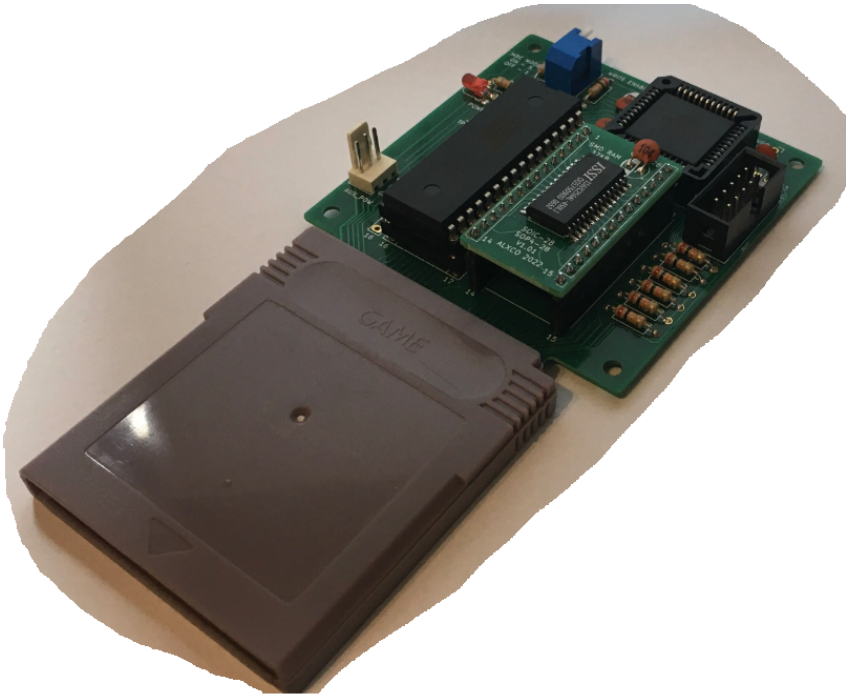
SQUAREBOI

An open-source GB/GBC Flashcart system



ALXCO 2022

Table Of Contents

-Acknowledgments

-Disclaimer & License

-Introduction

-Building Your Own
- ->Necessary Tools
- ->Choosing Which Boards To Make
- ->Sourcing Parts
- ->Assembling The Boards
  - ->>Squareboi Mainboard
  - ->>SMD RAM Daughterboard
  - ->>SMD ROM Daughterboard
  - ->>Stackable ROM Daughterboard
  - ->>Writeboi Programmer
  - ->>Cartridge Shell
- ->Burning Firmware
  - ->>Squareboi Mainboard
  - ->>Writeboi Programmer
- ->Final Assembly

-Flashcart Usage

-How Does It Work?
- ->Gameboy Cartridge Theory Of Operation
- ->Major ICs Used
- ->CPLD Code
- ->Adding Squareboi Support To Existing Programmers
- ->Writeboi Serial API

-Appendixes
- ->I - CPLD Development Board
- ->II - ROM Adaptor Board
- ->III - Game Compatibility

Acknowledgments

Squareboi's existence is only possible due to the continued efforts of multiple individuals dedicated to reverse engineering and documenting the Game Boy hardware.

This project builds heavily upon the work of Joonas Javanainen of Gekkio.fi, in particular his clear and concise documentation of the MBC's behavior, as well as accurate footprints for the cartridge connectors.
https://gekkio.fi/

The Game Boy Development Community's Pan Docs still provide the most comprehensive technical description of the Game Boy's inner workings, 27 years after first publication. It was instrumental for understanding the console's memory mapping, and interpreting the cartridge header data.
https://gbdev.io/

Alex of InsideGadgets was a major source of data and inspiration all throughout the project. His exceedingly thorough build log for a 2MB MBC5 cart helped navigate around many potential pitfalls.
https://www.insidegadgets.com/?/

Tauwasser's VHDL models for MBC1 and MBC2 proved invaluable in capturing the behaviour of these proprietary chips, and formed the basis of what would eventually become the MBC51 firmware.
https://wiki.tauwasser.eu/view/Main_Page

Disclaimer & License

Squareboi is the work of a hobbyist - And while significant care was taken to make it reliable, I cannot guarantee that it will work in your specific use case, or that it will be compatible with your hardware.

Squareboi is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0).
https://creativecommons.org/licenses/by/4.0/#deed-conditions

In essence, if you share or adapt this work, you must credit the author (ALXCO), and provide a link to the Squareboi repository, at:
https://github.com/ALXCO-Hardware/squareboi

Parts of this work (specifically, the footprints for the Game Boy cartridge connector and slot) are the creation of Joonas Javanainen, and licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0).
https://creativecommons.org/licenses/by/4.0/#deed-conditions
https://gekkio.fi/

This work uses the PySimpleGui library, which is licensed under the GNU Lesser Public License 3.0 (LGPL 3).
https://www.gnu.org/licenses/lgpl-3.0.en.html
https://pysimplegui.readthedocs.io

This work uses the PySerial library, which is Copyright (c) 2001-2020 Chris Liechti <cliechti@gmx.net> All Rights Reserved.
https://pyserial.readthedocs.io/

This work uses the Arduino KiCad Library, which is Copyright 2017-2018, Nicholas Parks Young, and licensed under the GNU LGPL v2.1.
https://github.com/Alarm-Siren/arduino-kicad-library/

Introduction

Over 40 years after its original introduction, the Nintendo Game Boy/Game Boy Color remains one of the best selling and most influential video game consoles of all time.

However, no new official cartridges for the system have been produced since 2002, and existing alternatives are either limited in functionality, of dubious build quality, or closed-source and hard to obtain.

