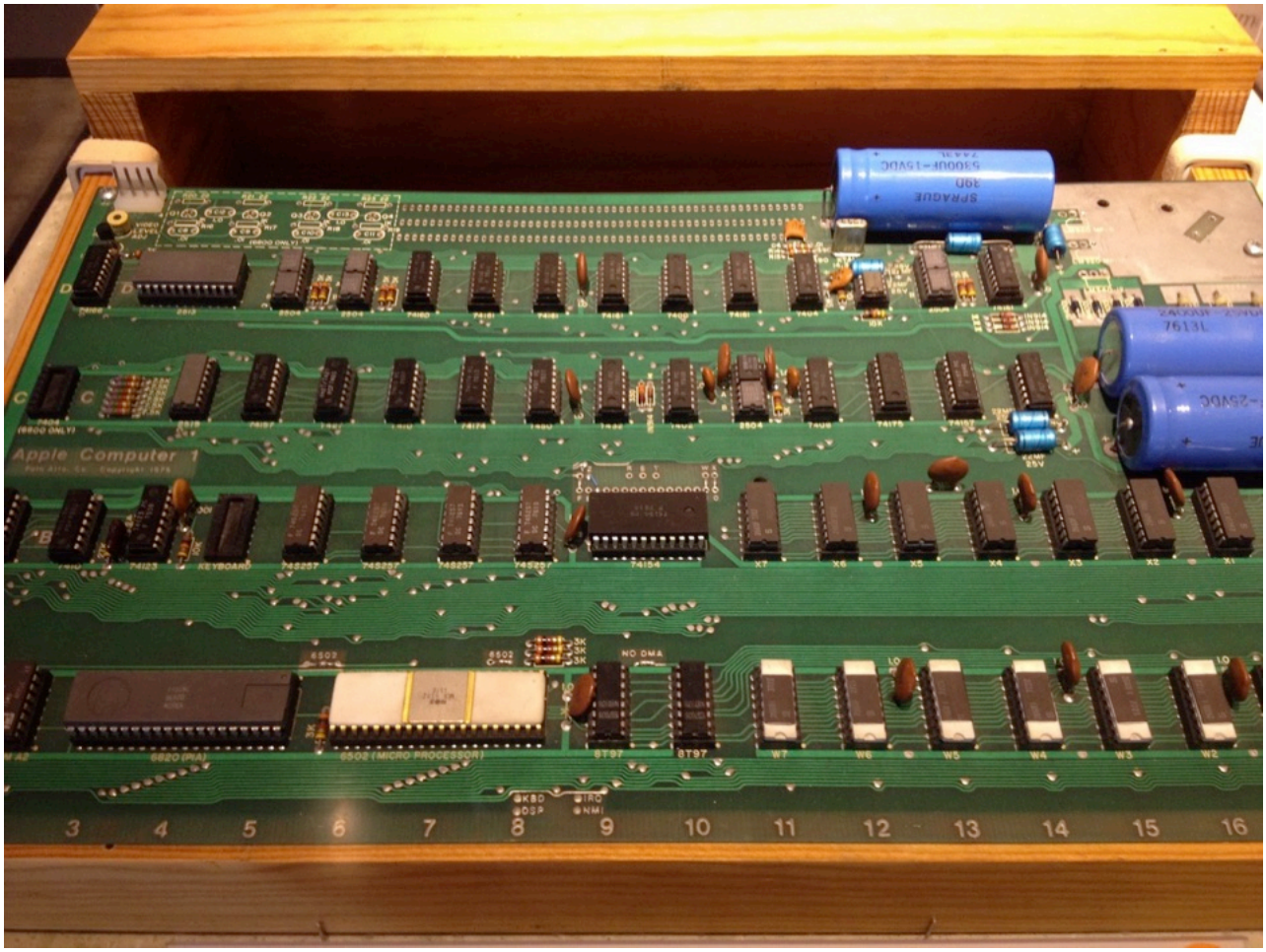
0

Introduction to gigatron

Thank you for your interest in the gigatron!

In 1975 Steve Wozniak famously designed the Breakout arcade game out of 44 simple chips, without using a microprocessor, simply because those weren't available to him at the time. When one year later the MOS 6502 and Zilog Z80 were launched, his Apple 1 started the microcomputer revolution. The debate still rages about which processor was the better one. But more interesting is to investigate if these devices were really necessary for the personal computer revolution at all: what would have happened if they had never appeared?

Here is the Apple 1 computer:



The large white chip is a 6502 CPU, running at 1MHz.

We chose to build a general purpose computer using simple TTL (transistor-transistor logic) integrated circuit logic, the main components used in computers in the 1970s. These components are still available on the market today. To make it interesting, we do not include a 6502, or any other microprocessor in our design! Our CPU is made out of a handful of TTL chips instead of a microprocessor. With this you can look inside the CPU, follow how data travels through it, and learn how computers really work at their lowest level. The fun is that this TTL CPU is still very powerful. We even believe it is faster than the first IBM PC.

Because we like to tinker with hardware and guess that you do too, we decided to make gigatron available as a do-it-yourself soldering kit, instead of it just being a one-off study object. The kit was sold from 2018 until 2020. Meanwhile, everything (the hardware, the software and this manual) have been open sourced. This assembly manual was included in printed form in the kit, to make buyers able to build their own working microcomputer and play video games on it. We have open sourced the manual as is, so it is still written as if you had bought a kit. You can use it as a reference, to learn more about the gigatron or to help in sourcing the components and build your own version.

The gigatron team,

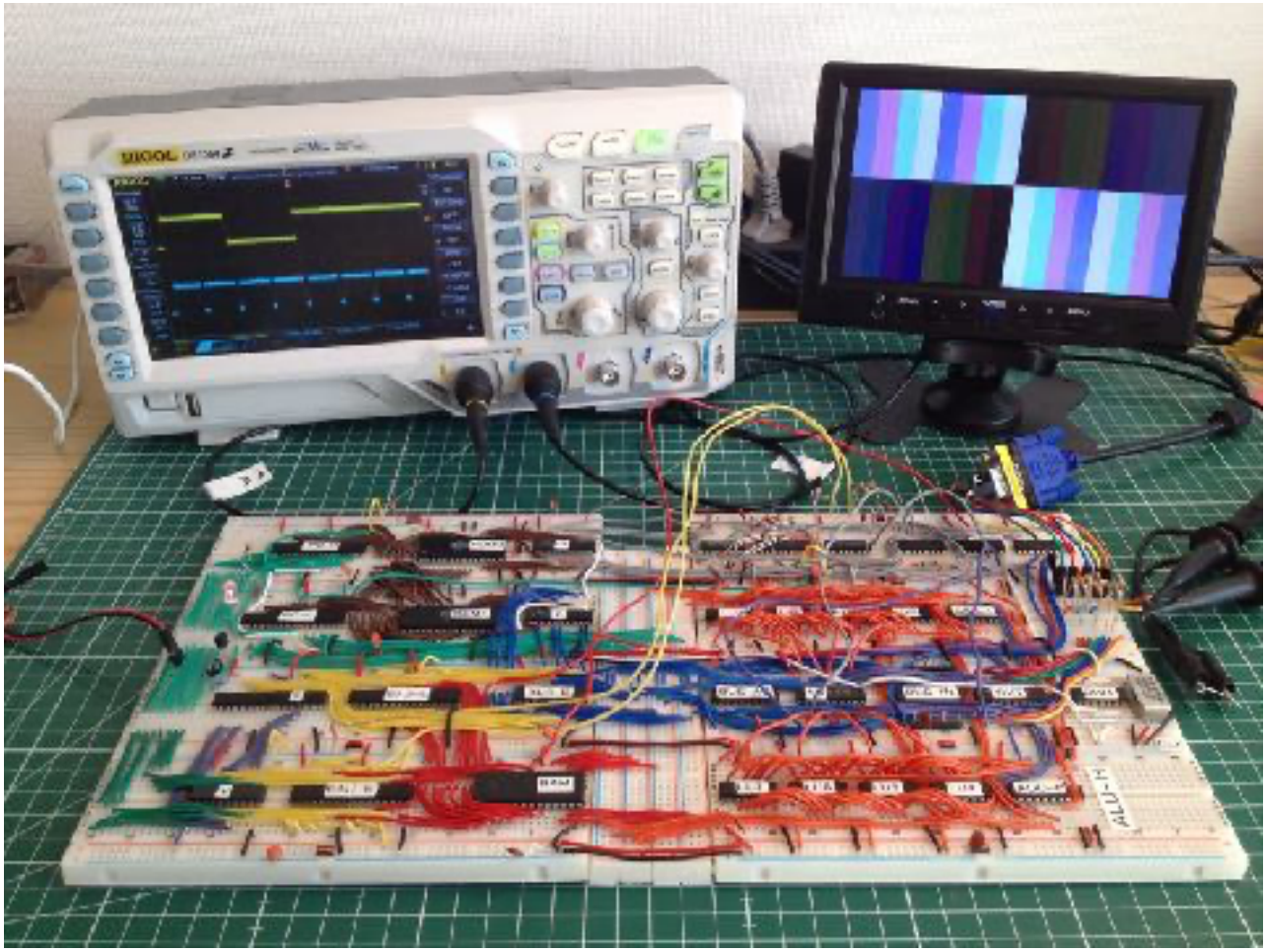
Marcel van Kervinck

Walter Belgers

Operating conditions:

- Power supply: 5.0V (±0.25V) DC, 100mA
- Operating temperature: 18-27 ºC, 65-80 ºF

Disclaimer: use your mind and common sense when building and using this kit. It contains small parts that present a choking hazard to small children. Hobby electronics might unintentionally interfere with radio signals or disturb other electronic devices. Therefore don't use this kit for other purposes than personal entertainment and to learn from it. For example, don't embed it in a product, or worse, use it as part of a safety-critical system. Switch off the device when leaving it unattended. Don't use a cheap USB power adapter, but one from a proper brand.



First video output on the breadboard

1

How to use this manual

The gigatron needs to be assembled before it can be used. In this assembly manual, we explain, step by step, how to build and test it. We also provide background information about the working of the components and the system as a whole. More in-depth information will be provided on our website https://gigatronttl.eu/.

If you are not familiar with electronic components, you can read chapter 2. This will give an understanding of what the components look like, what their function is and how they should be soldered. You can skip this chapter if you are already familiar with electronics.

Chapter 3 is used to get prepared for the actual building. It contains a checklist to make sure you have all the components, and tells you what tools are needed for assembly.

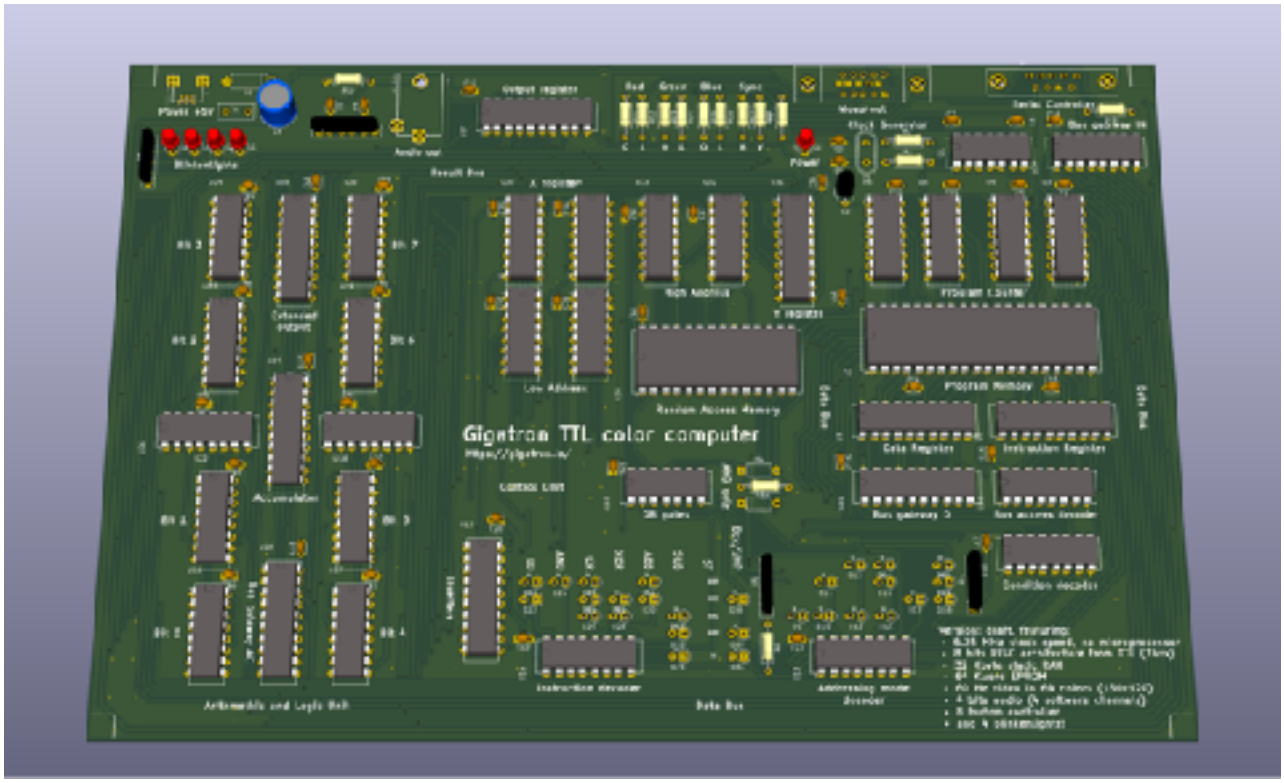
Chapter 4 has a quick course in soldering. If you are familiar with soldering already, you can skip chapter 4 and start the actual build.

Chapter 5 explains the process of building the gigatron and the PS/2 adapter, in easy to follow steps, testing along the way. Once you have finished chapter 5, you have a fully functioning computer!

Chapter 6 contains a short manual for using the software that is shipped with the gigatron.

The appendix includes gigatron schematics. If you are a hardware person, you will appreciate how we have designed the computer to work well with a minimal amount of components. At https://gigatronttl.eu/walkthrough you will find a one hour walkthrough of the hardware.

Be sure to check the forum at https://forum.gigatronttl.eu/ for more technical details and upgrades!



