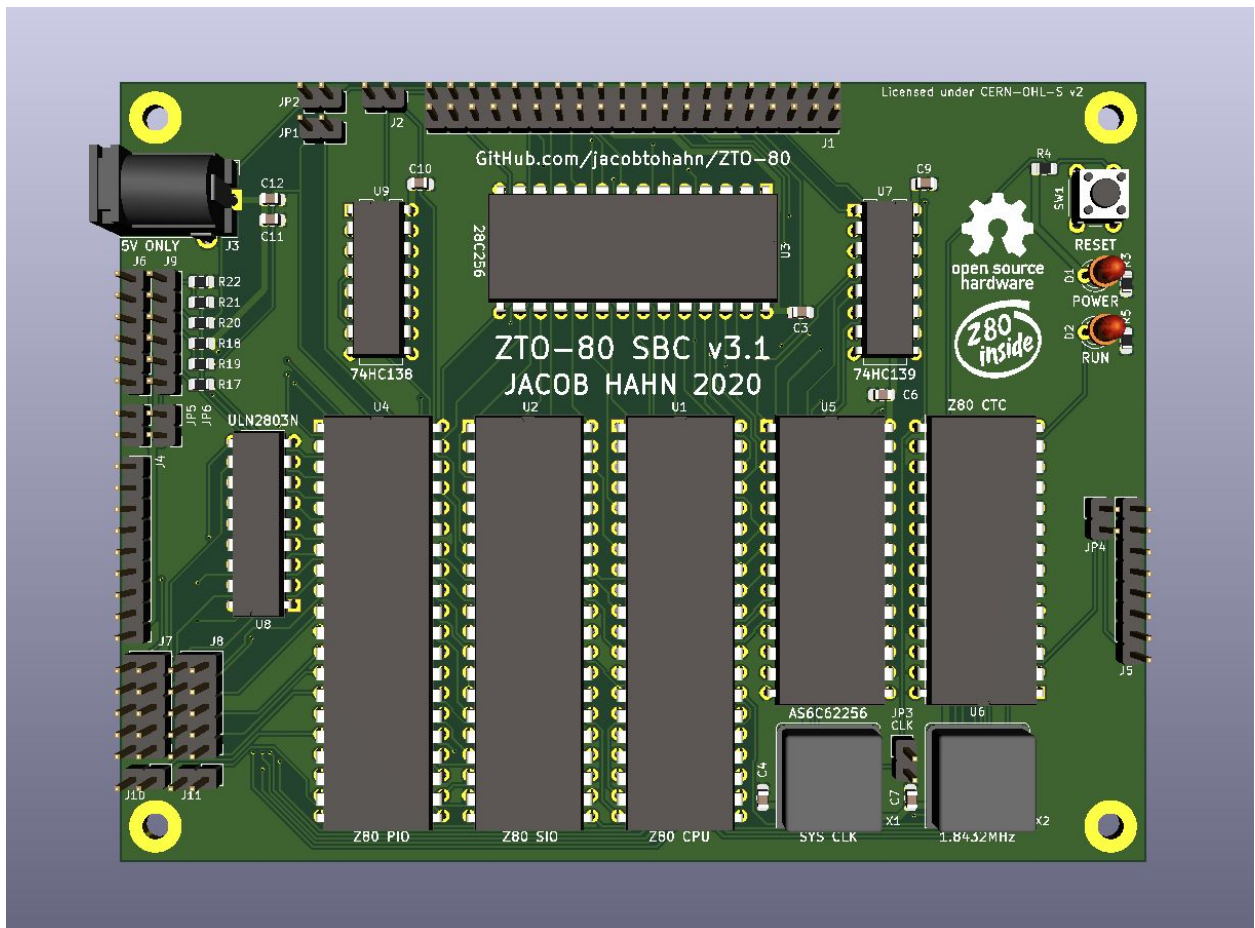


# ZTO-80 SBC

v 3.1



Written by Jacob Hahn

September, 2020

<https://github.com/jacobtohahn/ZTO-80>

Edition 1.2

# Table of Contents

<b>Introduction</b>	<b>2</b>
<b>System Overview</b>	<b>3</b>
CPU	3
Memory	3
I/O	4
Schematic	6
PCB	7
Headers and Jumpers	8
<b>Assembly</b>	<b>10</b>
Components List	10
Tools	14
Assembly Steps	14
<b>Software Setup</b>	<b>18</b>
Flashing BASIC	18
Using your SBC with your PC	22
<b>Troubleshooting</b>	<b>25</b>
<b>Contact</b>	<b>26</b>
<b>License</b>	<b>27</b>

# Introduction

The ZTO-80 SBC is a custom-built single board computer built around the Z80 CPU from Zilog. It was designed from the ground up to be the ultimate board for beginners to learn to develop for the Z80, while also being expandable enough to allow advanced users to have a capable platform to work with.

The goal of this document is to be a guide for getting started with the ZTO-80 SBC, even with no prior experience with the Z80. It starts with a basic overview of the system, and from there dives deeper into the specific parts. It also includes an assembly guide for building the system from the bare PCB.

