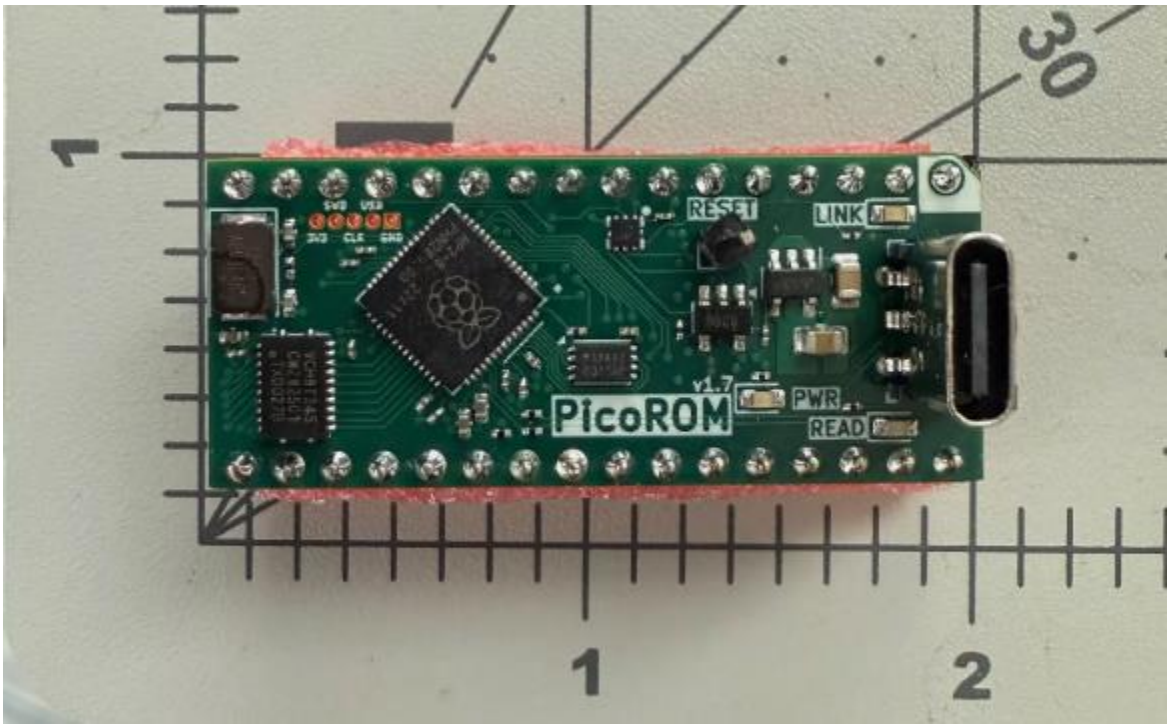
PicoROM



Contents

Purpose

This document describes how to use the commercially available PicoROM tool as an EPROM emulator.

This is extremely useful if you are developing older ROM based products because you can download your new ROM image over USB – instead of having to constantly erase and re-program physical ROM chips.

Where to Get PicoROM

You can purchase the PicoROM on EBAY here: https://www.ebay.com/itm/235981681314

At the time of this document, it costs $39.99.

I later found out that you can also buy it here: https://www.tindie.com/products/pbretro/picorom/

And it only costs $28.00!

Other Files you will Need

Go to the following site and read the instructions: https://github.com/wickerwaka/PicoROM

It is not immediately obvious, but there is a link in this site (several pages down in the 'installation' section) that gives you a link to the tools. At the time of this document the link was: https://github.com/wickerwaka/PicoROM/releases/tag/v1.7.1 but will change as future releases come out.

You need to download the 'picorom.exe' file appropriate for your operating system. I'm using Win64 so I got "picorom-x86_64-pc-windows-msvc.exe" and renamed it to "picorom.exe"

Put picorom.exe wherever is convenient for you – I put it in the same folder where my ROM image is generated.

Important Notes

Renaming your PicoROM Device

Use the following command to identify all connected USB PicoROM devices.

```
C:\yourDir>picorom list
```

Then using the unique ID shown in the list command, rename it as follows:

```
C:\yourDir>picorom rename E66138528361BB3 yourName
```

Now you can use ‘yourName’ instead of ‘E66138528361BB3’ (or whatever your ID was) in the commands.

Uploading ROM Images

This was the most confusing part of the tool which caused me hours of frustration.