Prototype board

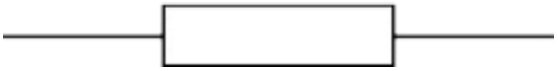
2

Crash course in electronics

Understanding what the components do

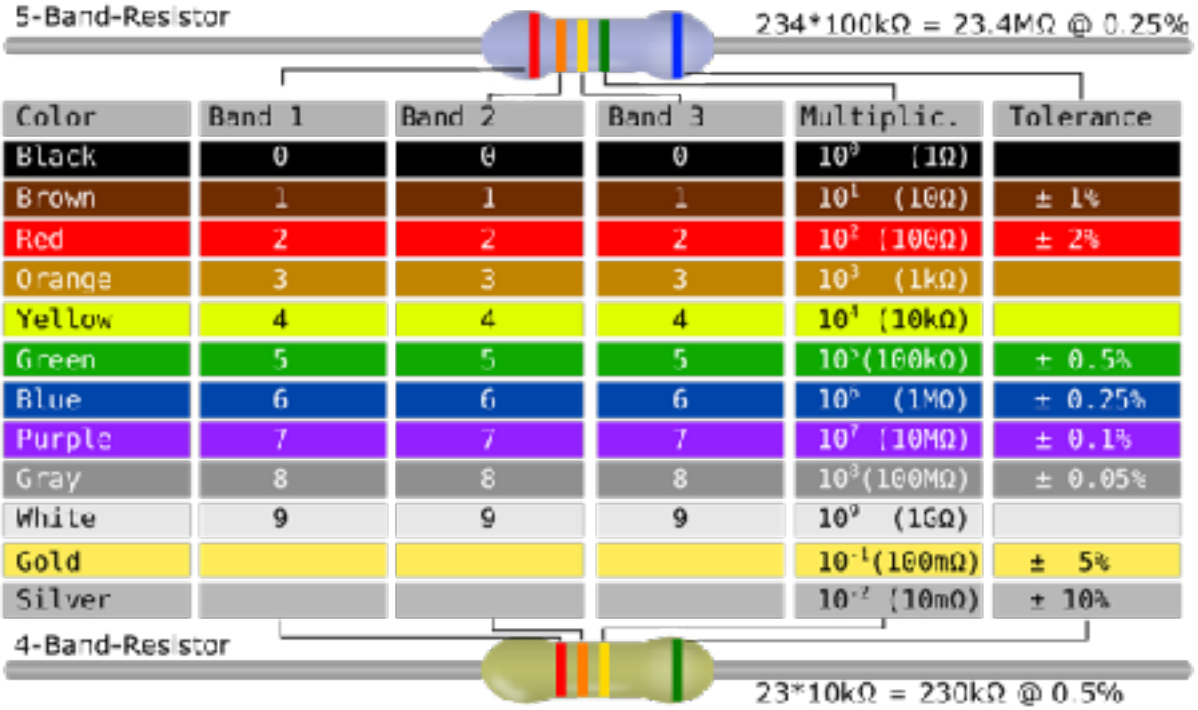
The first thing we are going to do, is to explain how to identify all the parts in the gigatron kit and explain what their function is. If you are already familiar with electronic components, you can skip this chapter and move on to chapter 3 to make sure you have all the components in your kit, before you start soldering.

Resistors



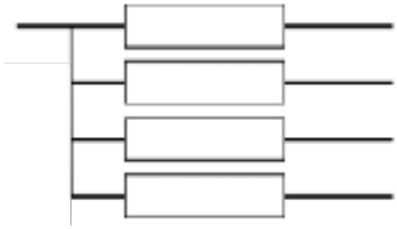
Resistor symbol

Resistors provide electronic resistance, useful for e.g. reducing the current flow. Resistors have two wires and can be installed either way. On the resistor, you will find an "electronic color coding" describing the amount of resistance it provides, measured in ohms (Ω). The number of colored rings and thus numbers, can differ per resistor. We have included only resistors with four colored rings, three that are evenly spaced and one a bit further away, which encodes the tolerance. The colors translate to values according to the following schematic:

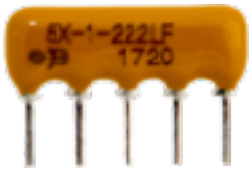


As you can see, the first two bands translate to a number, the third specifies the number of zeroes to add to that number. The 4-band resistor shown, has a red and orange band, meaning "23" followed by a yellow band, which means 4 zeroes need to be added, resulting in a value of 230000Ω or 230kΩ for short.

In the kit, you will also find a resistor array. This is a set of resistors that are connected and put in a single package, like so:



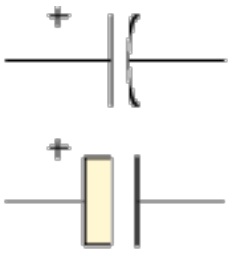
The array will look something like this:



There is always a marking (dot, stripe, …) at the pin that is common for all resistors. This means that it matters in what orientation you solder them in the PCB. "PCB" stands for Printed Circuit Board, the board that holds all the components and interconnects them.

Capacitors

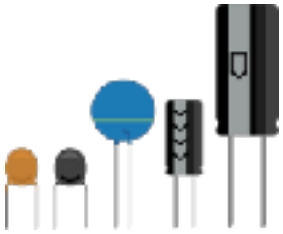
(Ceramic) capacitor symbol



Electrolytic capacitor symbol

A capacitor can store electrical energy, just like a tiny battery. The amount of charge they can hold is measured in farads (F). In most cases, capacitors with only fractions of a farad are used, such as µF (microfarad), nF (nanofarad) or pF (picofarad). They are useful for e.g. stabilising voltages. If the voltage increases, energy is stored. When it decreases, the energy is released. This is why you will see capacitors everywhere there is an integrated circuit.

In the kit, we included two types of capacitors: a number of ceramic capacitors and one electrolytic capacitor.



In the picture above, we see ceramic capacitors on the left. These are normally small, in the pF to nF range. They have no polarity, meaning they can be used either way around. The electrolytic capacitors have higher values, in the µF range. They are polarized, meaning it matters in what orientation you solder them in the PCB. The white band, in many cases filled with minus signs, points to the negative side. On the PCB, we have marked the orientation of these capacitors.

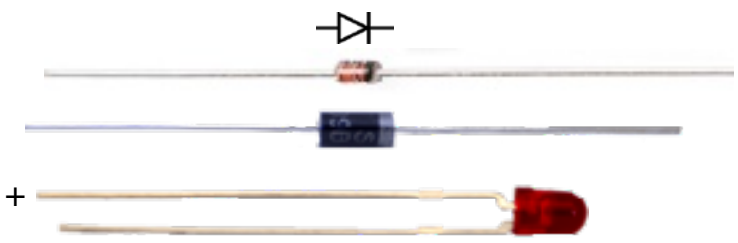
On the electrolytic capacitors, the value is written on them in legible, albeit tiny, form. For the ceramic capacitors, there are several systems in use. The capacitors we supply, have numbers such as "104". This "104" translates to $10x10^4$ (=100000) pF, which is 100 nF.

Diodes, zener diodes and LEDs

Diode symbol *Zener diode symbol* *LED symbol*

A diode is a component that allows current to flow from one side to the other, but not the other way around. In the pictures above, current flows from the "anode" on the left to the "cathode" on the right. A zener diode has a special characteristic: if the voltage goes beyond a specific value, the diode will allow current to flow in both directions. In the gigatron kit, we use a zener diode as protection: if the voltage is too high, the diode causes a shortcut, protecting the circuit. Thus it is normally not conducting any current. A light emitting diode, or LED for short, is also a diode, which emits light if current is flowing through it.

For diodes, the orientation is important, so it matters in what orientation you solder them. Diodes and zener diodes have a bar printed on them, which marks the cathode. LEDs have one long and one shorter pin. The long pin is the anode, the short is the cathode.

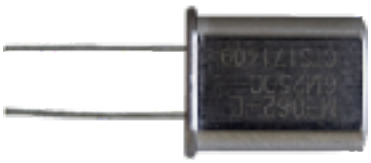


In the kit, you will find that the diodes are clear and the zener diode is black, to easily distinguish them.

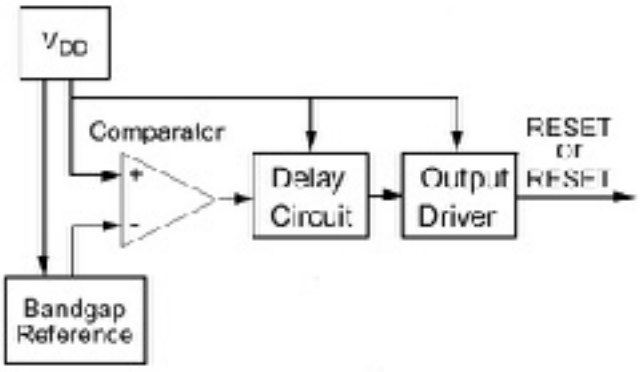
Quartz crystal

Quartz crystal symbol

A crystal contains piezoelectric material that mechanically resonates at a specific and very precise frequency. We use a crystal to provide a clock signal on which the computer operates. A crystal oscillator has no polarity: it can be soldered either way.



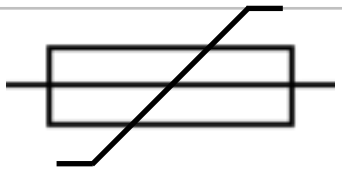
Supervisory circuit



The supervisory circuit consists of a number of components put into one small package. The circuit reacts upon a certain input voltage. In the kit, it is used at power up, to bring the gigatron in a known state: it sets the program counter to zero to start with the first instruction in memory. After supplying power, the supervisory circuit will hold the program counter at zero, until the voltage is stable at the right level. This makes sure we know what code the computer is executing after switching it on.

The supervisory circuit is in a package that looks like a transistor (see below). The circuit needs to be soldered in the right orientation. Since we do not use any transistors in the kit, it is clear what component is the supervisory circuit.

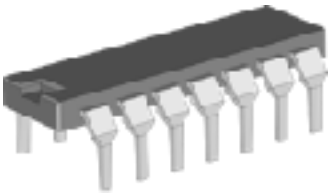
Resettable fuse



Resettable fuse symbol

A fuse is a device that, once the current flowing through it exceeds a certain value, stops conducting. It is used to protect a circuit from too much current. A normal fuse needs to be replaced once it is tripped. A resettable fuse resets itself once it is cooled down, so it can be reused. A fuse has no polarity: it can be soldered either way.

Integrated circuits and sockets



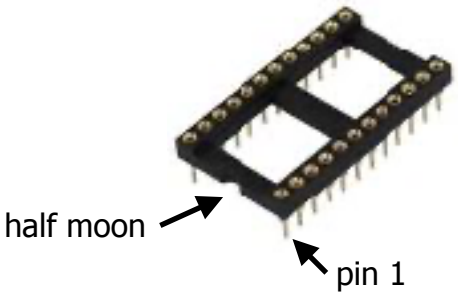
The main components within the gigatron kit are integrated circuits (IC, also called microchips or chips). ICs contain many tiny components such as transistors, etched onto a small piece of silicon. ICs can perform complex tasks: the whole microcomputer you are building could be made of one special purpose IC, but where is the fun in that? The gigatron kit contains only fairly simple ICs that have been around for many years. The 7400 series of ICs that the kit uses, were also used to build the computers of the 1960s and 1970s and they are still used to this day. They contain logical building blocks, such as (N)AND and (N)OR gates, counters, multiplexers etc.

Modern IC packages are very small and cannot easily be soldered by hand. We use ICs in the original form factor, called DIP (dual in-line package). It features a rectangular plastic or ceramic housing and two rows of pins. The pins go through holes in the PCB and are soldered on the other side of the PCB.

The pins of an IC are numbered. If you put the IC in front of you with the part number readable, you will find that on the left side there is a small half-moon shape cut out of the housing. This indent marks where pin number 1 is, in this case bottom left. There can also be a dot near pin one. The dot can be printed or cut out. It is also possible to have

both the half-moon shape and a dot. The pins count anti-clockwise, so the last pin is opposite the first one. On the PCB, there is a marking on the side where the marking on the IC should go.

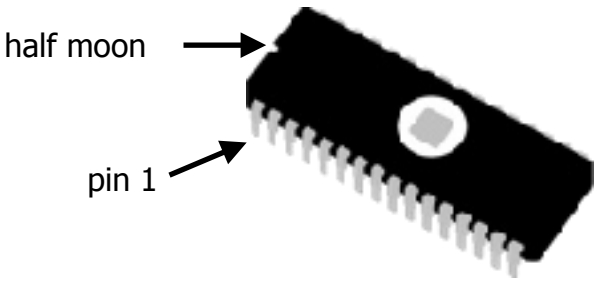
ICs can be soldered directly into the PCB, or put into sockets. Be careful to insert them in the right position.



An IC socket also has a half-moon shape cut out, which marks the side where pin 1 can be found.

Apart from the logical ICs, there are also ICs with RAM and ROM. The RAM IC is not very different from the others. The ROM that we use is an EPROM (Erasable Programmable Read Only Memory), which means the contents can be erased (using UV lighting) and reprogrammed (using a special programming device). This IC has a little glass window that exposes the silicon die underneath. The glass window should be covered to prevent erasure of its contents by sunlight.

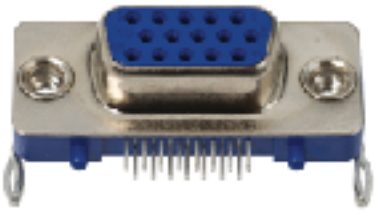
The Pluggy McPlugface PS/2 adapter contains a tiny microprocessor with built-in memory. It only has 8 pins.



Mini-USB power connector

Gamepad/Pluggy McPlugface connector

Audio connector



VGA connector

Pluggy McPlugface connector

PS/2 connector

Several connectors are included in the kit. These make it possible to hook up the power supply, the monitor, the gamepad that is included and an audio device. The Pluggy McPlugface adapter contains a connector that fits the gamepad connector and a PS/2 keyboard connector. All connectors have extra thick pins that need to be soldered, that provide a sturdy mounting of the connector to the PCB.

The gigatron needs a 5 volt power supply. Since USB power is widely available and provides 5V, we have chosen to use USB to power the device. The USB connector does not provide any data capabilities, it is purely for supplying power. The connector has pins for data, but these are not used and not soldered. If you power the kit with a power bank, you might find that the system shuts down after a while. This is due to the fact that the gigatron uses so little power that the power bank thinks nothing is connected. If you connect a USB-stick to the power bank, together with the gigatron, the power bank should keep on working.

First working PCB

3

Before you begin

Before starting to build the gigatron and the PS/2 adapter, we need to first make sure that all components are present and that you have all the tools.

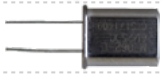
Below is the list of parts. Use this list to check if the kit as it has been sent to you is complete. Because of the way these components are packaged, you might find you have more components than needed, e.g. a few more than 30 diodes. You can do whatever you like with what is left over. There are two bags with components. One holds the components to build the actual computer, the other contains only a few components that you use to build an adapter to hook up a PS/2 keyboard to the gigatron.

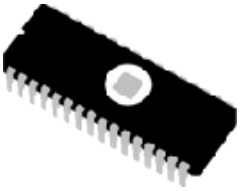
The pictures are meant as illustration only, the actual components may look slightly different. There could be slight variations in the type as well, e.g. if the list mentions "74HCT00" and you have an "M74HCT00B" that will be okay.

The ICs and sockets are pressed into conductive foam in the same lay-out as on the PCB. This makes it possible to see in one look if all ICs are there. The RAM and ROM ICs may have been put in their sockets already for you. In that case, you can simply leave them like that.