As new revisions of the ZTO-80 SBC are released, I will update this guide to reflect the newest changes. The newest version of the guide can always be found on my GitHub page, which is linked on the cover of this document.

If you have any problems or encounter any errors in this document, feel free to open up an issue on the GitHub page linked on the cover.

# System Overview

Here's a general overview of the ZTO-80 SBC's onboard hardware. See the [schematic](#) for memory and I/O addresses.

## CPU

The heart of the ZTO-80 SBC is the Z80 CPU, designed by Zilog. It has an 8-bit data bus, a 16-bit address bus, 208 bits of internal register space, and can run at speeds of up to 20MHz, making it a powerful CPU for its class. It can directly address up to 64K of memory using its address bus.

## Memory

The memory of the ZTO-80 SBC is split into two sections: ROM and RAM.

### ROM

The system contains 32K of ROM space, which is used to store programs. It is never written to by the CPU; instead, the user must program it with whatever software they would like to use beforehand. The CPU reads this program and executes the instructions contained within it. ROM keeps its contents even when power to the system is removed.

### RAM

The other 32K of address space is taken by RAM. This is the CPU's working memory. It can be directly written to and read by the CPU and is used to store data such as program variables. However, unlike ROM, all data stored on RAM is lost when power to the system is removed.

# I/O

The Z80's 8-bit I/O space is split among multiple devices on the ZTO-80 SBC. These devices each have a different purpose and provide features to the system.

## SIO

The Z80 SIO is a peripheral chip made by Zilog that provides serial I/O functionality. It features two serial ports, programmable functions such as baud rate and character length, and modem control signals on both ports.

On the ZTO-80, both serial ports are connected to [FTDI pinout](#) compatible headers (standard on most TTL serial to USB cables). All features can be customized in software, allowing for incredibly versatile serial functionality.

## PIO

The Z80 PIO is a peripheral chip made by Zilog that provides parallel I/O functionality. It features two parallel channels, each capable of functioning either as an input or output, four modes of operation for each port, and handshaking functionality on both ports.

On the ZTO-80 SBC, both ports are connected to headers for full user access. Channel B is also connected to a darlington transistor array, allowing it to output up to 500mA on a single output at a time, and up to 120mA on all outputs at the same time. This can be used for applications such as driving motors.