The Squareboi project seeks to provide enthusiasts with an open-source flashcart that is functionally equivalent to an original cartridge, made with readily available parts, and compatible with the majority of the GB/GBC library. An Arduino-based cartridge reader is also provided, capable of interfacing with a variety of existing cartridges.

A basic version of the Squareboi can be built entirely with through-hole components, making for a good intermediate-level DIY soldering project.

It is my hope that you will find the Squareboi useful – Either for running homebrew software, backing up your existing games and save data, or simply to get a better understanding of how that brave little console from 1989 really works.

Features:

>Up to 4MB ROM/32KB RAM storage space

>Stores save data without needing a battery

>Made of common, off-the-shelf parts

>Programmer works as a general purpose cartridge reader

>Easy to build: A basic version can be made with entirely through-hole components

>Powered by a single 5V rail

Building Your Own

Necessary Tools

A few tools are needed to assemble and program the board. Here's a basic list:

- A soldering iron, and your choice of solder. One with a small tip would be ideal, and a wedge tip if you intend to use daughterboards with SMD components.
- An USB-Blaster style JTAG programmer, for uploading the firmware to the CPLD. The cheap reproduction ones should be sufficient for this.
- An Arduino Mega 2560, to connect the Writeboi board to your computer.

You will also need the following pieces of software:

- UrJTAG, the Universal JTAG Library, Server and Tools, to program the CPLD. Available at http://urjtag.org/
- The Arduino IDE, to program the Arduino Mega. Available at https://www.arduino.cc/en/software
- KiCad 6, to open the schematic files and check component placement for each board. Available at https://www.kicad.org/

In addition, a few other tools can be helpful, if not strictly necessary:

- A hot air rework station. A hobby model should suffice- It can be useful if you make a mistake while soldering an SMD part and need to remove it from the board.
- A good quality multimeter and a magnifying glass, to verify your work after soldering.
- A PLCC extractor tool, in case you need to remove the CPLD from its socket after it is inserted.
- A Game Boy, Game Boy Color or Game Boy Advance. Aside from allowing you to test if the device works, it is a convenient way to power the cartridge while writing the firmware to it.

Choosing Which Boards To Make

At a minimum, you will need to build the Squareboi mainboard itself, and the Writeboi programmer board.

If you want to run programs that require external RAM to function, or save data, you will need to build the SMD RAM daughterboard.

If you want to store ROMs larger than 512KB and up to 2MB, you will need to build the SMD ROM daughterboard.

Finally, if you want to store ROMs up to 4MB in size, you will need to build a pair of the stackable SMD ROM daughterboards instead.

If using a second-hand CPLD, it is recommended that you also build the CPLD development board found in appendix I, as normal firmware upload might fail when using a chip that has been written to before.

Sourcing Parts

The most critical parts to procure are the boards themselves. There are a variety of prototype PCB manufacturers available online, the majority of which should have the capability to produce the two-layer boards used. Further details about the print settings needed are located in each board's section.

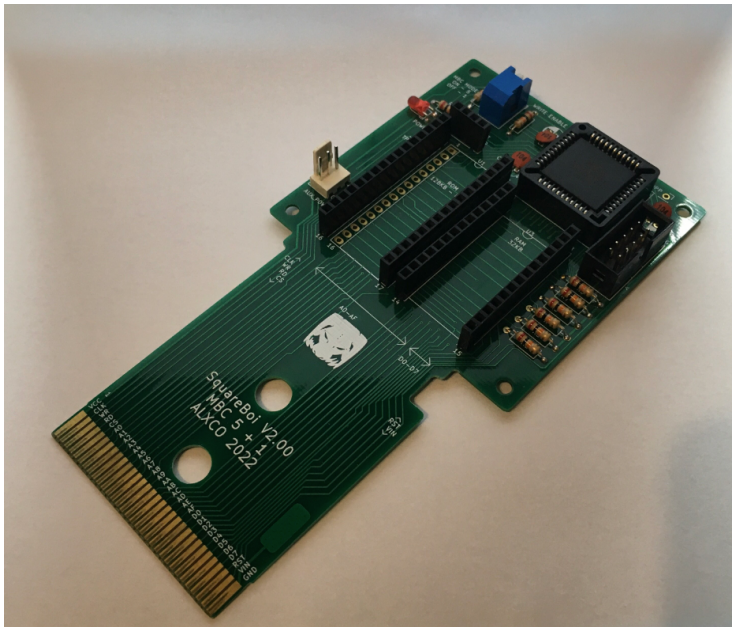
The passive components used are not critical, and can be sourced locally. Most resistors are marked as 20K Ohm, but any quarter-Watt resistor between 10K and 30K Ohm should suffice. Similarly, the 100K Ohm resistor on the Writeboi board could be replaced by any value between 100K and 300K, if needed. All resistors were specified at a 1/4W power rating as it is the most commonly sold one, but there should be no issue using 1/8W ones if needed.

For the capacitors, any ceramic disc capacitor with a value of 100nF or above will do, provided it can fit in a 2.5mm footprint.

For the ICs, however, it is highly recommended to source them from a reputable electronics dealer (Mouser, Digikey, Farnell etc) to make sure you are receiving a new product. The CPLD and Flash Memory chips have a limited number of write cycles, and a used part could be at the end of its lifespan. In addition, they also could have write protection enabled, which is not trivial to remove. If you must use a second-hand CPLD, please refer to Appendix I – You will need to build a separate development board.

Similarly, it is important to make sure you have good quality IC sockets and pin headers. The parallel buses used have a lot of pins, and a loose connection can lead to intermittent issues that are hard to diagnose.