A. The Gigatron itself:

Picture	Type	Description	Amount	Mark ☑ if present
	PCB	Circuit board	1	☐
	68Ω	Resistor (blue-grey-black)	2	☐
	750Ω	Resistor (purple-green-brown)	3	☐
	1kΩ	Resistor (brown-black-red)	1	☐
	1.5kΩ (1k5)	Resistor (brown-green-red)	3	☐
	2.2kΩ (2k2)	Resistor (red-red-red)	5	☐
	1MΩ	Resistor (brown-black-green)	1	☐
	10kΩ	R2R network (6 pins)	1	☐
	2.2kΩ	Resistor array (5 pins)	3	☐

Picture	Type	Description	Amount	Mark ☑ if present
	1N60P or BAT42	Schottky signaling diode	≥30	☐
	5.6V/5W	Zener diode	1	☐
	1.8V, 2mA, 3mm	LED	5	☐
	47pF	Ceramic capacitor	3	☐
	100nF	Ceramic capacitor	40	☐
	220µF	Electrolytical capacitor	1	☐
	6.25Mhz/30pF	Crystal	1	☐
	hold 0.2A/trip 0.4A	Multi-fuse	1	☐
	MCP100-450DI/TO	Supervisory circuit	1	☐
	40 pin	IC Socket	1	☐
	28 pin	IC Socket	1	☐
	74HCT04	Hex inverter	1	☐
	74HCT32	Quad 2-input OR	1	☐
	74HCT138	3-to-8 Decoder	2	☐
	74HCT139	Dual 2-to-4 decoder	1	☐
	74HCT153	Dual 4-to-1 multiplexer	9	☐
	74HCT157	Quad 2-to-1 line data selector	4	☐
	74HCT161	4-bit Presettable counter	6	☐
	74HCT240	Octal inverting buffer	1	☐
	74HCT244	Octal bus driver non-inverted	2	☐
	74HCT273	8-bit D-type register	3	☐
	74HCT283	4-bit Adder	2	☐
	74HCT377	Octal D-type flip-flops with common enable	3	☐
	74HC595	8-bit shift register	1	☐
	62256-55	32Kx8 SRAM	1	☐

Picture	Type	Description	Amount	Mark ☑ if present
	27C1024 (a golden sticker might obscure the part number)	64Kx16 EPROM (pre-programmed)	1	☐
	mini-USB	Power connector	1	☐
	15-pin sub-D female	VGA connector	1	☐
	9-pin sub-D male	Gamepad connector	1	☐
	3.5mm jack	Audio connector	1	☐
	mini-USB to USB type B	Cable	1	☐
		Game controller	1	☐
		Enclosure	1	☐
		Little rubber feet	8	☐

B: The PS/2 adapter (Pluggy McPlugface):

Picture	Type	Description	Amount	Mark ☑ if present
	PCB	Circuit board	1	☐
	IC socket	8-pin socket	1	☐
	ATtiny85	8-bit microcontroller (pre-programmed)	1	☐

Picture	Type	Description	Amount	Mark ☑ if present
	PS/2	Keyboard connector	1	☐
	9-pin sub-D female	Gamepad connector	1	☐

We have done our best to make sure that all kits sent out have at least the number of components needed. If, for whatever reason, you find that something is missing, please do not start building your kit, but contact us first at support@gigatronttl.eu.

Apart from the components, you will need the following tools:

- A soldering iron with a round tip, preferably 40~60 Watt.
- Solder containing flux.
- Optional: desoldering braid/wick or a desoldering pump.


See the next chapter if you need advice on what soldering iron or solder to choose.

Furthermore you will need:

- A multimeter.
- A pair of (small) side cutters.
- A magnifying glass (or you can use the camera on your smartphone).
- Optional: a pair of needle-nose pliers.




To prevent damage to the electronics, you could use an ESD mat, but this is optional.

If you do not have one, just use some precautions:

- Avoid a low humidity environment, e.g. when you can feel static discharge when touching doorknobs.
- Do not place the PCB and components on carpet.
- Before you start building the kit, touch a grounded metal object in your house once. This can be a metal radiator or a faucet for example.

4

Quick soldering course

And how to place components on the PCB

If you have soldering skills already, and have the tools listed in the previous chapter, you can skip this chapter and move on to the next, in which we describe how the kit is assembled.

Choosing a soldering iron

Soldering irons come in different shapes and forms. The cheapest are the pencil-shaped soldering irons. Better are the soldering stations. We do not recommend using a soldering gun, they are not very good in doing fine work.

A soldering iron has a certain wattage. Irons with lower values of 20~30 Watt will quickly lose heat while soldering and are not recommended and a bit inconvenient. A wattage of 40~60 Watt is preferred. The soldering iron should have a round tip. If you plan on doing more soldering, consider buying a soldering station. It heats up faster and probably comes with a stand and a sponge for cleaning the soldering iron.

Choosing solder

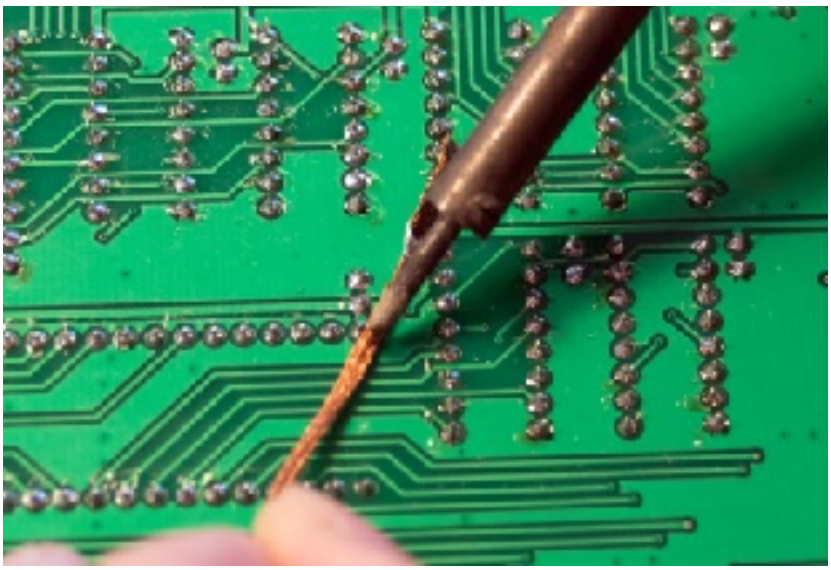
Solder also comes in different shapes and forms. There is lead-free solder and solder containing lead. The lead-free solder contains other additives that are not necessarily better for you when inhaled. Solder containing lead is easier to use, it melts at a lower temperature and is easier to work with, but may not be available in the place where you live. Soldering wire comes in different thicknesses. For this kit, 0.032” or 0.8mm diameter wire is fine, but a somewhat thinner or thicker wire will also work.

The image on the next page shows solder with lead. Sn63/Pb37 means 63% tin and 37% lead. You can also see the marking “2.2% Flux”. This solder has a core of what is called

"flux". Flux is needed to get things you solder together to actually bond. Solder forms a metallurgical bond by dissolving and chemically reacting with the base material. That base material, i.e. the metal pins on the components, reacts with oxygen in the air to form a tiny oxidized layer on the pins. This prevents the solder from bonding to it. The "flux" can be made from different elements. Rosin flux is made from the sap of pine trees. What does it do? At room temperature, nothing really. It does not conduct electricity, so it is no problem having it on the PCB. At higher temperatures however, it becomes acidic and removes the oxidized layer. This allows the solder to bond. When it cools down, it again becomes harmless, although it can leave a little residue. You can wipe it off if you like, but this is not needed and normally not done.

So, the flux is crucial in getting good solder connections. The amount of flux in the solder can vary, but this is not very important. Regular solder with flux will be fine.

Which ever solder you use, be sure to wash your hands afterwards and try not to inhale the fumes. The fumes contain all kinds of elements coming from the flux that are just as bad as cigarette smoke. The lead does not evaporate because the soldering iron does not get that hot.



Using desoldering wick

Desoldering braid/wick

If you need to remove solder, for instance, when you soldered a component the wrong way around, you could use desoldering braid. This is basically a mesh of thin copper wire that sucks in the solder. Place it over the solder that needs to be removed and heat it with the soldering iron. There is no need to cut off a piece, just use the end of the roll. The desoldering braid will turn silvery when it absorbs the solder. Keep using new pieces of desoldering braid until enough solder has been absorbed.

There are also desoldering pumps, but the braid is easier to work with. Still, it requires practice before you can desolder an IC. It is easier to cut all pins of the IC and then remove them. We can supply a new IC if needed.

Soldering

Turn on the soldering iron or station. If you can select a temperature, select something in the range of 600~700ºF or 320~370ºC. The higher the temperature, the quicker you can and need to solder. Do not put the soldering iron on the components for too long, as overheating may cause them to fail.

If your soldering iron is brand new, put some solder on the tip. If the soldering iron has been used before and is dirty, you can clean it by taking away the dirt with a moist sponge in quick movements.

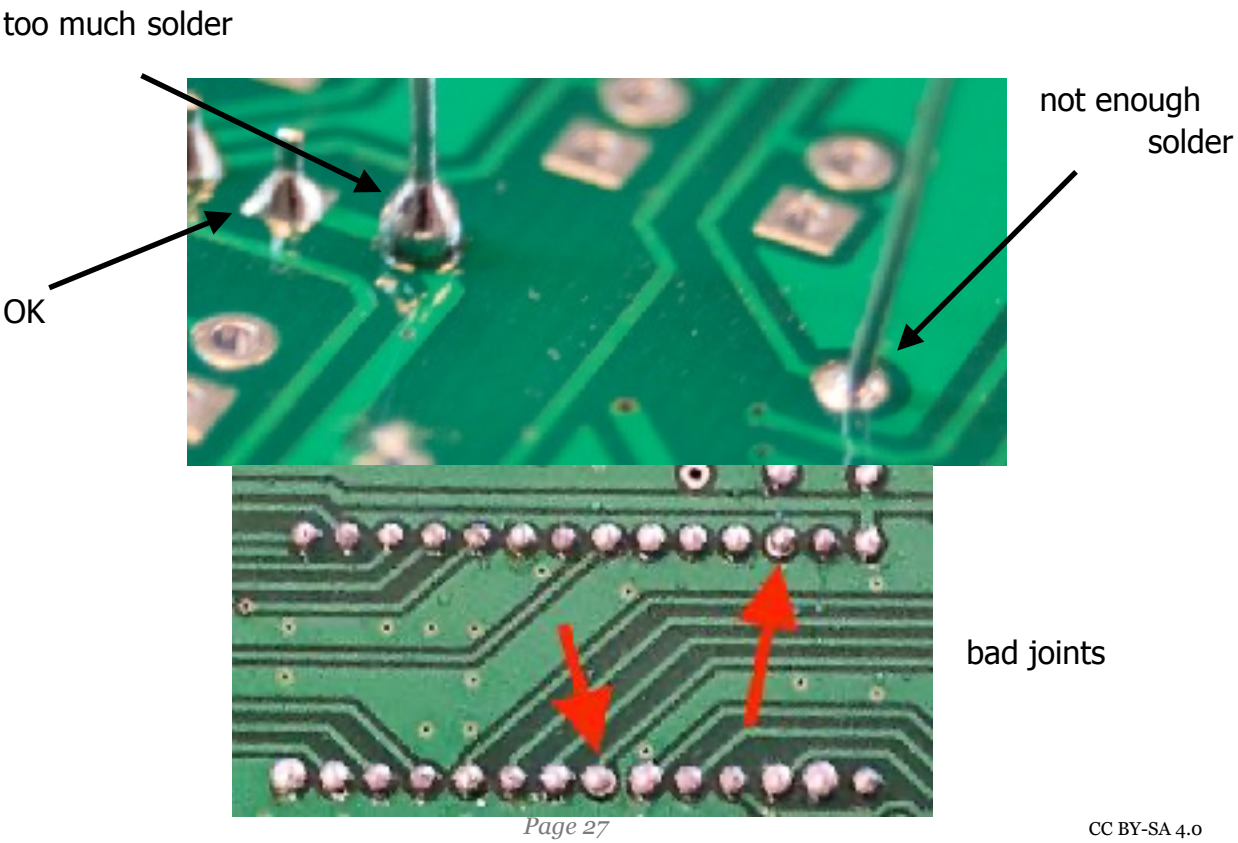
Before we can solder a component, we must put it on the PCB. In the gigatron kit, we use 'through-hole' components. These were once in common use, but nowadays they are almost universally replaced with SMD (surface mount device) components. Through-hole components can easily be soldered by hand.

Some components will fit into the board right away, some have pins that need to be bent, such as resistors. Take the resistor in your hand and use your nail to bend the pin, just a little bit away from where the pin begins (1 or 2 millimeters), like so:

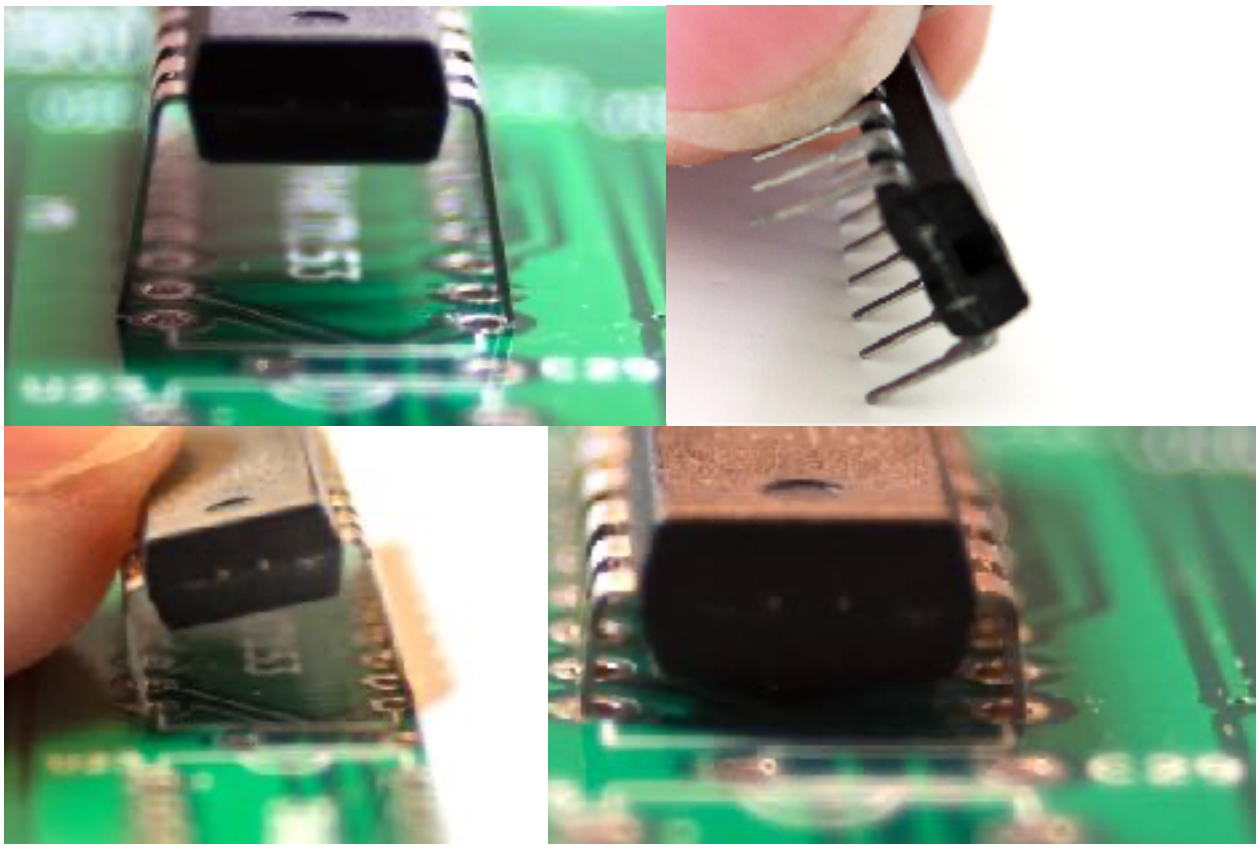
Now you can place the resistor (or other component) into the PCB (on the printed side). On the bottom of the PCB, the pins stick out. Resistors and some other components sit on the PCB. If you bent the pins correctly, the resistor should be in place. You can push the component down if it "almost" fits, the pins will then bend the last bit to match the holes.

Slightly bend the pins to the side, so the resistor will not fall out when you flip the PCB around:

Now you can start to solder. Make sure the soldering iron is up to temperature and be sure not to touch the metal half as it gets hot. Hold it against the PCB like a pen, just where the pin sticks through and heat it up for a second. Then, use your other hand to add solder, that will start to melt. Add the solder at the joint of the pin and the PCB, do not put it on the soldering iron. Remove the solder and remove the soldering iron.