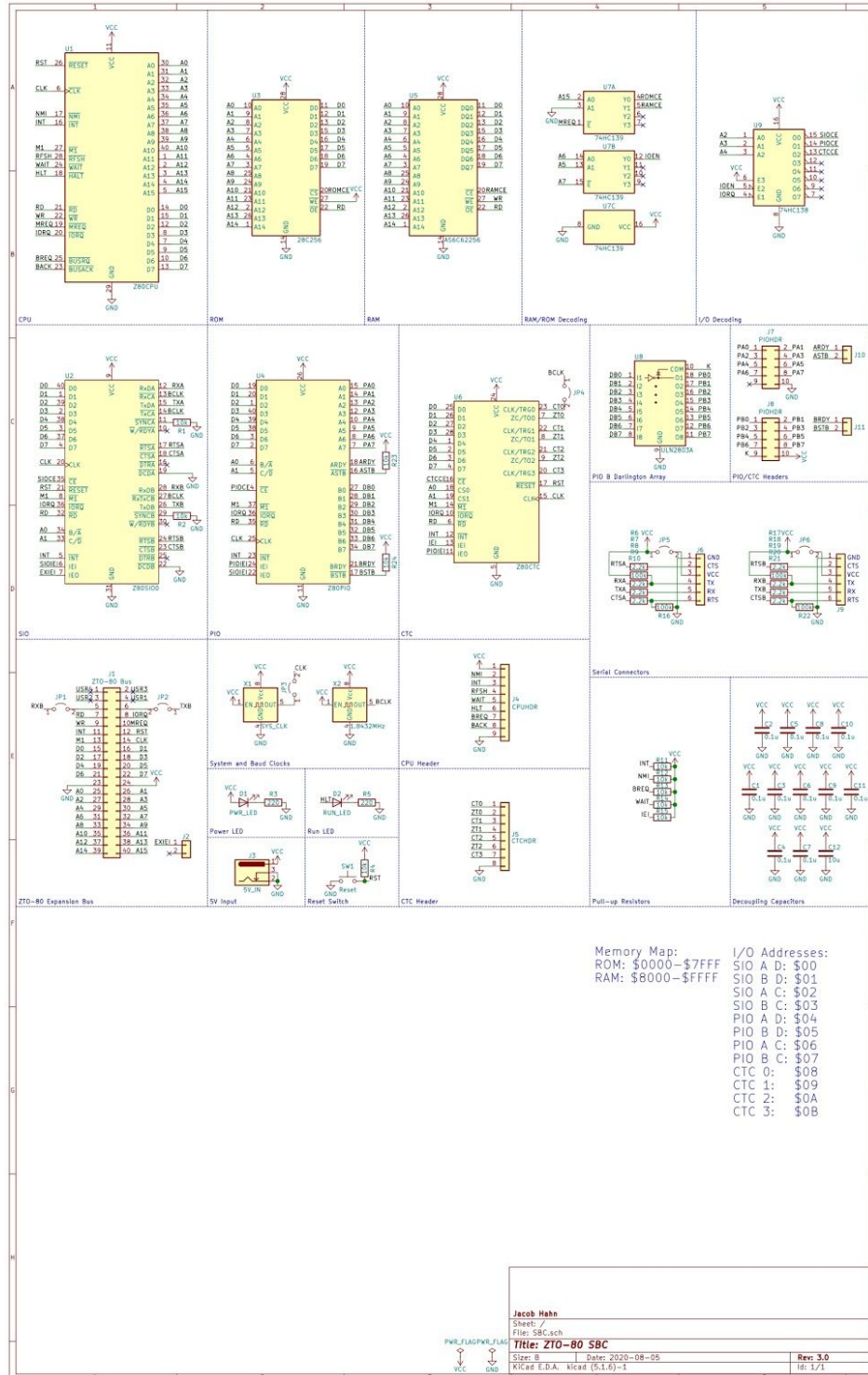
## CTC

The Z80 CTC is a peripheral chip made by Zilog that provides counting and timing capabilities to the SBC. It contains 4 individual counters/timers, all with trigger inputs and the first

three with zero count outputs. It can send an interrupt to the CPU when a timer reaches zero, which makes it easy to implement delays in software without the use of loops.

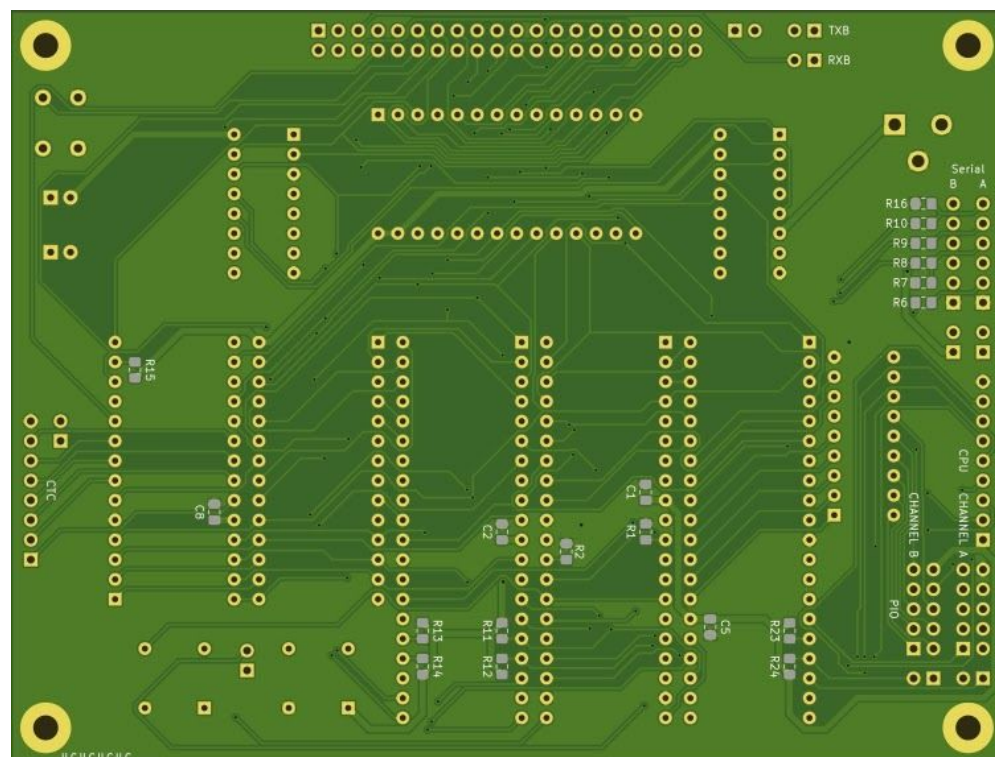
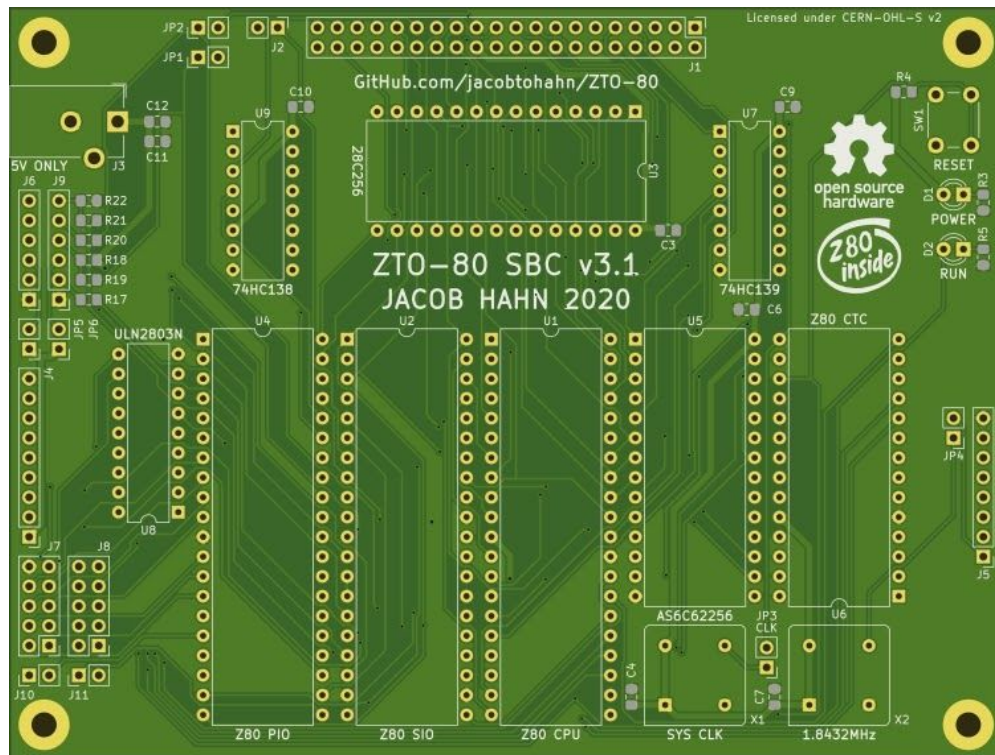
The CTC is mostly controlled by software, with very little external circuitry required. The only header for the CTC breaks out the trigger inputs and the zero count outputs, so that it is easy to connect timer/counter units to each other to create larger counters and longer time delays.