Squareboi Mainboard



Gerber file location: Hardware/KiCad 6/Mainboard/Mainboard_Gerbs.zip

Board printing options: two-layer, ENIG surface finish, 1mm board thickness

Bill Of Materials:

CPLD socket: PLCC44 socket, through-hole

Programming socket: 10-pin JTAG socket, aka 10-pin shrouded box header. Can be replaced by a 2x5 pin header if necessary.

Power LED: Any standard red LED with a 2.5mm pin pitch

Resistors: 20KOhm 1/4W through-hole resistor, 7 units

Capacitors: 100nF ceramic disc capacitor, 2.5mm pin spacing, 3 units

Mode switch: 2-position DIP switch, 90 degree orientation

RAM header: 1x14 2.54mm female header, 2 units

ROM header - One of the following options:
If using DIP ROM: DIP-32 socket, turned pin, 15.24mm width

Otherwise, 1x16 2.54mm female header, 2 units
plus
1x4 2.54mm female header

Auxiliary power input: 1x3 2.54mm pin header, ideally keyed (optional)

Integrated Circuits:

CPLD: Microchip/Atmel ATF1502ASL, PLCC44 package

DIP ROM – Optional. One of the following options:

Microchip SST39SF010A, DIP32 package – 128KB capacity

Microchip SST39SF020A, DIP32 package – 256KB capacity

Microchip SST39SF040, DIP32 package – 512KB capacity (Recommended)

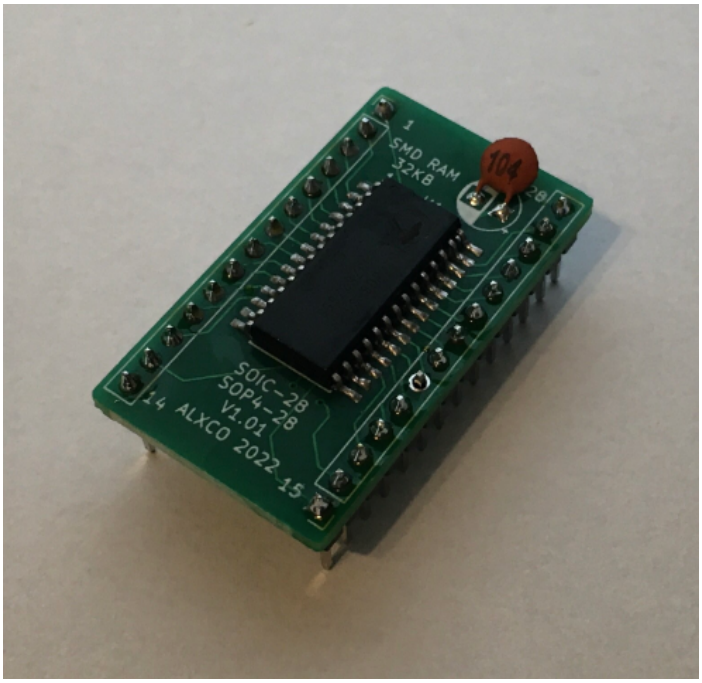
Notes:

> It is extremely important to use a gold-plated surface finish on this board, ENIG being the recommended one. Solder-based finishes like HASL just don't have the durability to be used in an edge connector, and are unlikely to last more than a few insertions.

If your PCB manufacturer can provide hard gold fingers for the edge connector, that could be used to provide additional longevity for the cartridge. This is not mandatory, however – standard ENIG shows no significant abrasion even after extensive use.

Original cartridges are made with 0.8mm thick boards, however in testing this proved to be too flexible for a PCB this size. 1.0mm board thickness was chosen as a compromise, as it seems to fit the cartridge slot easily while providing enough stiffness. If desired, it is possible to file the edge of the PCB at a slight angle to help with insertion.

SMD RAM Daughterboard



Gerber file location: Hardware/KiCad 6/RAMBoard/RAMBoard_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

Capacitor: 100nF ceramic disc capacitor, 2.5mm pin spacing

RAM header: 1x14 2.54mm pin header, 2 units

Integrated Circuit:

If saving data with FRAM is desired:

Cypress/Infineon FM18W08, SOIC28 package (Recommended)
or
Cypress/Infineon FM1808B, SOIC28 package

Otherwise, any 5V, JEDEC-compliant 32KB parallel SMD SRAM with a SOP-28 or SOIC-28 package should work. The following were validated to work with the system:

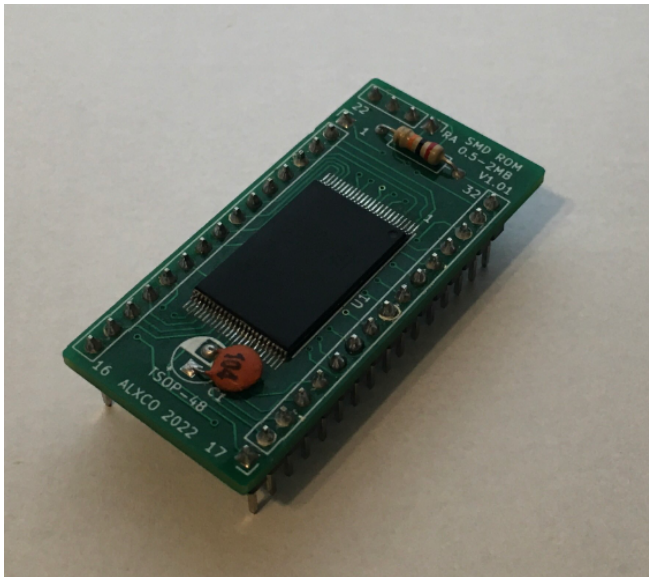
ISSI IS65C256AL, SOP-28 package
or
ISSI IS62C256AL, SOP-28 package

Notes:

At the time of writing this document, FRAM ICs are available in limited supply, and are likely to be the most expensive individual component of the build, at roughly 12 dollars per unit.