The pins of ICs also need to be bent to fit. New ICs have pins that are slightly slanted. IC sockets have straight pins and can be soldered right away. For ICs, a trick is to place the ends of the pins on a flat surface such as a table and carefully press them straight. Take care as this requires only a gentle push. The little push on both sides should straighten the pins so you can insert it easily into the PCB.



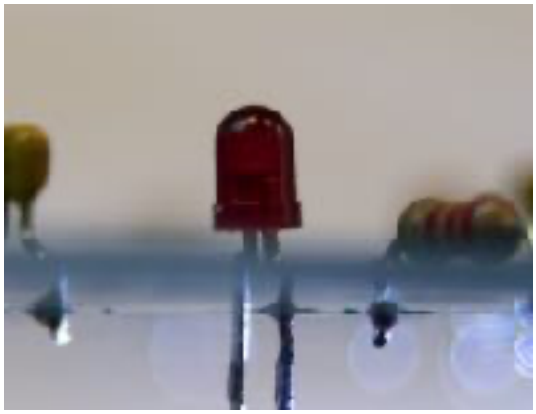
So, the order is:

- briefly heat up both the PCB's metal ring and the wire or pin
- apply solder
- remove solder
- remove soldering iron

You should now have successfully soldered the component onto the PCB! Make sure the solder joint looks smooth. There should not be any holes. Those happen when you do not use enough solder or when the temperature is too low. Neither should there be too much, this could cause electrical shorts with adjacent components. Cut away the excess wire, if there is any.

Next, check if the component is placed correctly, i.e. they sit on the PCB. If not, reheat the faulty joint and **gently** push the component in when the solder is liquid. If you push too hard, you will damage the PCB as the traces will come loose!

This can be handy for some components like LEDs that, in most cases, will never end up in the correct position on the first try. Here's the trick: you insert the LED into the PCB and solder one lead with just a little bit of solder. It is now probably not where you want the LED to be.



Next, apply heat from the soldering iron while, again **gently**, pushing the LED in the correct position. When you are satisfied, solder the other lead and re-heat the first one to make sure both leads are soldered well.

The same goes for ICs and IC sockets: solder one pin with a little solder, then the opposite pin. Next, see if it is placed correctly. If not, re-heat and **gently** push the component into position. Re-heat both joints and solder the rest.

Most components will go into the PCB up up to a certain position, like resistors, electrolytic capacitors, LEDs and ICs. But, transistors and some ceramic capacitors need to have a little bit of room between the PCB and the component. For the supervisory circuit, keep a bit of space between the PCB and the component. For ceramic capacitors, make sure that the whole capacitor itself is above the PCB.

If a bit of solder drops onto the PCB and forms a little ball, you can use your fingernail to remove it.

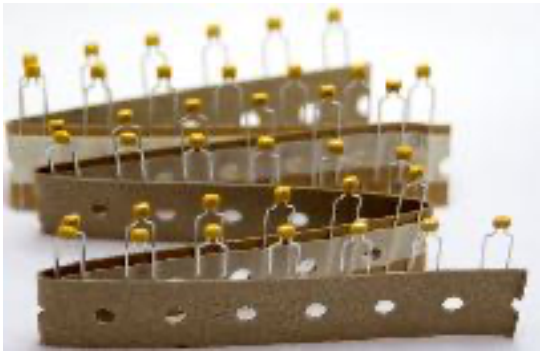
5

Assembly and testing

How to build your gigatron

This chapter explains how to build the kit, part by part. Before building, make sure that you know how to solder (see chapter 4) and have checked that you have all the components (chapter 3). There's quite a lot of components, but don't be intimidated, we will be soldering them part by part. Building the gigatron will take about 3 to 4 hours. As an alternative to using this chapter as assembly guide, the video instructions at https://gigatronttl.eu/build can be used.

1. To practice with soldering, we start by soldering **40 ceramic 100nF capacitors**, marked C5 through C44. There are at least 40 provided in the kit. Do not confuse them with the three 47pF capacitors. Only put in from C5 upwards, we will deal with C1 up to C4 later. C5 is located on the top, just below "Video Out", the rest is approximately clockwise on the board.



This is also a good time to remind you that the components are all placed on the printed side of the board (the front). The outline of the components is printed on the board to help you orient them correctly. The soldering is done on the backside!

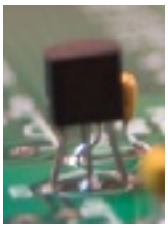
Another thing that is good to know, is that some pins will be easy to solder, whereas others seem to take more effort or time to heat up. This is due to the fact that sometimes, a pin has a single trace to another components and in other cases, the pin connects to the ground plane. This means there is a large metal surface connected to the pin, which causes the heat from your soldering iron to dissipate more quickly. So it is to be expected that the time it takes for the solder to melt differs for every solder joint.

After the capacitors are soldered in, cut away all the excess wire. (This is necessary for the resistors, capacitors, LEDs, crystal, multi-fuse, diodes and supervisory circuit.)

Check that there is no short circuit, by measuring the resistance between the +5V and Ground. One place to find these is by connecting to the two pins of C4 (which is not yet soldered in) on the top left. There should be infinite resistance. Some multimeters will measure a value in the MΩ-range.

2. Now we are going to solder the power circuitry. Needed are the **USB connector** (J1), **zener diode** (D1), **multi-fuse** (F1), **220µF capacitor** (C4), the **supervisory circuit**

(U2), **one LED** (D2) and **one 2k2 resistor** (R3, color code red-red-red).



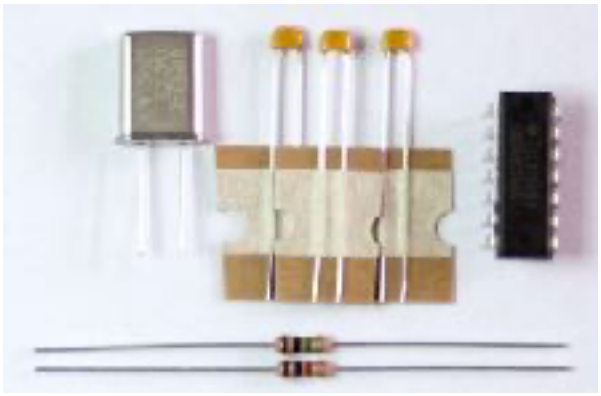
The USB connector has the smallest pins of all the components on the board, but there are only two to solder. Two large pins hold the connector firmly in place, the two small ones provide power. Three small pins are not soldered - they are for data connections we do not use. Make sure the zener diode, capacitor, supervisory circuit and LED are in the correct orientation. The zener diode has a stripe that is also printed on the board. The capacitor has a side that is marked with minus symbols, this should match the white side on the board. The supervisory circuit has its flat side to the right. Do not push it all the way down onto the board, let it sit 5-7mm (0.25") above the board. The LED has one round pad and one square pad. The long pin goes into the round pad marked '+', the short pin in the square pad.

You can now connect USB power to the gigatron. The LED should light up. If not, or if it is blinking, the most likely problem is that the power supply does not provide sufficient power. Please try another power supply. Also, check all the soldering.

It is good to know that the gigatron works on a voltage that is safe to humans, so nothing can go wrong if you accidentally touch (the components on) the board. Also, we have done our best to make the gigatron electrically safe: the zener diode and multi-fuse are included to prevent damage to the gigatron or the power supply in case of e.g. a short circuit or an excessive input voltage.

Always remove the USB power when commencing soldering again.

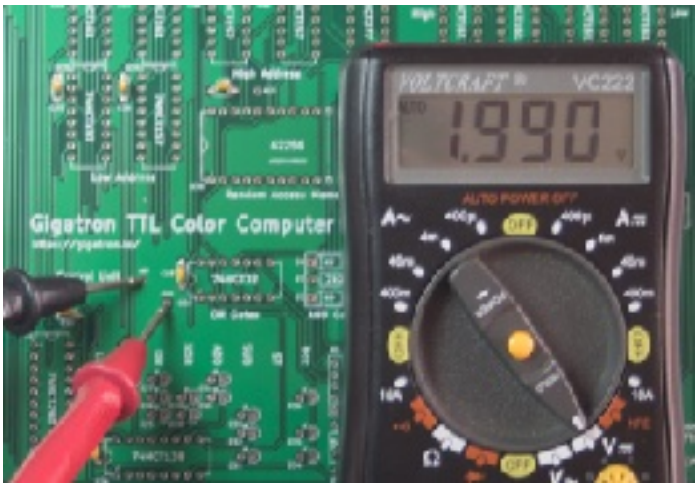
3. Next, we will be building the clock circuit that provides the timing signals for all the components (marked 'Clock Generator' on the board). Needed are the **crystal** (Y1), **three 47pF capacitors** (C1-C3), a **1MΩ resistor** (R1, color code brown-black-green), a **1kΩ resistor** (R2, color code brown-black-red) and a **74HCT04 chip**.



Apart from the chip, all components can be soldered in either orientation. See chapter 4 for more information about how to bend the pins of the IC to get it to fit into the board. If you solder it the wrong way around, cut the pins and remove them one by one, and request a new IC from us. In our experience, trying to get out the IC in one piece often results in damage to the board. This is true for all the IC's you are about to solder.

When done, the clock circuit should be working. The clock signal is a pulse. When using a volt meter, we can measure the average voltage of the pulse. When you are done soldering, turn on the board by plugging in the USB cable. If you look on the board below the large text "Gigatron TTL microcomputer", just to the right of the text "Control Unit", you will find two test pads. They are marked "GND" and "CLK1". Connect a Volt meter to these test pads. You should see a voltage of between 1.5 and 2.5 volts or -1.5 and -2.5 volts, if you swapped the two probes. This is because the voltage is alternating between 0 and 5 volts. If you do see a voltage between 1.5 and 2.5 volts, your clock circuit is working!

If this is not the case, check all the soldering and check if the IC is in the correct orientation and repeat the test with the multimeter until it works.



4. When you've gotten this far, you are ready to solder in all the ICs. First, solder the 28-pin socket on U36 (Random Access Memory) and the 40-pin socket on U7 (Program Memory), with the notch on the left. It could be that your kit has the RAM and/or EPROM already put into the socket. If this is the case, leave the ICs in their sockets and solder the combination into the PCB. Then, solder all the remaining ICs on the board, with the notch on the left or top as shown on the PCB. Mind the orientation, it is hard to desolder ICs without damaging the board. In the unfortunate case of a ending up with a wrongly soldered IC, it can be better to cut it loose and replace it with a new one. The layout of the ICs on the board is the same as how they are laid out on the conductive foam. For reference, here is a list of what IC goes where:

(U1)	(74HCT04)	(Clock generator)	U21	74HCT153	ALU Bit 0
(U2)	(MCP100)	(Supervisory circuit)	U22	74HCT153	ALU Bit 1
U3	74HCT161	Program Counter	U23	74HCT153	ALU Bit 2
U4	74HCT161	Program Counter	U24	74HCT153	ALU Bit 3
U5	74HCT161	Program Counter	U25	74HCT283	Adder
U6	74HCT161	Program Counter	U26	74HCT283	Adder
U7	40-pin socket	Program Memory	U27	74HCT377	Accumulator
U8	74HCT273	Instruction Register	U28	74HCT244	Bus Buffer (bottom left)
U9	74HCT273	Data Register	U29	74HCT161	X Register
U10	74HCT244	Bus Buffer (mid right)	U30	74HCT161	X Register
U11	74HCT139	Bus Access Decoder	U31	74HCT377	Y Register
U12	74HCT153	Condition Decoder	U32	74HCT157	High Address
U13	74HCT138	Addressing Mode Decoder	U33	74HCT157	High Address
U14	74HCT138	Instruction Decoder	U34	74HCT157	Low Address
U15	74HCT240	Inverters	U35	74HCT157	Low Address
U16	74HCT32	OR gates	U36	28-pin socket	Random Access Memory

U17	74HCT153	ALU Bit 4	U37	74HCT377	Output Register
U18	74HCT153	ALU Bit 5	U38	74HCT273	Extended Output
U19	74HCT153	ALU Bit 6	U39	74HC595	Input
U20	74HCT153	ALU Bit 7			

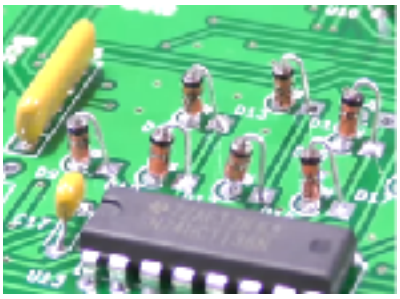
You have already soldered U1 and U2 on the board in the previous steps.

After every few ICs, do an inspection to see if adjacent pins are not soldered together and the ICs are in the right orientation. After you are done, you can power the board via USB. The "Power OK" LED should be on.

5. Next, solder on the **three resistor arrays** (R4, R5 and R8). Make sure pin 1 for each of these is in the correct spot. The PCB has a marking, the resistor array as well: a small

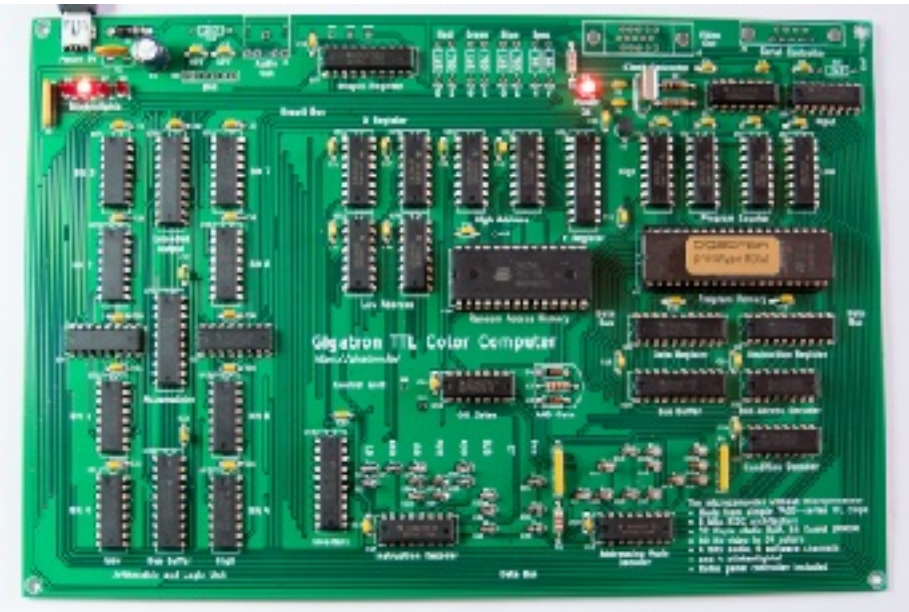
dot near the pin. Solder in the remaining **four LEDs** (D5~D8). The long pin goes into the hole marked '+'. If you power up the board, some or all of these LEDs will light up, as will the power LED.

6. Now we are going to build the gigatron instruction decoder using the diodes. You will need **30 diodes** (D3, D4 and D9 up to D36) and **two 2k2 resistors** (R6 and R7, color code red-red-red). The diodes D3 and D4 lay flat. The stripe on the diode should match the marking on the board. Diodes D9~D36 are standing up. So for these, only bend the pin on the side that has the black ring. That pin goes into the square padded hole, the other in the round



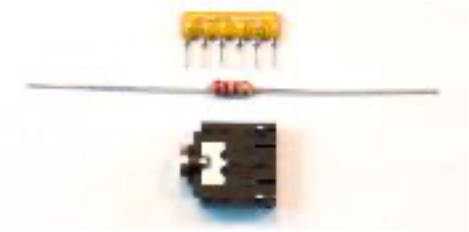
padded hole. In short: the diodes are on the left with the black ring on top, the wires are on the right.