# Schematic





# PCB





# Headers and Jumpers

This section describes the various headers and jumpers located on the ZTO-80 SBC. Take a look at the schematic for pinouts.

## **J1 and J2: Expansion Header**

J1 is the standard ZTO-80 Expansion Header. It is the same header used to connect any modular system module to a backplane.

J2 is added on to the main header to add support for the Z80 Family interrupt priority structure.

## **J4: CPU Control Signal Header**

J4 breaks out the control signals used by the CPU to provide a way to more directly control the Z80 using external signals.

## **J5: CTC Control Signal Header**

J5 breaks out the channel control (trigger and zero count) signals from each of the 4 CTC channels. You can use jumpers on these pins to tie channels together for larger timers or counters.

## **J6 and J9: TTL Serial Headers**

J6 and J9 are the TTL serial outputs from channel A and B, respectively. They are designed to be compatible with the standard [FTDI pinout](#), so any generic TTL serial to USB cable should be compatible with them.

## **J7 and J8: PIO Headers**

J7 and J8 are channels A and B, respectively, of the Z80 PIO. They include the 8 bits of channel I/O, GND (VCC on channel B), and J8 contains a cathode pin for the ULN2803A.

## **J10 and J11: PIO Handshaking Headers**

J10 and J11 are headers that break out PIO handshaking pins on channels A and B, respectively.

## **JP1 and JP2: Expansion Bus Serial Jumpers**

JP1 and JP2 connect the TX and RX pins of SIO channel B to the TXB and RXB pins of the expansion header J1.

## **JP3: Clock Enable**

JP3 disconnects or connects the system clock. It is useful for when you need to drive the clock externally for testing or debugging.

## **JP4: CTC Channel 0 Trigger to Baud Clock**

JP4 connects the trigger input pin of CTC channel 0 to the baud clock. This allows the user to create software time delays by dividing the clock signal.

## **JP5 and JP6: External Serial Power Jumpers**

JP5 and JP6 connect the system VCC to the VCC pin of either TTL serial port, allowing the port to either provide power to the system or for the system to provide power to a serial device. **Important:** leave these jumpers OPEN if receiving power from the 5V barrel jack! Only connect one power source at a time!

# Assembly

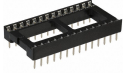


Below is a guide on how to assemble the ZTO-80 SBC, starting with a list of components.

## Components List

Image	Qty.	Description	Reference
	1	ZTO-80 SBC PCB	N/A
	1	Z84C00xxPEG Z80 CPU*	U1
	1	Z84C40xxPEG Z80 SIO/O*	U2
	1	28C256 ROM	U3
	1	Z84C20xxPEG Z80 PIO*	U4
	1	AS6C62256 RAM	U5
	1	Z84C30xxPEG Z80 CTC*	U6
	1	74HC139 RAM/ROM Decoding	U7

	1	ULN2803A Darlington Transistor Array	U8
	1	74HC138 I/O Decoding	U9
	1	System Clock Oscillator	X1
	1	1.8432MHz Oscillator	X2
	1	2x20 Pin Header	J1
	1	PJ-102A Barrel Jack	J3
	1	1x9 Pin Header	J4
	2	2x5 Pin Header	J7, J8
	2	1x6 Pin Header	J6, J9
	9	1x2 Pin Header	J2, J10, J11, JP1-JP6