The command to upload a ROM image is as follows:

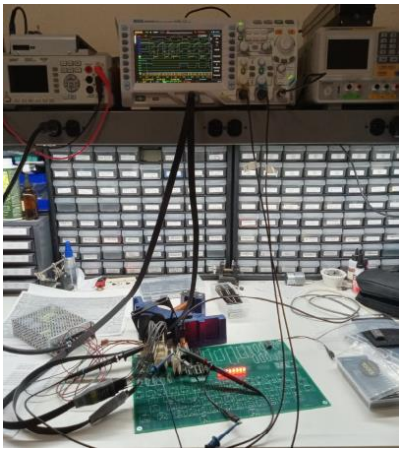
C:\yourDir>picorom upload yourName ROMImage.ext {-s} {64Kbit}

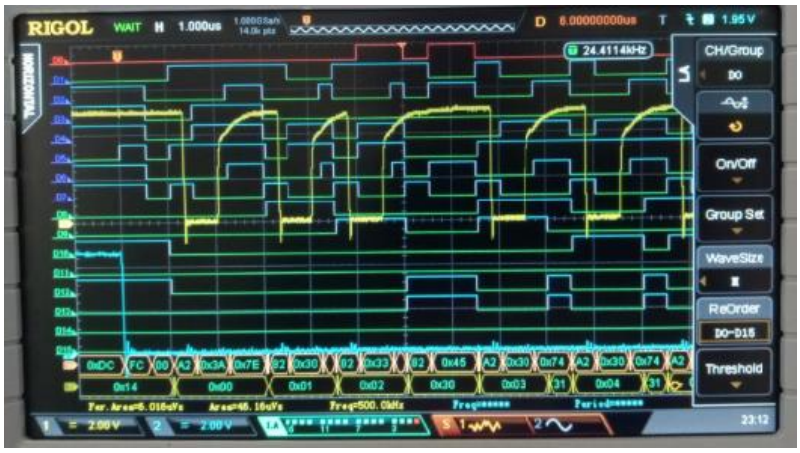
Where:

- {-s} optionally commits the image to flash – you can optionally do this later with the 'commit' command as well.
- {64Kbit} optionally tells the tool the size of the device to emulate so that addresses above this range will wrap back to zero. {64Kbit} emulates a 2764 8Kbyte ROM.

ROMImage.ext MUST BE A BINARY IMAGE

I wasted hours because the tool gave NO error message when I tried to use a hex file image – so I thought it worked properly and processed the hex file. It was only after spending hours setting up a logic analyzer so I could see the data being returned from the PicoROM after reset.





The data was 0x3a, 0x30, 0x33 for the first 3 memory read operations.

The first line of my hex file was:

:03000000C3D30067

Naturally, I was expecting the first three accesses to be 0xc3, 0xd3, 0x00.

However, the tool treated the hex file "AS THOUGH" it were a binary image – so the first 3 bytes in the rom were: 0x3a (':'), 0x30 ('0'), 0x33 ('3').

It would be much better, in my opinion if the tool rejected any file that did NOT have a '.bin' extension.

Hex to Binary Conversion

You can get a program that will convert hex to binary files here:

https://sourceforge.net/projects/srecord/files/srecord-win32/

and run it using the following command:

srec_cat.exe HexFile.hex -Intel -o BinaryFile.bin -Binary

For more information please refer to the following page:

https://srecord.sourceforge.net/

Don’t try to Read the PicoROM Contents with a PROM Burner

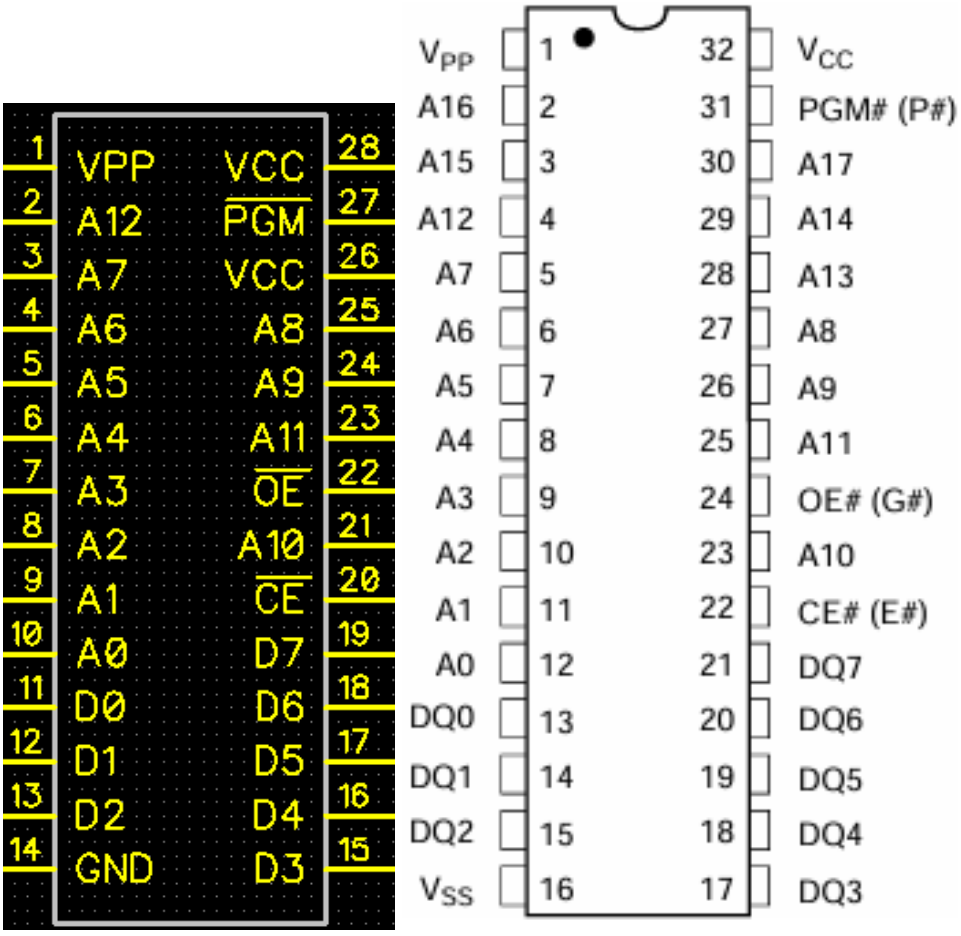
I initially tried this (before setting up the logic analyzer) to try and see what values were being returned from the device.