If saving data is not a necessity, and you only want to run software that needs the additional cartridge RAM space, there should be no downside to the use of a standard SRAM module instead.

SMD ROM Daughterboard



Gerber file location: Hardware/KiCad 6/ROMBoard/ROMBoard_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

Resistor: 20KOhm 1/4W through-hole resistor

Capacitor: 100nF ceramic disc capacitor, 2.5mm pin spacing

ROM header: 1x16 2.54mm pin header, 2 units
and
1x4 2.54mm pin header

Integrated Circuit:

Alliance M29F200FB, TSOP48 package – 256KB capacity (Not recommended)
or
Alliance M29F400FB, TSOP48 package – 512KB capacity (Not recommended)
or

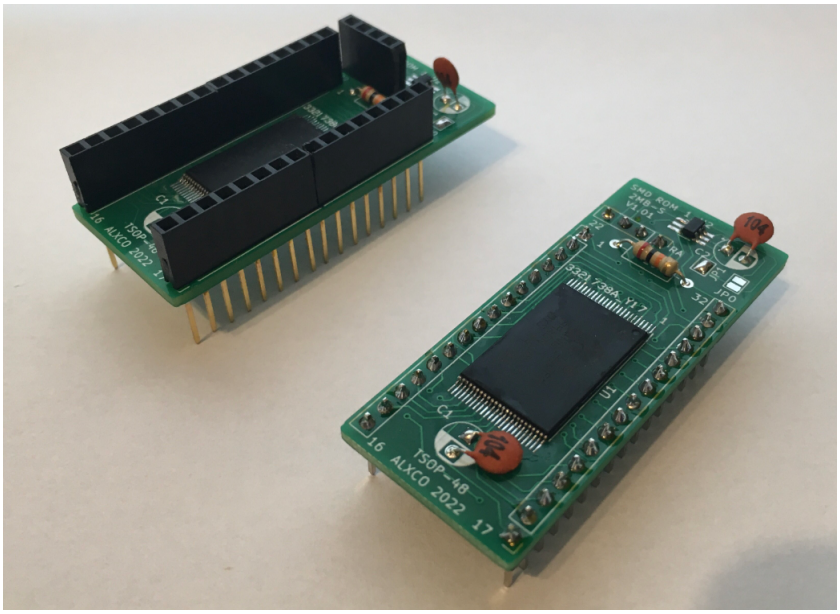
Alliance M29F800FB, TSOP48 package – 1024KB capacity
or
Alliance M29F160FB, TSOP48 package – 2048KB capacity (Recommended)

Notes:

The TSOP-48 package used in this board has very tight pin spacing, and can be difficult to solder. It is recommended to use magnification and verify your work carefully, to avoid the risk of shorts.

For this reason, it is only recommended that you build this board if you plan on using the higher capacity ICs – For 512KB and below, going with DIP memory is an easier solution.

Stackable ROM Daughterboard



Gerber file location: Hardware/KiCad 6/ROMBoard/ROMBoard_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

Resistor: 20KOhm 1/4W through-hole resistor

Capacitor: 100nF ceramic disc capacitor, 2.5mm pin spacing

ROM header:
1x16 2.54mm pin header, 2 units
and
1x4 2.54mm pin header

or stackable header versions of the above

Integrated Circuits:

Alliance M29F160FB, TSOP48 package – 2048KB capacity

SN74LVC1G19 1-of-2 demultiplexer, SOT-23-6 package

Notes:

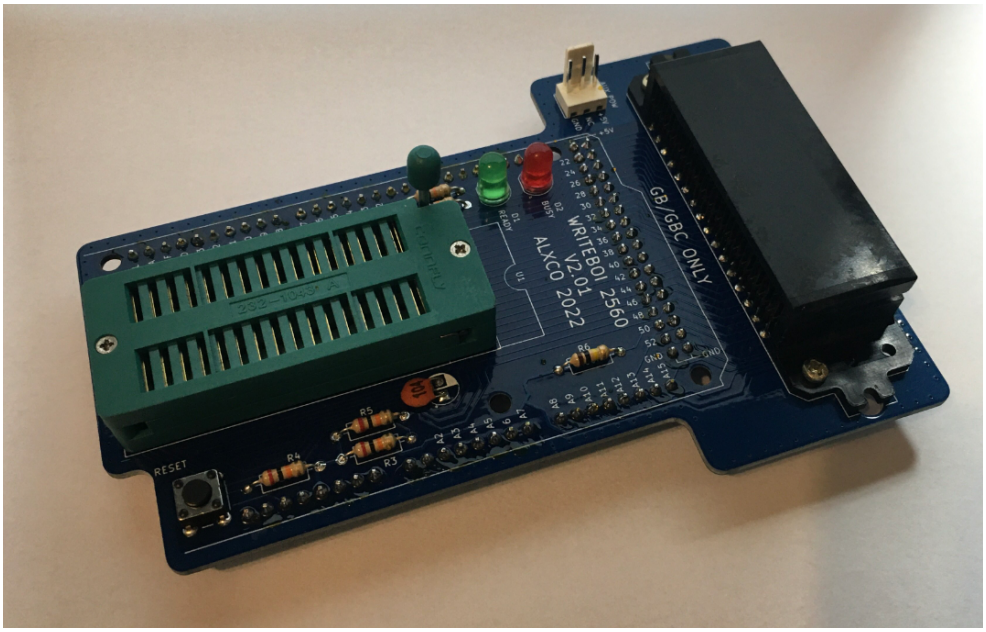
This board is very similar to the standard ROM daughterboard, with the difference that two of them can be connected using stacking headers, to reach a total of 4MB memory capacity.