	1	1x8 Pin Header	J5
	2	220 Ohm 0805 Resistor	R3, R5
	10	10k Ohm 0805 Resistor	R1, R2, R4, R11-R15, R23, R24
	8	2.2k Ohm 0805 Resistor	R6, R8-R10, R17, R19-R21
	4	100k Ohm 0805 Resistor	R7, R16, R18, R22
	11	0.1uF 0805 Capacitor	C1-C11
	1	10uF 0805 Capacitor	C12
	1	6x6mm Tactile Pushbutton	SW1
	2	Red 3mm LED	D1, D2
	3	40-Pin DIP Socket	U1, U2, U4

	3	28-Pin DIP Socket	U3, U5, U6
	2	16-Pin DIP Socket	U7, U9
	1	18-Pin DIP Socket	U8

\*The “xx” in this Z80 Family product number represents the clock rate of the part. Replace it with the correct clock rate when searching for the part online. For example, if you replace the “xx” with “08”, the part has a 8MHz clock rate. “10” would represent a 10MHz clock, etc. Be sure that the clock rate of your part is less than or equal to the speed of your system clock.

# Tools

To build and use your SBC, you will only need a few basic tools:

- A **Soldering iron** and **solder** to solder your connections
- **Rosin flux** (preferably in the form of a flux pen or paste flux) to help solder SMD components
- **Helping Hands** to hold your board off the table and hold parts to the board
- **Tweezers** to allow you to hold SMD components
- **Isopropyl Alcohol** and a **toothbrush** to clean your board after assembly
- A **USB to 5.5mm Barrel Jack Cable** or other 5v power source to power your SBC
- A **USB to serial cable** ([TTL serial](#)) to communicate with the SBC

## Assembly Steps

Follow these steps to assemble your ZTO-80 SBC:

### Step 1: SMD Passives

SMD components may seem terrifying to beginners at soldering, but with just a tiny bit of practice, they are very easy to work with. If you've never worked with them before, I would recommend trying an SMD soldering learning kit, which you can find for cheap on Amazon.

I've made a video about how to solder SMD passives and you can find it [here](#). There are also plenty of other tutorials and different methods that you can find with just a Google search.

It is **important** to match the component values to their respective locations. These can be found in the components list table.



Both capacitors and resistors used here are not polarized and can be rotated either direction.

## **Step 2: Headers**

This is where the Helping Hands come into play. Because you can't bend the pins of the headers to hold them in place, you need to find a different way to hold them straight. One way is to use some Blu-Tack to hold the header in place. Another is to use one hand of the Helping Hands to hold the board and the other hand to hold the header in place. Once you solder one pin, you can continue as normal.

## **Step 3: Sockets**

Install the IC sockets into the reference locations found in the components list. **It is important to match the orientation of the socket with what is indicated on the silkscreen of the PCB!** To hold the socket in place as you're soldering so it doesn't fall out, you can bend down the pins in opposite corners after you insert it into the board. Don't install the actual ICs yet.

## **Step 4: Other Components**

All the other components are straightforward to solder, and you can bend the pins of most to hold them in place. When it comes time to solder the LEDs, make sure that they are rotated the correct way, with the flat side of the lens facing the square pad on the PCB.

## **Step 5: Clean the Board**

Using a toothbrush or other stiff brush, apply isopropyl alcohol and clean any flux residue from the board. Flux looks like a clear to yellow-brown liquid around solder joints. It's difficult to remove,

but with the help of lab wipes you should be able to remove most of it. Clean as much off as possible.

Before installing your ICs, plug in your power source and make sure the power LED lights.

For extra reassurance, test the input voltage to make sure it is around 5 volts (+/- 0.5 volts).

## **Step 6: ICs**

You shouldn't ever be directly soldering ICs to a board; if they fail or if you need to use them somewhere else, it would be very difficult to remove them. Instead we use sockets, which you have already soldered to the board.

Insert the ICs into their proper sockets. Be sure they are not inserted backwards. The pin 1 marking (usually a semicircle engraving on one end of the IC) should match up with the pin 1 marking of both the socket and silkscreen (a similar semicircle engraving on the socket and a marking on the silkscreen).

IC pins may be bent slightly outwards from the factory, which could make it difficult to insert them into the sockets. To fix this, slightly insert one set of legs into the socket, and then carefully apply pressure to the IC perpendicular to the long side of the socket until the other set of legs are above their holes in the socket. Then, push down to insert all the legs.

## **Final Checklist:**

- ☐ All components are installed and there are no empty footprints
- ☐ All SMD parts are in their proper places
- ☐ All pins have been soldered on all components
- ☐ There are no solder bridges between component legs
- ☐ Both LEDs are soldered in the correct orientation

- ❑ Both sockets are soldered in the correct orientation
- ❑ All ICs are installed in the correct orientation
- ❑ All headers are exactly perpendicular to the board

# Software Setup

Now that your ZTO-80 SBC is fully assembled, it's time to hook it up to a computer and use its built-in BASIC interpreter. However, before this can happen, you need to flash the BASIC ROM file to your ROM chip.