However, my burner (Dataman Pro) issued stern warnings about this possibly damaging the burner and voiding the warranty.

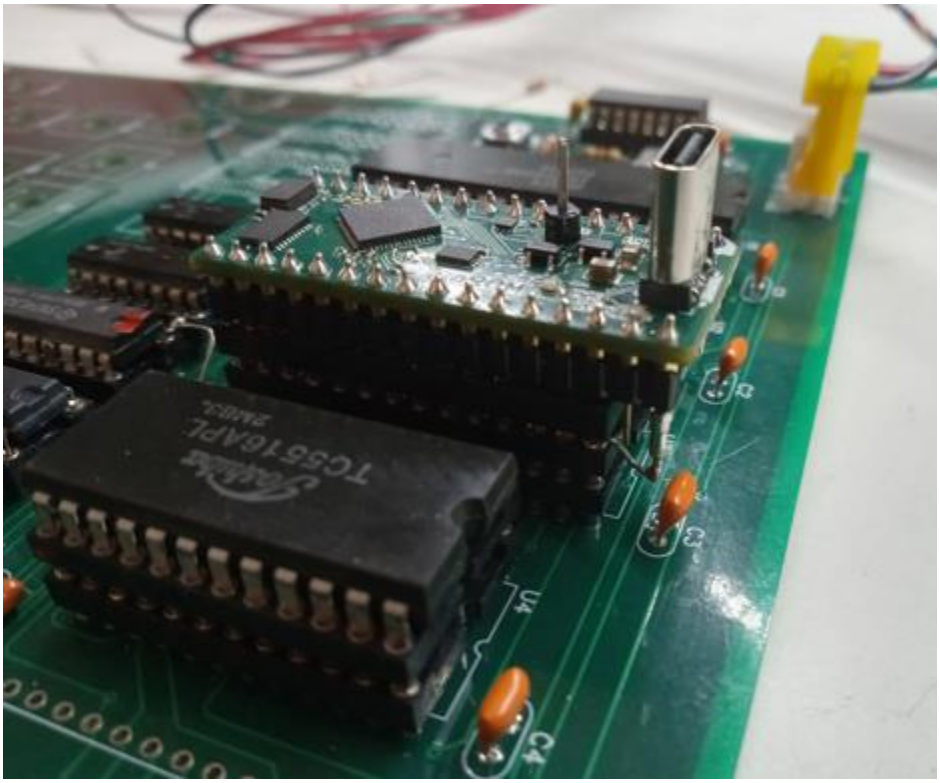
Emulating Smaller ROMs

PicoROM is a direct plug-in replacement for a 27C020 ROM. However, and especially when dealing with older designs, this is way too large a ROM and you need to adapt it to emulate something smaller – like a 2764.

Fortunately, all of the common ROM DIP packages have backwards compatible pinouts – in that the socket gets bigger with each larger ROM size, but the common pins stay the same. So, as you can see below left, a single socket layout will accommodate 2716/2732/2764 ROMs. And below right the PicoROM 27C020 adds another 4 extra pins on the top.



Because the PicoROM is larger than the old design chips, it ‘hangs’ off the edge – see image below.



In order to power the PicoROM, it needs to either (or both):

1. Be connected to a powered USB cable
2. Have pin 32 connected to +5V

As can be seen above, a simple way to do this is to use a few machine sockets to raise the PicoROM up, and solder a wire from pin 28 of the 2764 PCB socket pin (+5V) to the PicoROM pin 32. In this example, I pulled a pin out of the top socket pin 28 and used it to connect to the PicoROM pin 32 – so that the PicoROM could easily be removed, if necessary, without having any permanent alterations.

At first, I thought it necessary to also remove pins 1, 26, and 27 from the socket (so that the multiple +5V connections are not connected to the PicoROM upper address pins) but this is not necessary as long as you correctly specify the ROM size in the “upload” command.

```
C:\yourDir>picorom upload yourName ROMImage.ext -s 64Kbit
```

When the ‘64Kbit’ is specified, this tells the PicoROM to simply ‘ignore’ all address lines above the 8Kbyte limit (A13 and above.) So, it doesn’t matter that these upper address lines are tied high.