Do note that only the 2048KB version of the flash IC is compatible with this board.

The demultiplexer used has very small text on its package, which can make pin 1 hard to identify. Take care not to solder it backwards, or it will quickly burn out.

When using a single one of these boards, make sure the JP0 jumper is connected.
When stacking two modules, one must have JP0 set, and the other JP1.

Writeboi Programmer



Gerber file location: Hardware/KiCad 6/Writeboi/Writeboi_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

Resistors: 20KOhm 1/4W through-hole resistor, 3 units
and
10KOhm 1/4W through-hole resistor, 2 units
and
100KOhm 1/4W through-hole resistor

Capacitor: 100nF ceramic disc capacitor, 2.5mm pin spacing

LEDs: Any standard LED with a 2.5mm pin pitch, 2 units, ideally of different colors

Reset switch: Any standard 6mm tactile switch (Optional)

Auxiliary power output: 1x3 2.54mm pin header, ideally keyed (optional)

Pin headers:
1x8 2.54mm pin header, 5 units
and

1x10 2.54mm pin header
and
2x18 2.54mm pin header

DIP memory socket: One of the following:

DIP-32 socket, turned pin, 15.24mm width
or
32-pin DIP ZIF socket

Cartridge connector:

Any replacement cartridge slot for the original Game Boy – Make sure it is the through-hole version, the GBC/GBA connectors will not work.

And

M2.5 machine screw, at least 8mm long, 2 units
and
M2.5 hex nut, 2 units
and
small washer, 2 units

Notes:

Some replacement cartridge slots have very short pins – It is recommended to check the one you purchase against the side of a standard-thickness circuit board to verify. If the pins are not long enough to protrude through the board, it might be convenient to print a 1.2mm thick one instead.

The machine screws and nuts are not mandatory, but will make the cartridge slot more resistant to rough handling.

Cartridge Shell

Bill Of Materials:

Reproduction Game Boy Cartridge Shell, with a small screw

12mm M3 nylon screw, 8 units (Optional)

11mm M3 female-to-female nylon threaded standoff, 4 units (Optional)

3D printed backplate (Optional)
file: Case/Backplate.amf

3D printed backplate, for stackable memory (Optional)
file: Case/Backplate_Stack.amf

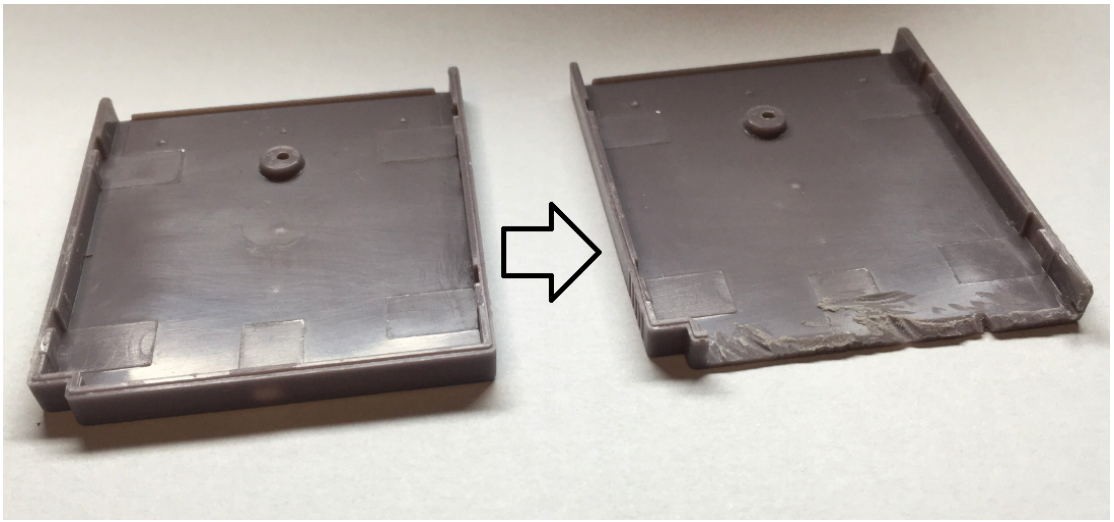
3D printed Backplate, with supports (Optional)
file: Case/Backplate_Supports.amf

All 3D printable parts are available as both FreeCAD FCSTD files, for editing, and exported AMF files, for use with slicers.

Assembly:

The Squareboi Mainboard's lower half is designed to fit in a standard reproduction cartridge shell, commonly available online.

Please note that you will need to cut off a section from the top of the back piece of the shell, as indicated in the picture below:



Cutting pliers or a rotary tool with a cutting disk should work well for this.

Insert the mainboard into the back part of the shell, aligning the round protrusion with the hole in the board, then slide the front half up to close the shell. Secure with a small screw into the plastic from behind.

Place the backplate+supports behind the mainboard, aligning the supports with the IC socket locations. Place the four screws facing in, and secure by threading in the standoffs.

Leave the daughterboards and second half of the backplate off until final assembly, as the board's firmware must be programmed before they are attached.