You can use either a dedicated programmer like [this one](#) or use an [Arduino as a programmer](#) to flash the EEPROM. The steps after step 1 below are for the linked dedicated programmer, the TL866. You can follow guides for an Arduino programmer online.

## Flashing BASIC

### Step 1: Download the ROM file

To get the latest ROM file for the SBC, head over to the [Releases](#) page on my GitHub and download the latest release of the ROM32K.hex file.



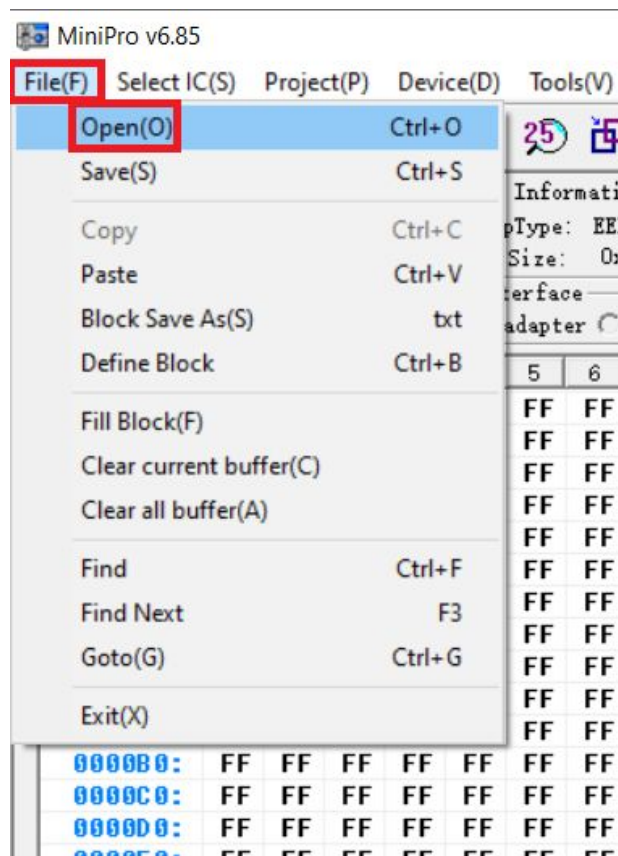
The screenshot shows the GitHub release page for 'ZTO-80 BASIC v0.0.1'. On the left sidebar, there is a 'Latest release' badge, the version 'v0.0.1', a commit hash '35a0f09', and a 'Compare' button. The main content area displays the release title 'ZTO-80 BASIC v0.0.1' in blue, followed by the author 'jacobtohahn' and the release date '9 days ago' along with '11 commits'. Below this, it states 'Initial release of ZTO-80 BASIC. Tested to work on SBC V2.0.' A section titled 'Assets 3' contains three items: 'ROM32K-v0.0.1.hex' (highlighted with a red box), 'Source code (zip)', and 'Source code (tar.gz)'.

## Step 2: Open ROM32K in the programmer's software

Next, you're going to want to open up the downloaded hex file in your programmer's software, which was either provided with the programmer on a CD or could be downloaded online. Check the included instruction manual for more details.

First, plug your programmer into your computer with the included USB cable.

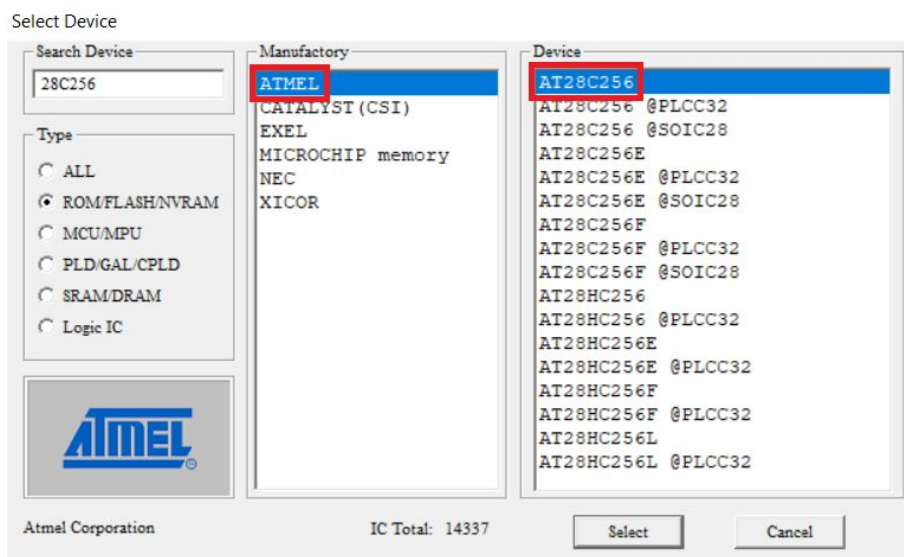
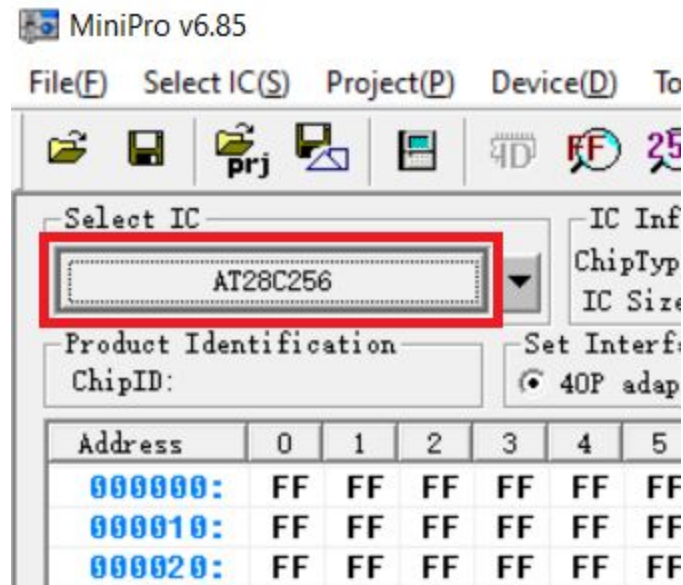
Select File>Open and navigate to the downloaded ROM.