7. Insert the **62256 RAM and 27C1024 EPROM ICs** (U36 and U7) into their sockets, if they are not already in there. Do this very carefully: if you push on the IC when the pin is not willing to slip into the hole, it will bend or break. But also, once you are absolutely sure all pins have slipped in and will move further down, press firmly so the ICs sit tight. Apply power. Now, the LEDs on the top left of the board should move in a test pattern, going back and forth. Congratulations! The gigatron is now running code!



If the "Power OK" LED is off or blinking, the most likely problem is that the power supply does not provide sufficient power. Please try another power supply.

If you still do not get the LED pattern, the problem most probably lies with the soldering. Check all the solder joints. Make sure there is a good bond between the pin and the PCB. Check that there are no shorts caused by solder overflowing to another pin. You can use a magnifying glass or the camera in your smartphone to carefully check all the solder joints. Resolder where necessary and try again.



8. To do something more useful than flashing LEDs, all that is left is connecting inputs and outputs. For the audio output, solder in the 6-pin **resistor array** (R9) with the dot matching the square hole, **one 2k2 resistor** (R10, color code red-red-red) and the **3.5" jack plug connector** (J2). Depending on the kind of connector that is supplied in the kit, you might need to solder three or five pins. The large pins are not connected but are soldered to firmly connect the connector to the PCB. There is no difference in functionality between the two variants. The audio connector might not lay completely flat on the PCB. That is no problem.

9. For the game controller, only **one 2k2 resistor** (R19, color code red-red-red) and the **9-pin connector** (J4) are needed. For more sturdiness, solder the clips as well.

10. We conclude with the VGA output. Use **two 68Ω resistors** for sync (R17 and R18, color code blue-grey-black). Use **three 750Ω resistors** for each of the high bits of the six bit RGB output (R12, R14 and R16, color code purple-green-brown) and **three 1k5 resistors** for the low bits (R11, R13 and R15, color code brown-green-red). Solder in the **15-pin VGA connector** (J3) including the clips.

11. Connect a VGA-compatible monitor to the VGA connector, apply power and your gigatron should display the start screen! You could also use an HDMI monitor, if you have an (active) VGA-to-HDMI adapter. With some VGA-to-HDMI adapters, the edges of the screen will not be shown.

The gigatron comes with pre-programmed software contained in an EPROM memory module. Therefore, to get a working microcomputer, only the hardware assembly is necessary. There is no need to program it or load external software. Every time you turn on the gigatron, it will run the program that we pre-programmed into the EPROM.

If you do not get the start screen, the problem most probably lies with the soldering. Check all the solder joints. Make sure there is a good bond between the pin and the PCB. Check that there are no shorts because the solder is overflowing to another pin. You can use a loupe or the camera in your smartphone to carefully check all the solder joints. Resolder where necessary and try again.

In very rare cases, the display may be incompatible with the signal generated by the gigatron. Please try another computer monitor. We have tested many monitor types and we have not yet encountered any compatibility issues ourselves. It is good to know that as long as the LED lights are running, there will also be a video signal, because both are created by the same part of the software.

If you cannot get the system to work, check https://gigatronttl.eu/diagnostics for diagnostics or consult the forum at https://forum.gigatronttl.eu/ If that does not help, contact support@gigatronttl.eu. Tell us:

- What the system is doing or is not doing;

- What you have done to fix it;
- Send us close-ups of the soldering on the board, and an overview of the whole board, from both sides.

12. To complete the build, all that is left is putting the gigatron in its casing. That is as easy as opening the case, dropping the PCB in and closing the lid again. No screws are used. We have supplied 8 little rubber feet that are self-adhesive. You can put four on the bottom of the casing (one in each corner) to avoid scratching. The remaining four can be used to put under the PCB in the four corners. This way, the soldering will nor rest on the case. This is optional, there is no problem if you decide not to use the rubber feet on the PCB. Just see how you think the PCB fits best.

13. From the backside of the case, you can connect:

- USB power via the mini-USB plug;
- a computer screen via the VGA plug (with 15 holes), optionally via a VGA to HDMI converter (depending on the converter, you may lose a bit of the screen);
- the supplied game controller via the serial plug (with 9 pins) (or the PS/2 adapter, see next page);
- speakers or an amplifier via the audio jack.

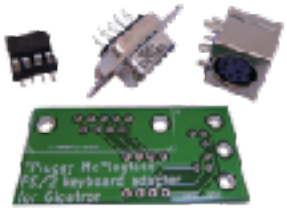
Note that the game controller is a so-called Famicom controller. Other controllers will not work, even if they use the same connector. The controller can be marked 'I' or 'II', this makes no difference.

The audio output is intended to be connected to PC speakers or a Hi-Fi stereo amplifier. You can also connect headphones, but the volume might be a bit low and there is no volume control on the computer.

Pluggy McPlugface PS/2 Adapter

Next, we will build the PS/2 adapter. You can use the gigatron without it (using the controller instead) to play the built-in games, but to write BASIC programs, you will need this adapter and a PS/2 keyboard (not included).

The PS/2 adapter is a small PCB with a connector that plugs into the game controller port. It has another connector to connect your PS/2 keyboard (not included). The 'brains' is an ATtiny85. This tiny 8-bit RISC-based microcontroller can translate the signals coming from the PS/2 keyboard to signals the gigatron can understand. The microcontroller contains 8kB of flash memory, that we have pre-programmed with the translation software. The adapter also contains an image of Tiny BASIC v2, exactly the same BASIC that is already pre-programmed in ROM v3. Tiny BASIC can be loaded into the gigatron using the Loader application as well, so it can be used on older ROM versions. The space that is left can store your own BASIC program, but saving programs will only work with ROMv3 or newer.



14. First, solder on the **IC socket**, either with or without the **ATtiny85** microprocessor still inserted. It should be inserted **<u>on the side that says "ATtiny85 8 MHz"</u>**. Make sure the half moon cutout in the IC socket matches the half moon drawn on the circuit board. The ATtiny85 is more delicate than the components used in the gigatron. If you insert it the wrong way around and power it on, you will destroy it. Also, if your soldering skills are not great, you might want to solder in the empty socket and let it cool before inserting the ATtiny85. Be careful: if you hold the socket with your fingers while soldering, you might burn them!

15. Next, solder on the **PS/2 connector**. It goes on the same side of the board, next to the ATtiny85. The connector has 6 small pins and 3 large ones, that keep the plug in place. We have had cases where we needed to carefully jiggle the plug to get the pins into the holes.

16. The last part to solder is the **DB9 plug**, which goes on **<u>the other side</u>** of the PCB, that says "Pluggy McPlugface PS/2 keyboard adapter for Gigatron". The 9 small pins need to be soldered. The plug might or might not have big pins that go in the big holes. If that is the case, you will probably need some force to insert the plug. These pins can be soldered, but you will require a soldering iron that generates enough heat to be able to solder them. Not soldering them is fine too, the construction is already sturdy enough without it.

17. Congratulations, you're all done!

The Pluggy McPlugface adapter does not come with an enclosure. You can just connect it to the gigatron without one.

PS/2 keyboards are actually quite complicated devices. We have tested as many keyboards as we could, but the possibility exists you come across a keyboard that does not work. Problems might arise if the keyboard draws too much power. Try another USB cable or power supply first. Also, some keyboards take a very long time to initialise, up to one minute.

We have heard success stories of people using a USB keyboard with a converter from USB to PS/2. Wether this works, depends on both the converter and adapter. At the forum (https://forum.gigatronttl.eu/) you can find more information about which devices do or don’t work.

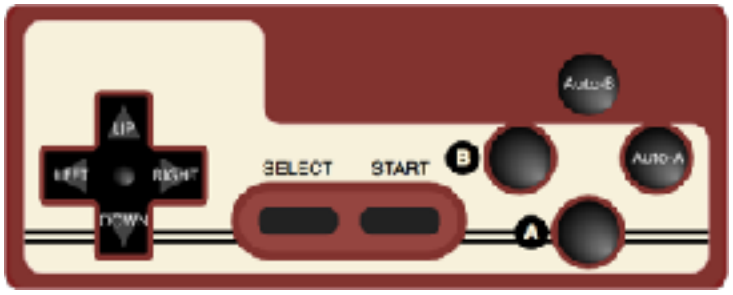
6

User manual

Using the built-in software

Controller and keyboard

You can either connect the game controller or the PS/2 adapter, they cannot be connected both at the same time. The standard gigatron controller is a Famicom-like game controller. The left side is for selecting a direction. [A] and [B] are on the right hand side as shown in the picture. Both have an "Auto repeat" button next to them.



The [Start] button is the soft reset button. Holding it down for 2 seconds stops the current application and returns to the menu. The [Select] button cycles through the number of retro scan lines (0 to 3, default is 1). Applications run faster with more retro scan lines enabled, and some work only well in mode 1 or up.

We have programmed the PS/2 adapter in such a way that the keyboard can be used to provide the game controller outputs, so with just a PS/2 keyboard attached, you have access to all of the functionality. In other words: the PS/2 keyboard replaces the controller and gives you more functionality.

Push the DB9 plug of the PS/2 adapter into the game controller port and connect a PS/2 keyboard to the PS/2 connector. The PS/2 adapter fits nicely on the back of the gigatron.

Once booted, you will find that these keys on the keyboard emulate the controller buttons:

Controller	PS/2 keyboard equivalent
Left, Right, Up, Down	Left, Right, Up, Down arrow keys
SELECT	Page Down
START	Page Up
A	End, Delete or Backspace (so not they A key)
B	Home or Insert (so not the B key)

Controller	PS/2 keyboard equivalent
2 second press on SELECT	Ctrl-Alt-Del (no need to keep holding these down)

If your keyboard is not working, possible problems are the keyboard or the power. Try another power supply or USB cable. If that doesn't work, try another keyboard if you can. Also, it might take up to a minute for the keyboard to initialise.

Since the adapter has to work with different kinds of keyboard layouts, the possibility exists to switch between keyboard mappings. Mappings for US (US), British (GB), German (DE), French (FR), Italian (IT) and Spanish (ES) keyboard is built in. Press Control-Alt-F1 for the US mapping, Control-Alt-F2 for British and so on. Your selection will be stored in the microcontroller so this has to be done only once. Only US-ASCII codes are supported so no accented characters or shapes for instance.

Tiny BASIC can be loaded from the menu. If you have an old ROM version that does not contain Tiny BASIC, you can load it from the adapter into memory. Move the arrow down to `Loader` and activate it by pressing End, Delete or Backspace. You wil see the message `Ready to load`. You can press Control-F1 to display help. To load Tiny BASIC, press Control-F2. This will start a data transfer from the ATtiny85 to the gigatron and you will see pixels changing color during the load process. When done, Tiny BASIC will automatically start.

Menu

After power up the menu appears and the four "blinkenlight" LEDs start running in a pattern. Use the [Up], [Down], [Left] and [Right] buttons to navigate the menu, and [A] on the game controller or Del on the keyboard to start the desired application.

The top line displays the amount of detected memory. The kit comes with 32K. If there is no RAM inserted, the menu will not appear and the four LEDs will all be off.

Snake

The snake runs across the screen and can't stop. Use the arrows for changing its direction. The green blocks are food and must be eaten. Each time the snake grows a bit longer. Bright blocks are poisonous and must be avoided. Don't hit the walls, and don't bite your own tail.

The countdown timer is used for score keeping. If you eat food before the counter reaches zero, the remaining seconds will be added to the game score, and five seconds added to the timer (with a maximum of 15). If the counter reaches zero, the next food doesn't give any points, but it will still increase the timer.

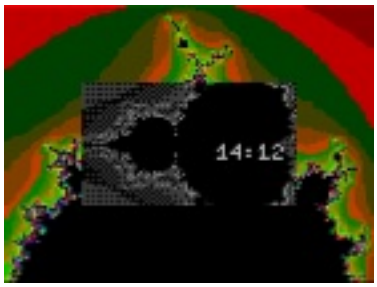
The game starts automatically. If the player doesn't change direction before reaching the first wall, the gigatron plays a game by itself.

Racer

You control Walter's car. Steer with the [Left] and [Right] buttons. Accelerate with [A] and break with [B]. The road is very slippery, so be careful to stay on track. The objective is to complete laps as fast as possible without hitting the curbs.

Racer works best with the game controller.

Mandelbrot



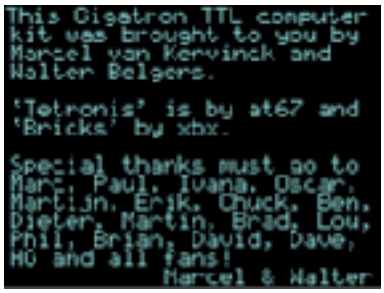
This application draws many views of the Mandelbrot fractal, an intriguing and endlessly complex figure that emerges from a simple mathematical formula: $z^2 + c$.

The clock in the center counts hours and minutes. It is a 24-hour clock. You can set it by pressing [A] and holding it down. The color changes to yellow and then you can use the arrows to set the time. Use [Left] and [Right] to change hours, and [Up] and [Down] to change minutes. Shortly after releasing the [A] button, the color changes back to white. The colon stops flashing once the clock is set.

Pictures

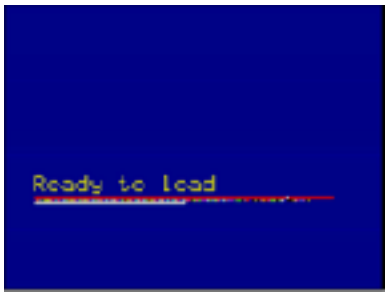
This converts the gigatron into a picture frame. ROM v4 and later come with two classic colorful high-resolution images built in. Every 15 seconds the next image is shown. Hold down a button if you want to look at the current image for a bit longer. ROM v5 has pictures of the creators of the gigatron.

Credits



Many thanks to those who have helped, directly or indirectly, in making the gigatron to what it is today. Your enthusiasm, support, ideas and other contributions are invaluable!

Loader



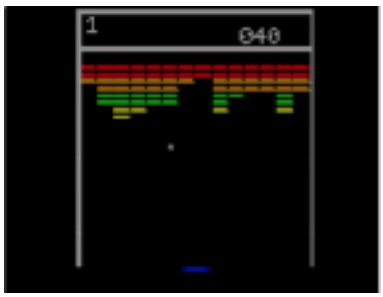
The loader listens to data packets on the controller port and stores them in memory. You can hook up an external device to the controller port and send commands or complete new programs into the gigatron this way and run them. The PS/2 adapter can send BASIC using the loader. When in the loader, press Control-F1 to activate this. Loading BASIC this way is only needed for older ROMs, that do not contain BASIC. In your system, you can just run BASIC from the menu.

Tetronis

Blocks are falling down. You can move them left or right within the box, using the left and right arrows. Using the up arrow, you can rotate the falling block. The down key will increase the speed the block is falling with. Your objective is to stack the blocks tightly. When the blocks create a full horizontal line, it will disappear and earn you points, shown at the bottom. When there is no space left for the block to fall, the game ends. In the top right corner, you can already see the shape of the next block that will fall.

Thanks to at67 for this contribution!

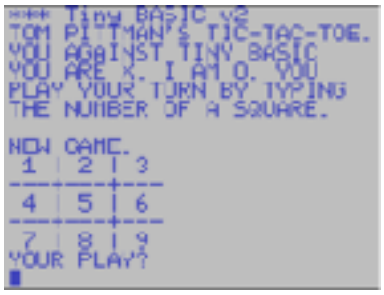
Bricks



The objective is to clear all bricks and prevent the ball from falling out of the bottom of the screen. You have a bat that you can move left and right, to bounce the ball back up. On the top, there are colored blocks that disappear when hit by the ball. Each block taken away will earn you a point. The score can be seen in the top right. On the top left, the amount of lives left is shown.