Burning The Firmware

Squareboi Mainboard

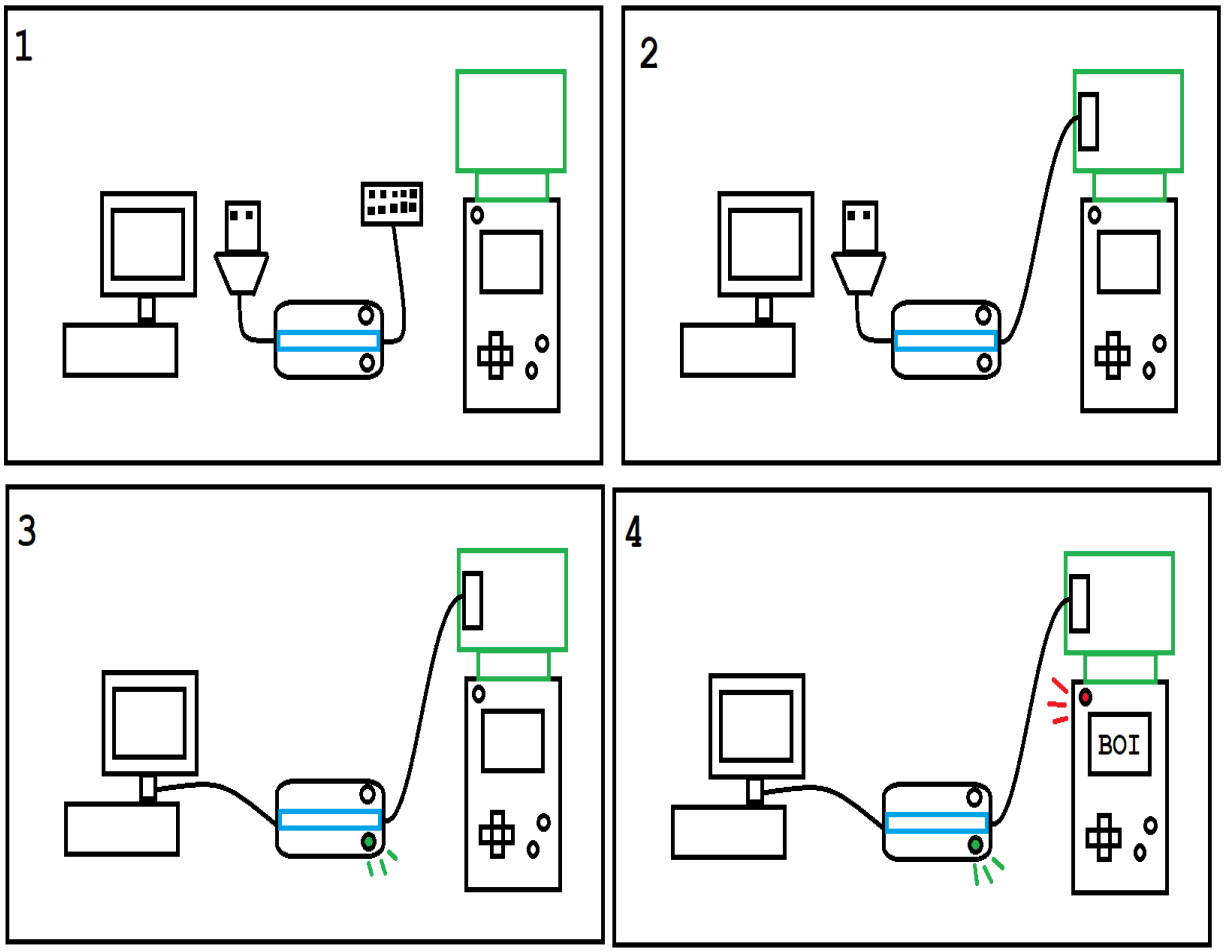
The following steps will most likely not work if you are using a CPLD that has been previously programmed with non-squareboi firmware. If using a second-hand CPLD, please refer to Appendix I – CPLD Development Board;

The JTAG programmer is fragile, and could be damaged if connected in the wrong order. Take care to follow the instructions as closely as possible. As a rule of thumb:

When establishing a connection, first plug in the JTAG header, then connect the programmer via USB to power it, and only then turn on the device to be programmed.

When breaking a connection, first turn off the programmed device, then disconnect the programmer's USB lead, and only then remove the JTAG header.

For reference, follow the image below:



First, make sure UrJTAG is installed on your machine, and no daughterboards are attached to the mainboard. Locate the .svf programming file in the Firmware folder, and copy it to whichever folder UrJTAG is installed in.

> As of writing this document, the correct file name is ATFMBC51_V203.svf

Make sure the Game Boy has a fresh set of batteries, as the writing process can consume a lot of power. Connect the USB Blaster's JTAG header to the Squareboi's, making sure the orientation is correct. Then, connect the USB blaster to the computer being used for programming. Finally, plug the cartridge into the Game Boy, and turn it on.

> Using a Game Boy is a convenient way to power the cartridge, but it's not the only one. It's not possible to keep the Writeboi programmer plugged into the same computer that's writing firmware (The Arduino interferes with the serial port the USB blaster uses), but you could power it with an USB power bank, and then use it to power the cartridge.
>
> Finally, you could use a bench power supply to feed the Squareboi 5V through the AUX_POW header, but this is not recommended, as the data bus will be left floating, which could cause programming to fail.
>
> If you encounter problems while burning the firmware, refer to appendix I – CPLD Development Board.

Run UrJTAG, and type in the following commands, in order:

```
cable usbblaster
detect
```