### Step 3: Choose your ROM IC

Now that you have your ROM file open, you need to tell the programmer what chip we are programming.

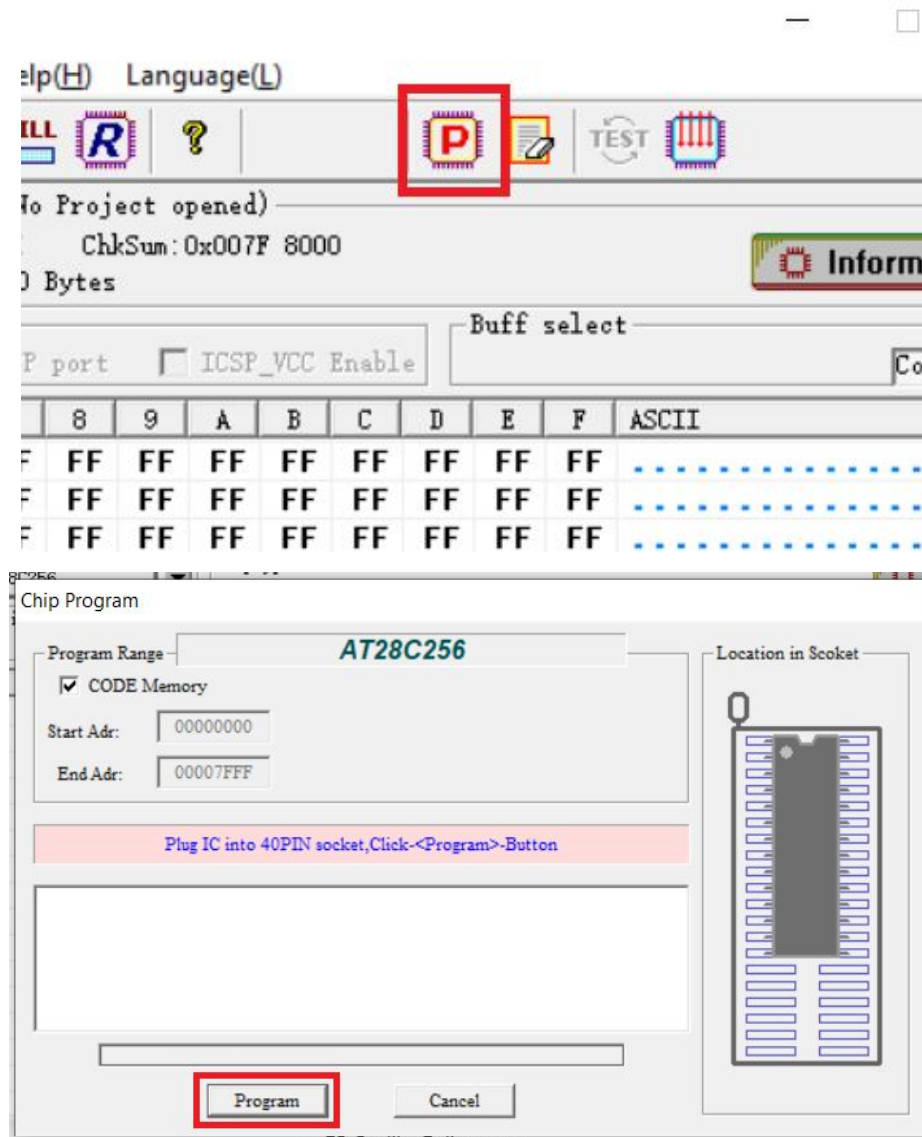
Click on the “Select IC” button near the top left, and search for “AT28C256”. Choose the one by Atmel with no “@” suffix in the list.