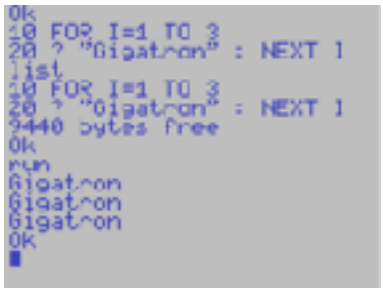
Thanks to xbx for this contribution!

Tic-Tac-Toe



This starts BASIC with a pre-installed program, Tic-Tac-Toe. This program was listed in the January 1977 newsletter from the famous Homebrew Computer Club. It was written by Tom Pittman and it runs as-is on the gigatron. You will need to attach a keyboard to be able to play.

BASIC



Tiny BASIC is based on the original Tiny BASIC, which is a small BASIC interpreter developed by Dennis Allison in 1975, but has more commands. You can use it to write your own programs.

When Tiny BASIC starts, it prints out the amount of free bytes. It will then present the "`Ok`" prompt and wait for your input.

This manual does not provide a full course in programming in BASIC, but we do want to give you a head start. We will describe the commands and give some sample programs that you can enter and adapt. **At https://gigatronttl.eu/basic you will find a more in-depth tutorial**. Some sample BASIC programs can be found at https://github.com/kervinck/gigatron-rom/tree/master/BASIC.

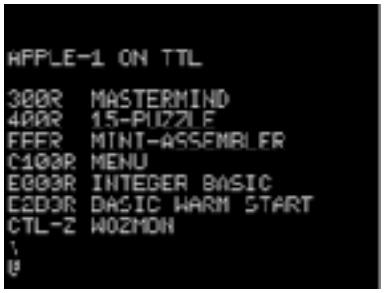
The following commands are available in Tiny BASIC:

Command	Example	Description
PRINT or ?	PRINT “Hello”	Print something on the screen
AT	AT 5*A,B	Position the cursor on the screen (X,Y), the Y argument is optional
PUT	PUT 64	Write an ASCII character on the sceen
IF THEN	IF A>1 THEN GOSUB 100	Conditional execution
GOTO	GOTO 200	Continue execution at given line
INPUT	INPUT A	Read input and store it in a variable
LET	LET B=7*A+1	Calculate an integer value and assign it to a variable, the ‘LET’ keyword is optional
LINE	LINE 25,68	Draw a line from the current position (W,H), the W is the width and H the height (both can be negative)
FOR	FOR I=1 TO 100	Start a loop, in which a variable goes through a range
NEXT	NEXT I	End of the loop for the given variable
POKE	POKE 42,8	Store a byte in memory at the given address
GOSUB	GOSUB 300	Execute a subroutine at the given line
RETURN	RETURN	End the subroutine and continue execution
REM or ‘	REM Some comment	Comment
:	A=1 : B=2	Separate multiple statements on a line, the line length cannot exceed 25 characters
MODE	MODE 3	Set retro line mode
CLS	CLS	Clear screen
NEW	NEW	Clear BASIC memory
LIST	LIST	List BASIC program in memory
RUN	RUN	Run BASIC program in memory
END	END	END program and return to prompt
PEEK	A=PEEK(47)	Read a byte form memory from the given address
RND	N=RND(100)	Give a random value between 0 and the given number
USR	PRINT USR(496)	Calls a vCPU function directly from RAM
SAVE	SAVE	Save program in the PS/2 adapter (ROMv3 or newer needed)

MS BASIC (DEVROM only)

Microsoft BASIC for 6502. Micro-Soft's first BASIC ran on the Altair computer, which has an Intel 8080 CPU. For machines with 8kB or more of RAM, their version of BASIC had more commands. This version was ported to the 6502 architecture. Commodore used this for their PET computers. The Commodore version has an easter egg that triggers when the command `WAIT 6502,1` is given.

WozMon (ROMv4) / Apple-1 (DEVROM)



WozMon (the Woz Monitor) is a recreation of the WozMon ROM monitor program that was present in the Apple-1 computer, written by Steve Wozniak. It runs in 6502 emulation mode.

The program starts with a message and has no (ROMv4) or a blinking at-sign (DEVROM) as prompt. You can read or write memory and jump to a memory location.

To read memory, enter one or more hexadecimal addresses and press enter. The input is case insensitive. You can also specify ranges using the notation `address.address`. Examples are "`2a`", "`0000.00FF`" and "`1 2a 40.4f`". WozMon remembers the last used location and this can then be omitted in a subsequent range definition. For example, if you would give the command `10` to read one byte, and after that `.20`, that second command would show the memory in locations 11 to 20.

Writing data can be done by adding `:value` to an address. The command `2a:22` will display the current value of memory location `2A` and will then write `22` in it. This will turn the background purple. The command `2f:04` will make the blinkenlights run faster. You can write multiple values in consecutive memory locations by using the form `address:value1 value2 value3 ..`

You can start execution at a certain memory location by using the command `address R`. Several Apple-1 programs have been preloaded into memory in the DEVROM, listed in the start-up message. For instance, you can run the Mastermind program starting at memory location `300`, by giving the command `300R`. Pressing `Control-Z` gets you back to WozMon.

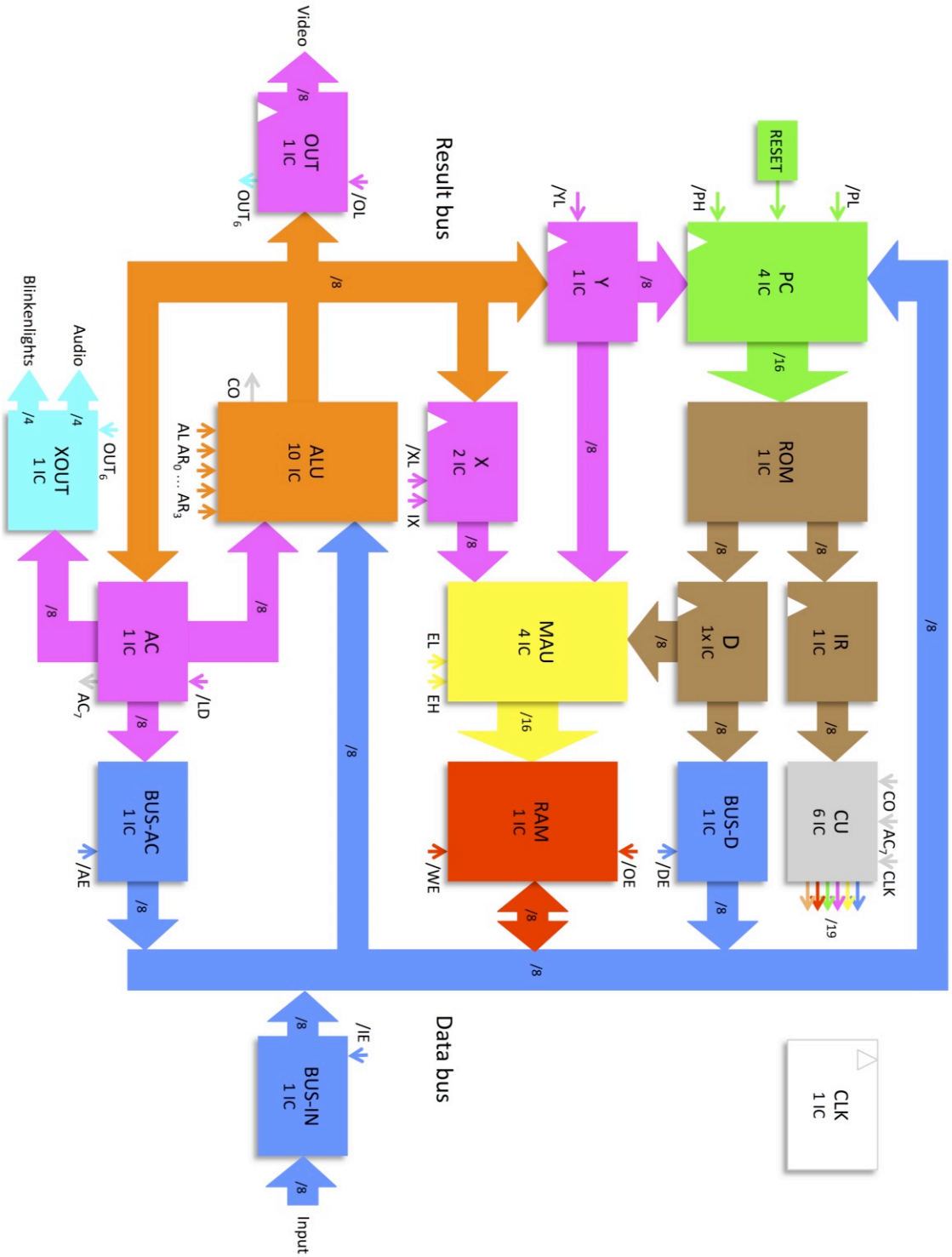
Further reading

We are still making the gigatron more accessible for hacking by publishing on our website, in GitHub and on the Hackaday project pages. The best place to keep in touch with new developments in this area is by joining the gigatron Hackers forum at https://forum.gigatronttl.eu/.

Appendix A

Schematics

To learn more about the inner working of the hardware, we include the schematics of the gigatron. More information is available at https://gigatronttl.eu/.



These schematics use 74HCT but most components can be replaced directly with either 74LS or 74HC. Two notes to keep in mind: (1) 74LS uses more power and has lower V_OH. E.g. when replacing the OUT register with 74LS, adjust the RGB resistors accordingly. (2) For the clock 74HCT always gives the most desirable duty cycle and reliability.

Marcel van Kervinck and Walter Belgers

Sheet: /
File: Gigatron.sch

Title: Gigatron TTL microcomputer

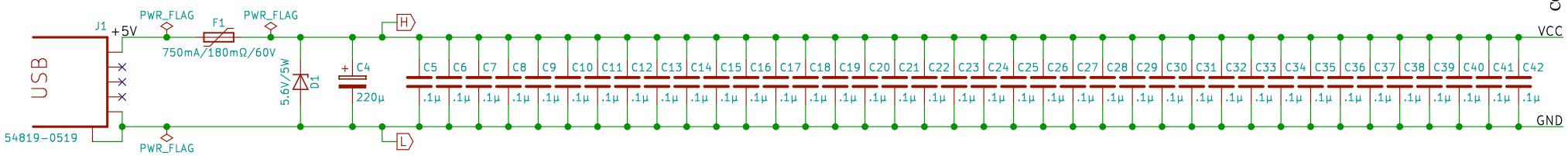
Size: A4 | Date: 2018-05-10 | **Rev: Release**

KiCad E.D.A. kicad 4.0.6 | Id: 1/8

Power

Multifuse F1 helps protect the upstream power supply against faults/shorts in the computer (e.g. those caused by build errors or abuse)

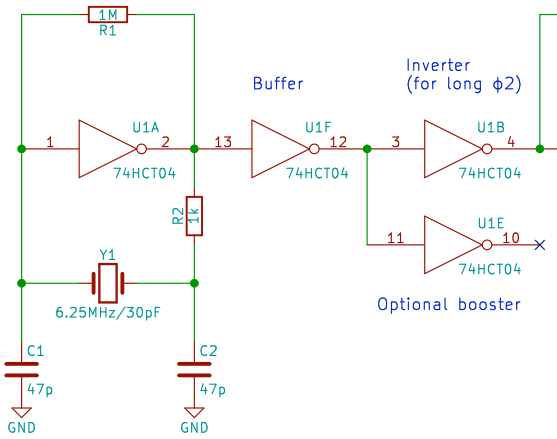
A decoupling (or bypass) capacitor in close proximity to every IC's power pin provides transient current when switching between logic levels.



Zener diode D1 protects the circuit against reverse current and overvoltage

Clock

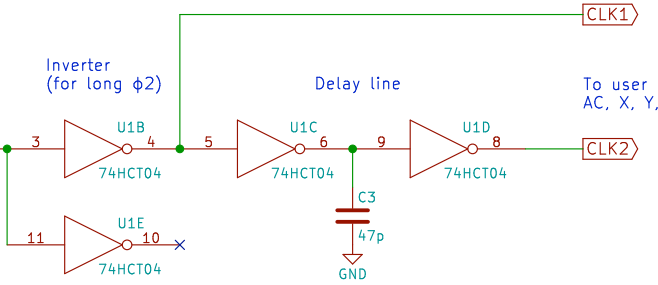
Pierce oscillator



Buffer

Inverter (for long ϕ2)

Delay line



Optional booster

To instruction fetch and control unit
PC, IR, D, WE

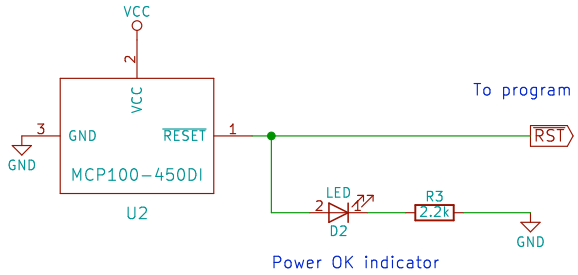
CLK1

To user registers
AC, X, Y, OUT

CLK2

The 47pF of C3 provides an additional CLK2–CLK1 shift of 5ns, which is required when using the original 70ns SRAM at 6.25MHz, or when overclocking in general. (It is not needed when using a 55ns SRAM module at 6.25MHz.)