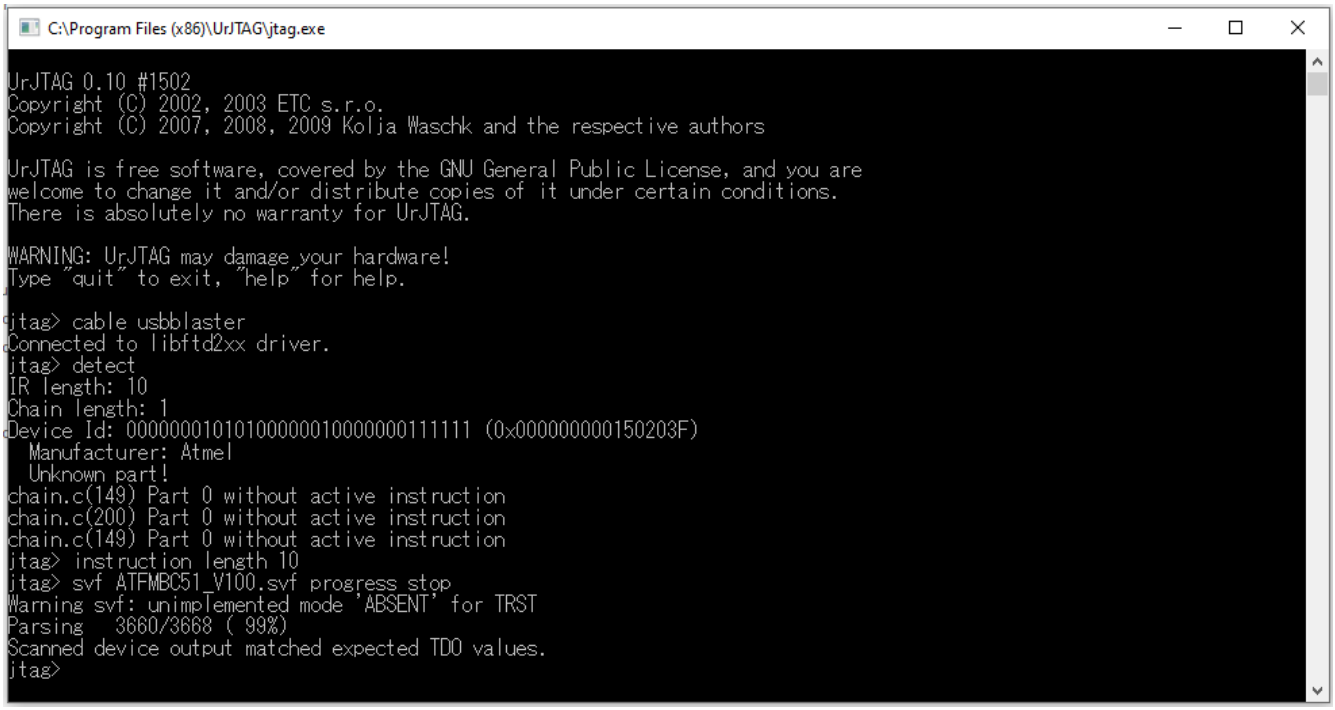
At this point, the program should report that it has detected an Atmel part, with 1502 near the end of its device ID. If not, check your connections and try again.

Then, type in the following commands:

```
instruction length 10
svf [FILENAME].svf progress stop
```

With [FILENAME] being the name of the .svf file for the desired firmware version. The cartridge will then be programmed. This may take a few minutes.

Following is an image of a successful programming attempt:



If programming fails, check your connections and try again. If problems persist, you might need to build a dedicated programming board - Refer to Appendix I - CPLD Development Board.

Otherwise, your CPLD is now programmed.

Finally, close UrJTAG, turn off the Game Boy, disconnect the USB blaster from your computer, and then unplug the JTAG header.

Writeboi Programmer

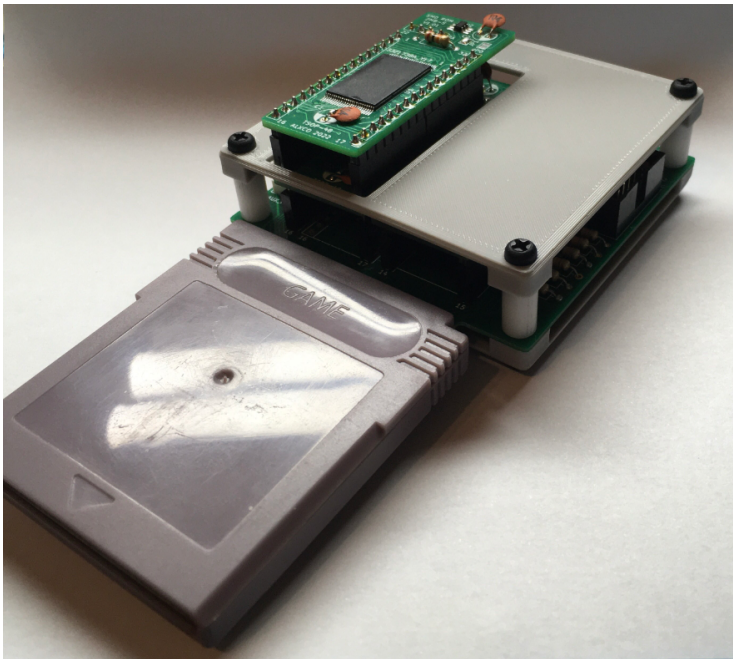
Make sure the Arduino IDE is installed in your machine, and that the USB Blaster is not connected.

Connect the Writeboi to the Arduino, making sure all pins align correctly, then connect it via USB.

Open the Firmware/Writeboi.ino/Writeboi_ino.ino file in the Arduino IDE, then press the Upload button to compile and upload it to the board.

If all goes well, the Writeboi should now be programmed correctly.

Final Assembly



Insert your choice of ROM, either DIP or daughterboard-based, into the leftmost socket. If using the stackable version, insert the one with stacking headers first, and then the other one on top of it.

Insert the RAM daughterboard, if built, into the rightmost socket.

If you're using DIP memory without a cartridge slot on your Writeboi, you'll want to insert it on the DIP slot of the Writeboi and upload your ROM to it now, then insert it onto the Squareboi.

> When inserting daughterboards, make sure the pin number marks in the corners align with the corresponding ones in the mainboard! Damage may occur if they are connected backwards.
>
> When inserting DIP memory, align the notch in the IC with the half-circle mark in the slot.