### Step 4: Program the ROM

Now that your programmer knows what ROM chip it's working with, it's time to program it.

In the top bar of the programming software, click the icon of the IC with the letter “P” in the middle. In the menu that comes up, use the “Location in Socket” diagram to properly insert your IC into the programmer. Select “Program” and wait for the process to finish. Then, you can click cancel, remove the IC from the programmer, and plug it into the ZTO-80 SBC. You’re good to go!





# Using your SBC with your PC

The ZTO-80 SBC communicates with your computer over a serial connection. In order to see this connection, your computer needs a software called a serial terminal emulator. I personally use RealTerm because it is free and easy to use, and supports a variety of serial formats. Other emulators include PuTTY, TeraTerm, ZTerm, Termite, and dozens of others.

Due to the number of terminal emulators out there, I won't be giving a specific tutorial; I will only give the settings that you should use with your software.

## Step 1: Install your software

Use a search engine to find the download page of your preferred serial terminal emulator and download the version for your operating system.

## Step 2: Connect your SBC

The ZTO-80 SBC uses serial channel A to communicate with a computer. Thus, you should connect through the channel A TTL serial header, J6, using a serial to USB cable. On your PC, be sure to install the driver for whichever serial-to-USB chip your cable uses. This info should be in the manual or the product listing.

Connect both the serial and power cables.

## Step 3: Set up the port

Now that you've plugged in your SBC, you can set up the terminal to communicate with it.

The first thing to do is to select the serial port that your SBC is connected to. There should be a box to select a serial (COM) port in your terminal emulator. Most computers these days have no hardware serial ports built in, so there should only be one option to choose from, which is guaranteed to be your serial-to-USB cable. If it doesn't show up, make sure you installed the driver correctly.

### **Step 4: Set up the baud rate**

The next option is the baud rate. This option selects how fast serial data is transmitted and received. The dropdown to select it is likely located somewhere near where you selected your serial port. There should be many values, but the one we're looking for is 57600 baud. This is pre-programmed into the SBC, and both the terminal and SBC must use the same baud rate.

### **Step 5: Set up handshaking**

Lastly, you need to set the handshaking. This allows the SBC to tell the computer if it is out of buffer space to store incoming data. Find the option for "RTS/CTS" handshaking and set it to that. While the SBC will work without handshaking, it won't be able to handle an overflow of data, which could cause problems in some cases.

After this, apply your settings if required.

## Step 6: Test the SBC

With both your power and serial cables plugged in, press the reset button on the SBC, located above the power LED.

If everything was set up correctly, you should see this text appear in your terminal:

```
ZTO-80 By Jacob Hahn  
Original Code Copyright Grant Searle  
ZTO-80 BASIC v1.X.X Startup
```

```
Cold or warm start (C or W)?
```

Press C to start BASIC with a fresh boot. You will then see:

```
Memory top?
```

This is asking for the top of the RAM space that BASIC can use. Press return to let it select the maximum amount of RAM.

Finally, you should see:

```
Z80 BASIC Ver 4.7b  
Copyright (C) 1978 by Microsoft  
32382 Bytes free  
Ok
```

This means that your SBC is working!

The “Ok” is called the BASIC *prompt*, which lets you know that BASIC is awaiting input from the user.

You may now enter BASIC commands. See the manual [here](#) for more info on how to use BASIC.

# Troubleshooting

To troubleshoot your SBC, I recommend that you open an issue on the [project's issues](#) page. Before doing so, try filling out the [final checklist](#) to see if you may have missed anything simple.

If you've done all that, feel free to open an issue. Here's a couple of things to keep in mind when doing so:

- Search through the issues to make sure someone hasn't already asked the same question
- Filter results by the "good first issue" label to find common beginner issues
- Use the "SBC Troubleshooting" issue template, which automatically adds the "SBC troubleshooting" label

# Contact

If you need to contact me, the best way is through email. The next best way is to send me a message on Hackaday.io, although I can't guarantee I will see it. Please do not contact me about buying or selling anything unless it has to do explicitly with the ZTO-80 project.

Email: [jtohahn@gmail.com](mailto:jtohahn@gmail.com)

Hackaday.io profile: [hackaday.io/jtohahn](https://hackaday.io/jtohahn)

# License

If you're not building off of this project, this doesn't really apply to you, but if you are, here are the license details you are required to follow:

Copyright Jacob Hahn 2020.

This source describes Open Hardware and is licensed under the CERN-OHLS v2. You may redistribute and modify this documentation and make products using it under the terms of the CERN-OHL-S v2 (<https://cern.ch/cern-ohl>). This documentation is distributed WITHOUT ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING OF MERCHANTABILITY, SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE. Please see the CERN-OHL-S v2 for applicable conditions.

Source location: <https://github.com/jacobtohahn/ZTO-80>

# History

8/14/2020 - Edition 1.0.

8/18/2020 - Edition 1.1, added license, added changes from SBC revision, and fixed minor typos.

9/15/2020 - Edition 1.2, updated for SBC v3.1, fixed grammar/typos.