Reset



To program counter PC

Power OK indicator

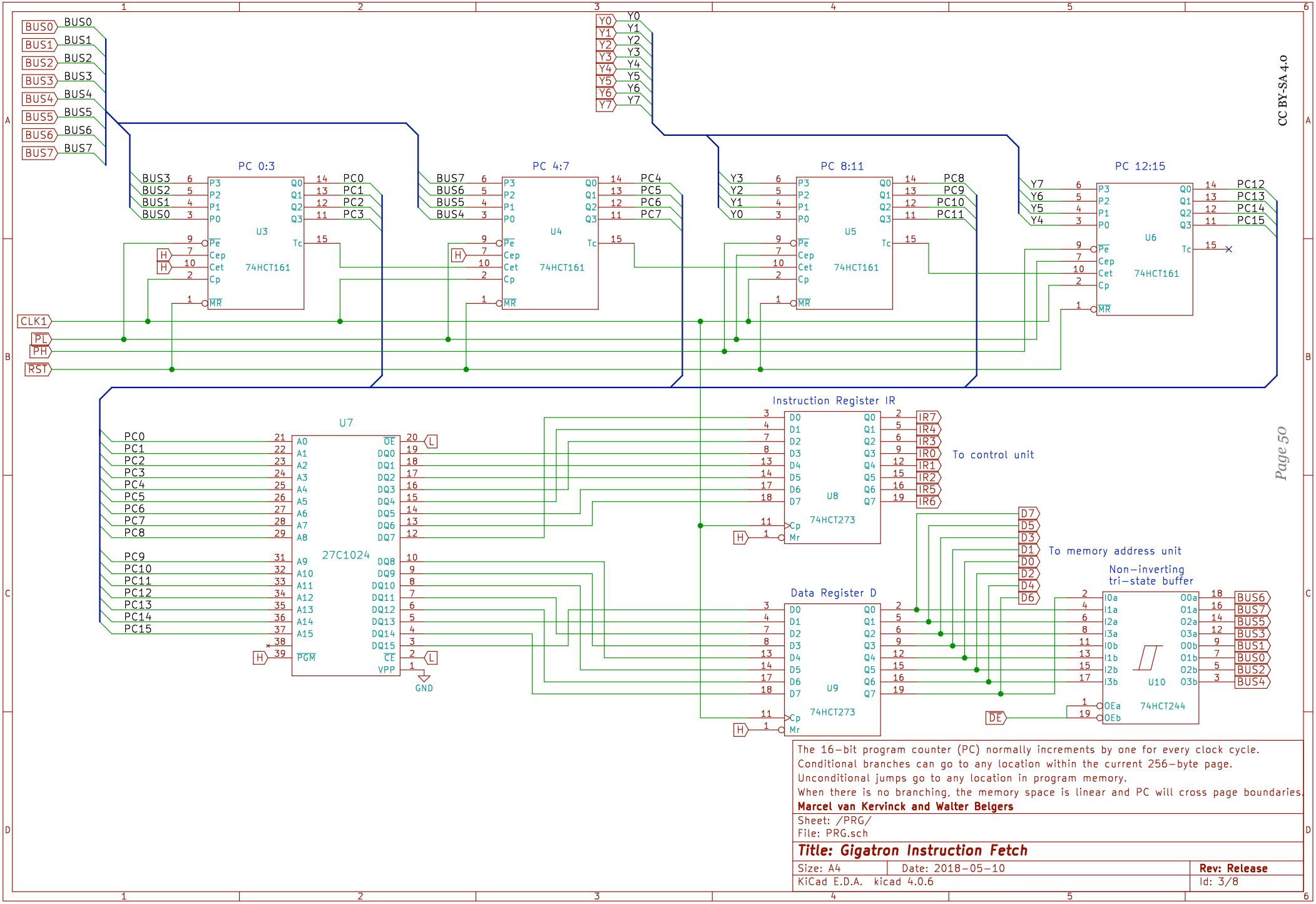
(1) Power comes from the +5V that a USB charger provides. The data lines are not used. (2) The oscillator generates 6.25MHz square waves. The HCT-inverter gives a long 2nd clock phase. This is useful for the /WE pulse derived from it. (3) The Power-On Reset (POR) supervisory circuit holds /RESET low for 350ms during power up and brown outs.

Marcel van Kervinck and Walter Belgers

Sheet: /PCR/
File: PCR.sch

Title: Gigatron Power, Clock and Reset

Size: A4	Date: 2018-05-10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 2/8



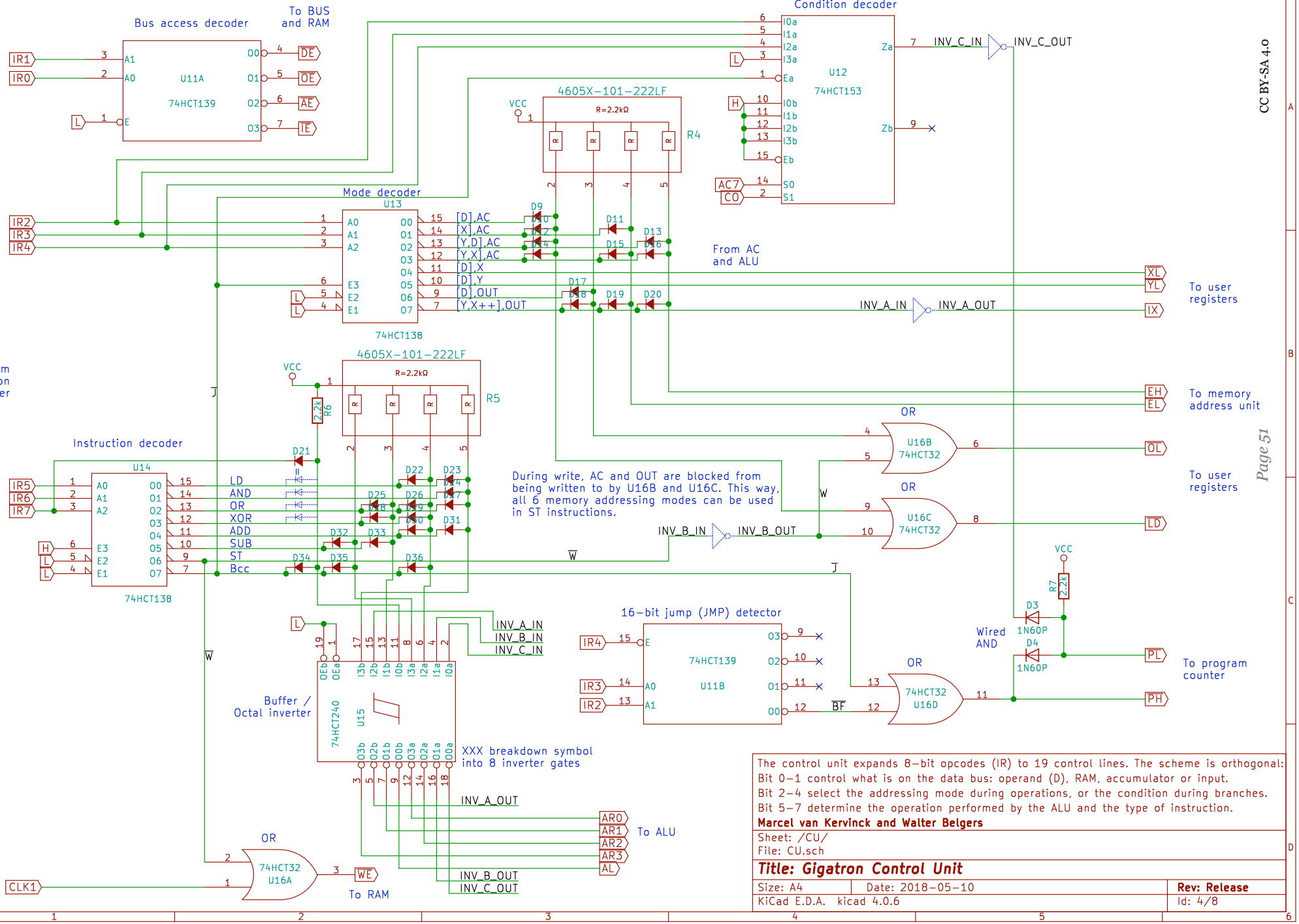
The 16−bit program counter (PC) normally increments by one for every clock cycle.
Conditional branches can go to any location within the current 256−byte page.
Unconditional jumps go to any location in program memory.
When there is no branching, the memory space is linear and PC will cross page boundaries.

Marcel van Kervinck and Walter Belgers

Sheet: /PRG/
File: PRG.sch

Title: Gigatron Instruction Fetch

Size: A4	Date: 2018−05−10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 3/8


Bus access decoder
To BUS and RAM
U11A
74HCT139
IR1
IR0
L
DE
OE
AE
IE
Condition decoder
U12
74HCT153
INV_C_IN
INV_C_OUT
AC7
CO
H
4605X-101-222LF
R=2.2kΩ
R4
Mode decoder
U13
74HCT138
IR2
IR3
IR4
[D],AC
[X],AC
[Y,D],AC
[Y,X],AC
[D],X
[D],Y
[D],OUT
[Y,X++],OUT
From AC and ALU
INV_A_IN
INV_A_OUT
XL
YL
IX
To user registers
EH
EL
To memory address unit
OL
LD
PL
PH
From instruction register
J
W
R5
R6
2.2k
Instruction decoder
U14
IR5
IR6
IR7
LD
AND
OR
XOR
ADD
SUB
ST
Bcc
During write, AC and OUT are blocked from being written to by U16B and U16C. This way, all 6 memory addressing modes can be used in ST instructions.
U16B
U16C
74HCT32
INV_B_IN
INV_B_OUT
16-bit jump (JMP) detector
U11B
IR4
IR3
IR2
BF
U16D
Wired AND
R7
D3
D4
1N60P
Buffer / Octal inverter
74HCT240
U15
XXX breakdown symbol into 8 inverter gates
INV_A_IN
INV_B_IN
INV_C_IN
INV_A_OUT
INV_B_OUT
INV_C_OUT
AR0
AR1
AR2
AR3
AL
To ALU
CLK1
U16A
WE
To RAM
CC BY-SA 4.0
Page 51
The control unit expands 8-bit opcodes (IR) to 19 control lines. The scheme is orthogonal:
Bit 0-1 control what is on the data bus: operand (D), RAM, accumulator or input.
Bit 2-4 select the addressing mode during operations, or the condition during branches.
Bit 5-7 determine the operation performed by the ALU and the type of instruction.
Marcel van Kervinck and Walter Belgers
Sheet: /CU/
File: CU.sch
Title: Gigatron Control Unit
Size: A4
Date: 2018-05-10
Rev: Release
KiCad E.D.A. kicad 4.0.6
Id: 4/8

Memory Address Unit

Quad 2:1 MUX

U35 74HCT157

X4 D4 X5 D5 X7 D7 X6 D6 L

I0a I1a I0b I1b I0c I1c I0d I1d S E

Za Zb Zc Zd

A4 A5 A7 A6

Quad 2:1 MUX

U34 74HCT157

X1 D1 X2 D2 X0 D0 X3 D3 EL L

A1 A2 A0 A3

Quad 2:1 MUX

U33 74HCT157

Y5 H Y6 H Y1 H Y3 H From control unit EH L

A13 A14 A9 A11

Quad 2:1 MUX

U32 74HCT157

Y0 H Y7 H Y4 H Y2 H L

A15 A15

A8 A12 A10

32kB/70ns static RAM

U36 HM62256BLP-7

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14

D0 D1 D2 D3 D4 D5 D6 D7 OE WE CS

BUS0 BUS1 BUS2 BUS3 BUS4 BUS5 BUS6 BUS7 OE WE L

The 6.25 MHz system has a clock period of 160 ns.
In each clock cycle 1 RAM access and 1 ALU operation can be done sequentually. Therefore the computer needs an SRAM speed of 70ns for reliable operation. In the kit 55ns SRAM is included for additional headroom.

EH	EL	AH	AL	Notation
L	L	YYYYYYYY	XXXXXXXX	[y,x]
L	H	YYYYYYYY	DDDDDDDD	[y,$dd]
H	L	00000000	XXXXXXXX	[x]
H	H	00000000	DDDDDDDD	[$dd]

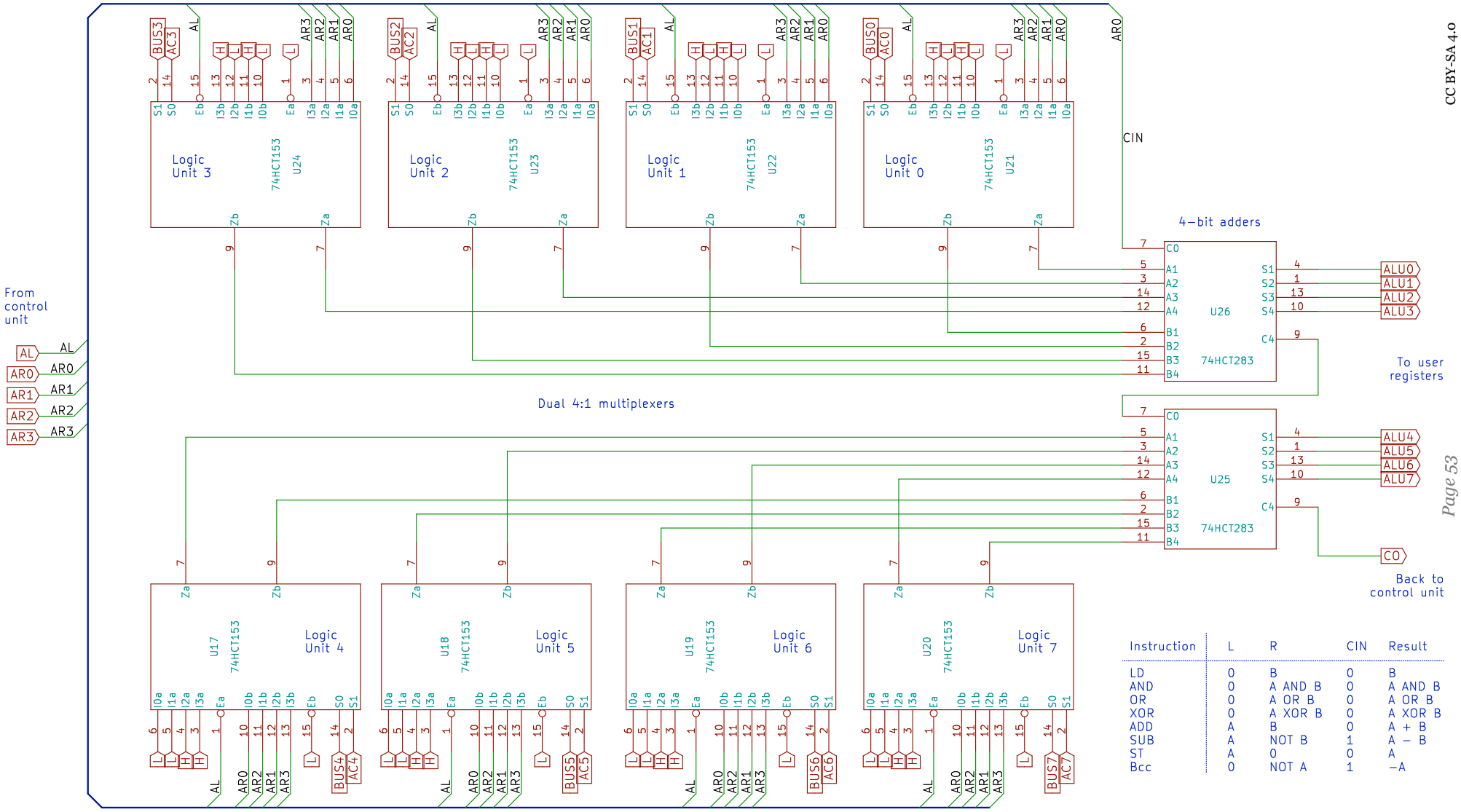
For speed and simplicity the memory addressing space is two-dimensional.
The address unit can create addresses in four different ways, as shown in the table.
The low address byte originates either from the operand (D) or from the user X register
The high address byte is either zero or comes from the user Y register.

Marcel van Kervinck and Walter Belgers

Sheet: /RAM/
File: RAM.sch

Title: Gigatron Memory and Address Unit

Size: A4	Date: 2018-05-10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 5/8



Instruction	L	R	CIN	Result
LD	0	B	0	B
AND	0	A AND B	0	A AND B
OR	0	A OR B	0	A OR B
XOR	0	A XOR B	0	A XOR B
ADD	A	B	0	A + B
SUB	A	NOT B	1	A − B
ST	A	0	0	A
Bcc	0	NOT A	1	−A

Each of the eight logic unit packages contains two 4:1 multiplexers we call L and R. R is fully programmable by the truth table hosted in the control unit (AR0:3). It performs logic functions on both accumulator (A) and bus data (B). L will either be zero or the value of the accumulator. Therefore it needs just one control line (AL).
The stage with two 4-bit adders combines L and R for a final result. The table lists resulting operations that we use.

During jump instructions, `-A' is computed so that the carry out (CO) indicates if the accumulator is zero.
During store instructions, `A' is `computed' so that, depending on addressing mode, the accumulator can be copied to X or Y as well.

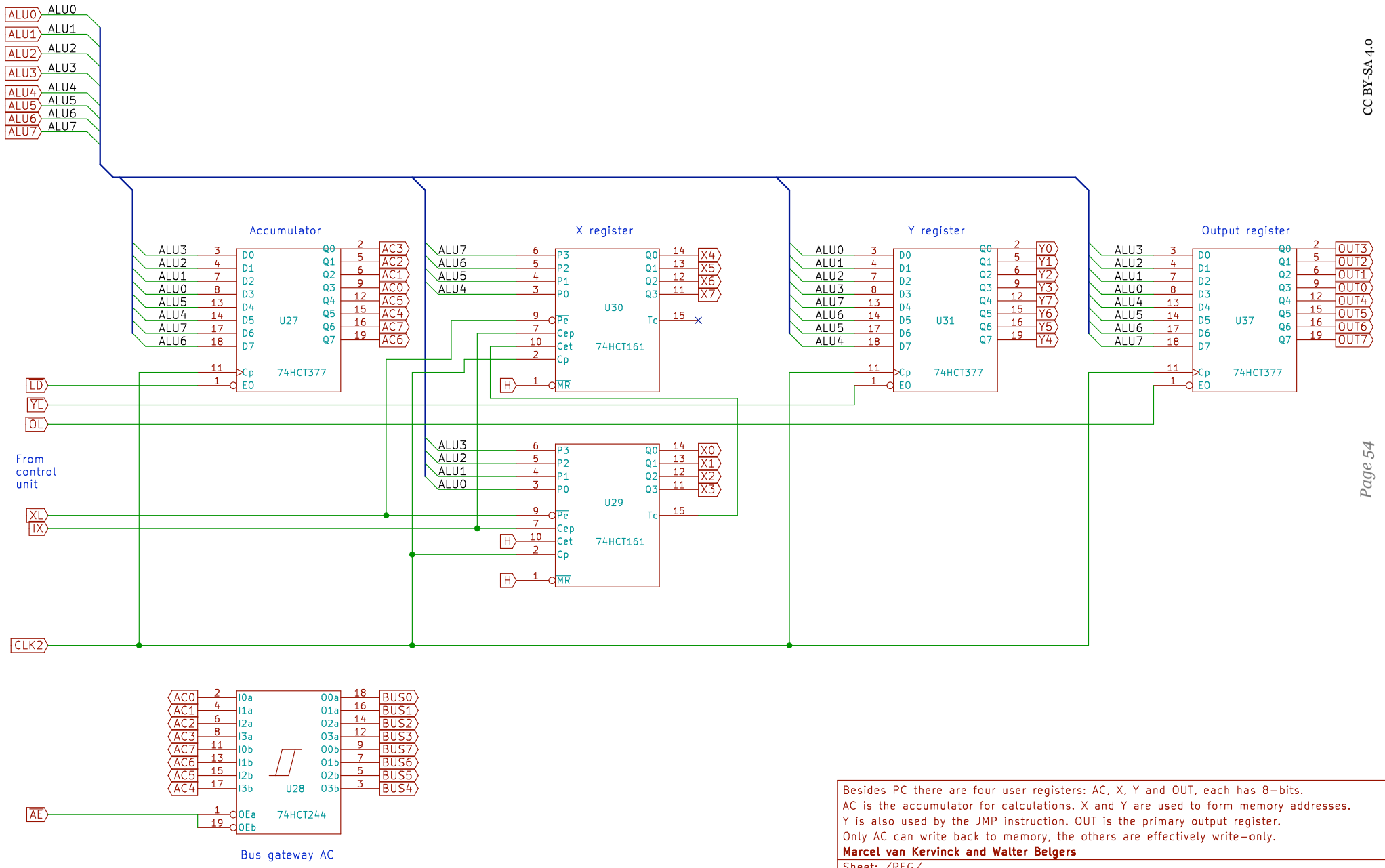
This unit is the equivalent of two large 4-bit 74181 ALU chips that were at the heart of many 1970s minicomputers. This design is inspired by Dieter Mueller's excellent notes starting from http://6502.org/users/dieter/a1/a1_4.htm

Marcel van Kervinck and Walter Belgers

Sheet: /ALU/
File: ALU.sch

Title: Gigatron Arithmetic and Logic Unit

Size: A4	Date: 2018-05-10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 6/8



Besides PC there are four user registers: AC, X, Y and OUT, each has 8-bits.
AC is the accumulator for calculations. X and Y are used to form memory addresses.
Y is also used by the JMP instruction. OUT is the primary output register.
Only AC can write back to memory, the others are effectively write-only.

Marcel van Kervinck and Walter Belgers

Sheet: /REG/
File: REG.sch

Title: Gigatron User Registers

Size: A4	Date: 2018-05-10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 7/8

The RGB resistor values depend on the 74xx sub-family of the OUT register:
7400/74LS: 390Ω / 820Ω
74HCT/HC: 750Ω / 1500Ω

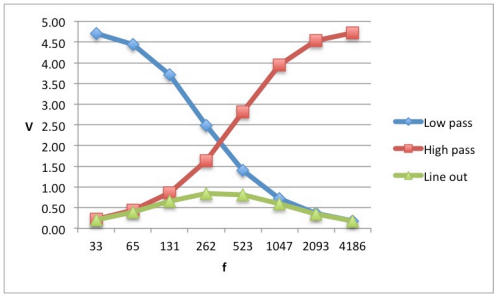
J3
DB15_FEMALE_HighDensity_MountingHoles

OUT0 — 1k5 R11 — RED
OUT1 — 750 R12 — RED
OUT2 — 1k5 R13 — GREEN
OUT3 — 750 R14 — GREEN
OUT4 — 1k5 R15 — BLUE
OUT5 — 750 R16 — BLUE
OUT6 — 68 R17 — HSYNC
OUT7 — 68 R18 — VSYNC

GND

DB15 connector for 64 colors analog
RGB video out (VGA compatible)

Attenuation from 74HCT output to audio
line level by two overlapping filter slopes:
f_L (700Hz) > f_H (160Hz)



Blinkenlights

R8
R=2.2kΩ

LED D8
LED D7
LED D6
LED D5

Extended output

U38
74HCT273

AC3 AC2 AC1 AC0 AC4 AC5 AC6 AC7
D0 D1 D2 D3 D4 D5 D6 D7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
Cp
Mr
H

DAC
R=10kΩ
R9

Low pass
C43 0.1µ

AC coupling
High pass
C44 0.1µ
2.2k R10

Line out
J2
TIP TSH RSH RING SLEEVE
CONN-JACK-STEREO-R/A-3PIN-3.5MM

Audio
3.5mm stereo audio connector
for PC speakers or amplifier

Page 55

Game controller input

VCC
2.2k R19
U39
74HC595

SER_PULSE
SER_LATCH
SER_DATA
SER
SRCLK
SRCLR
RCLK
G
IE
QA QB QC QD QE QF QG QH QH
BUS0 BUS1 BUS2 BUS3 BUS4 BUS5 BUS6 BUS7

J4
DB9_MALE_MountingHoles

DB9 connector for a
"Famiclone" game controller

OUT gives 6 color bits and 2 sync bits for VGA. XOUT is the extended output register.
XOUT is controlled indirectly through /HSYNC and AC. On a positive edge it clocks in AC.
XOUT drives 4 blinkenlights and simple 4-bits audio. The waveform comes from software.
/HSYNC and /VSYNC double as clocks for the game controller and receiving shift register.

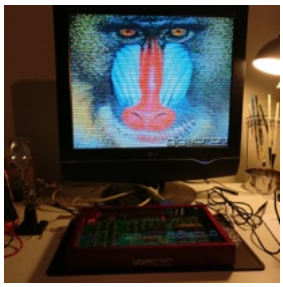
Marcel van Kervinck and Walter Belgers

Sheet: /PER/
File: PER.sch

Title: Gigatron Peripherals

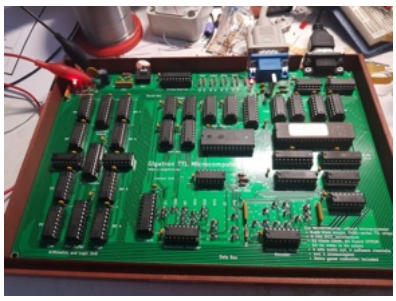
Size: A4	Date: 2018-05-10	**Rev: Release**
KiCad E.D.A. kicad 4.0.6		Id: 8/8

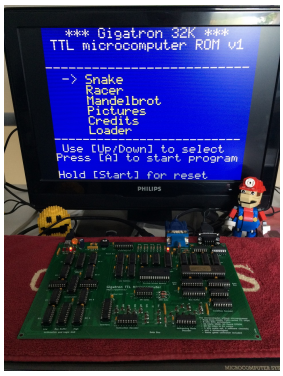

*** Gigatron 32K ***
TTL microcomputer ROM v1
-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader
Use [Up/Down] to select
Press [A] to start program
Hold [Start] for reset




TTL microcomputer ROM v1
-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader
5:54 / 12:15

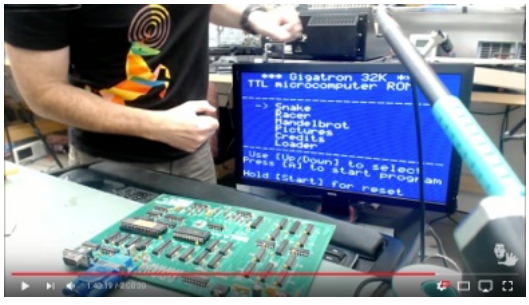

Gigatron TTL Microcomputer

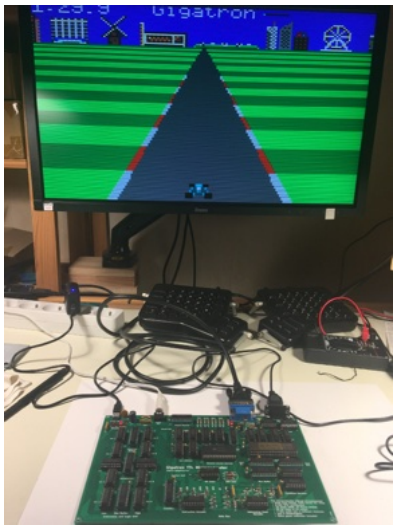



*** Gigatron 32K ***
TTL microcomputer ROM v1
-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader
Use [Up/Down] to select
Press [A] to start program
Hold [Start] for reset


*** Gigatron 32K ***
TTL microcomputer ROM v1
-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader
Use [Up/Down] to select
Press [A] to start program
Hold [Start] for reset


Gigatron


*** Gigatron 32K ***
TTL microcomputer ROM


Gigatron


-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader
Use [Up/Down] to select
Hold [Start] for reset


1andelbrot
00 09
gigatron


*** Gigatron 32K ***
TTL microcomputer ROM v1
-> Snake
Racer
Mandelbrot
Pictures
Credits
Loader

Appendix B

Simulator

The gigatron is so simple that it can be simulated in just a few lines of C code. We include it here to make it easier to understand what the TTL chips do. Hardware-wise there is nothing more to it! Most functionality is in fact coming out of software. You can find the annotated ROM source code on our website https://gigatronttl.eu/.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct { // TTL state that the CPU controls
  uint16_t PC;
  uint8_t IR, D, AC, X, Y, OUT, undef;
} CpuState;

uint8_t ROM[1<<16][2], RAM[1<<15], IN=0xff;

CpuState cpuCycle(const CpuState S)
{
  CpuState T = S; // New state is old state unless something changes

  T.IR = ROM[S.PC][0]; // Instruction Fetch
  T.D  = ROM[S.PC][1];

  int ins = S.IR >> 5;       // Instruction
  int mod = (S.IR >> 2) & 7; // Addressing mode (or condition)
  int bus = S.IR&3;          // Busmode
  int W = (ins == 6);        // Write instruction?
  int J = (ins == 7);        // Jump instruction?

  uint8_t lo=S.D, hi=0, *to=NULL; // Mode Decoder
  int incX=0;
  if (!J)
    switch (mod) {
      #define E(p) (W?0:p) // Disable AC and OUT loading during RAM write
      case 0: to=E(&T.AC);                              break;
      case 1: to=E(&T.AC); lo=S.X;                      break;
      case 2: to=E(&T.AC);         hi=S.Y;              break;
      case 3: to=E(&T.AC); lo=S.X; hi=S.Y;              break;
      case 4: to=  &T.X;                                break;
      case 5: to=  &T.Y;                                break;
      case 6: to=E(&T.OUT);                             break;
      case 7: to=E(&T.OUT); lo=S.X; hi=S.Y; incX=1; break;
    }
  uint16_t addr = (hi << 8) | lo;

  int B = S.undef; // Data Bus
  switch (bus) {
    case 0: B=S.D;                          break;
    case 1: if (!W) B = RAM[addr&0x7fff]; break;
    case 2: B=S.AC;                         break;
    case 3: B=IN;                           break;
  }

  if (W) RAM[addr&0x7fff] = B; // Random Access Memory

  uint8_t ALU; // Arithmetic and Logic Unit
  switch (ins) {
    case 0: ALU =        B; break; // LD
    case 1: ALU = S.AC & B; break; // ANDA
    case 2: ALU = S.AC | B; break; // ORA
```

```
    case 3: ALU = S.AC ^ B; break; // XORA
    case 4: ALU = S.AC + B; break; // ADDA
    case 5: ALU = S.AC - B; break; // SUBA
    case 6: ALU = S.AC;     break; // ST
    case 7: ALU = -S.AC;    break; // Bcc/JMP
  }

  if (to) *to = ALU; // Load value into register
  if (incX) T.X = S.X + 1; // Increment X

  T.PC = S.PC + 1; // Next instruction
  if (J) {
    if (mod != 0) { // Conditional branch within page
      int cond = (S.AC>>7) + 2*(S.AC==0);
      if (mod & (1 << cond)) // 74153
        T.PC = (S.PC & 0xff00) | B;
    } else
      T.PC = (S.Y << 8) | B; // Unconditional far jump
  }
  return T;
}

void garble(uint8_t mem[], int len)
{
  for (int i=0; i<len; i++) mem[i] = rand();
}

int main(void)
{
  CpuState S;
  srand(time(NULL)); // Initialize with randomized data
  garble((void*)ROM, sizeof ROM);
  garble((void*)RAM, sizeof RAM);
  garble((void*)&S, sizeof S);

  FILE *fp = fopen("gigatron.rom", "rb");
  if (!fp) {
    fprintf(stderr, "Error: failed to open ROM file\n");
    exit(EXIT_FAILURE);
  }
  fread(ROM, 1, sizeof ROM, fp);
  fclose(fp);

  int vgaX=0, vgaY=0;
  for (long long t=-2; ; t++) {
    if (t < 0) S.PC = 0; // MCP100 Power-On Reset

    CpuState T = cpuCycle(S); // Update CPU

    int hSync = (T.OUT & 0x40) - (S.OUT & 0x40); // "VGA monitor" (use simple
                                                 // stdout)
    int vSync = (T.OUT & 0x80) - (S.OUT & 0x80);
    if (vSync < 0) vgaY = -36; // Falling vSync edge
    if (vgaX++ < 200) {
      if (hSync) putchar('|');              // Visual indicator of hSync
      else if (vgaX == 200) putchar('>');   // Too many pixels
      else if (~S.OUT & 0x80) putchar('^'); // Visualize vBlank pulse
      else putchar(32 + (S.OUT & 63));      // Plot pixel
    }
    if (hSync > 0) { // Rising hSync edge
      printf("%s line %d xout %02x t %0.3f\n",
             vgaX!=200 ? "~" : "", // Mark horizontal cycle errors
             vgaY, T.AC, t/6.250e+06);
      vgaX = 0;
      vgaY++;
      T.undef = rand() & 0xff; // Change this once in a while
    }
    S=T;
  }
  return 0;
}
```