If using the 3D-printed case, place the second backplate over the nylon standoffs, and secure it with screws.

Flashcart Usage

If you're using Windows, there is a pre-compiled version of the Writeboi GUI in the Releases section of the repository. Do note that the executable is unsigned, which can trigger false positives with some antivirus programs.

Otherwise, you can run the original Python script, found at Software/Writeboi_Source/WriteboiGUI.piw
In that case, you will need to install:

Python, version 3.10 or above:
https://www.python.org/

PySerial:
https://pyserial.readthedocs.io/en/latest/pyserial.html

PySimpleGUI:
https://pysimplegui.readthedocs.io/en/latest/

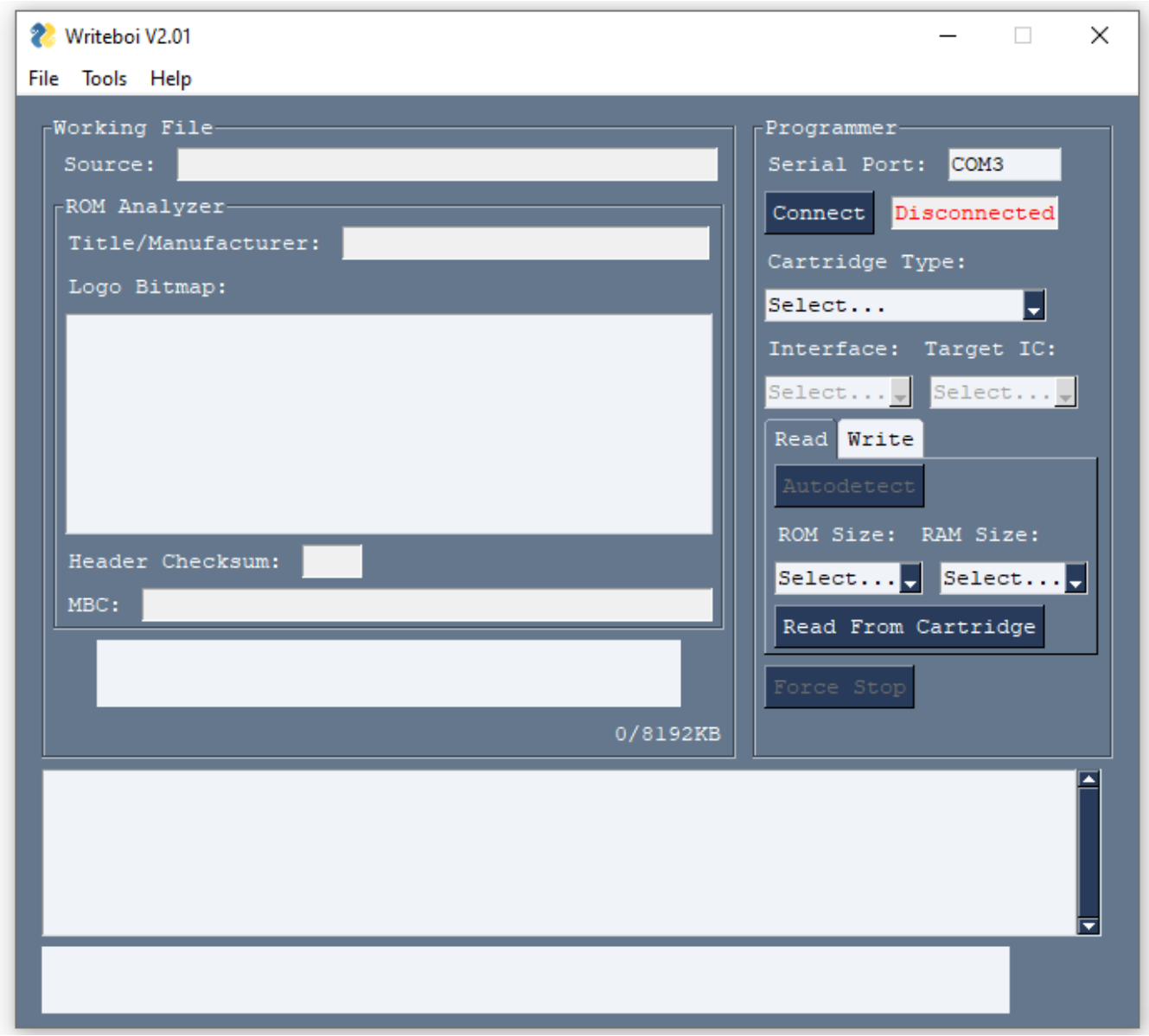
If you prefer, there is also a command line based version that does not rely on PySimpleGUI. Open Software/Writeboi_Source/WriteboiCLI.py using your text editor or IDE of choice, and follow the instructions within.

On either case - Insert the cartridge you want to read/write into the Writeboi's cartridge slot, or the DIP ROM you want to program into the DIP32 socket, making sure it is oriented properly, then connect the Arduino to an USB port.

If using the cartridge slot, make sure the Write Enable DIP switch is set to ON, and that the MBC mode switch is set to the MBC corresponding to the ROM you'd like to use.

Neither the Game Boy cartridges nor the DIP memory were designed to be hot-swapped. Please only insert or remove them when the Writeboi is off, or damage may occur.

Start the Writeboi GUI. You should be greeted with the following screen:



Press the Connect button to initiate communications with the Writeboi.

To write a file the cartridge, first load it into memory using the File/Open menu. Then, select the cartridge type you are using, as well as the MBC mode it is currently set to, using the drop-down menus to the right. Use the Target IC dropdown to select the type of memory chip used, or if you'd like to write to RAM. Finally, change to the Write tab, and press Write To Cartridge. The process should now proceed automatically, and might take up to several minutes, depending on the size of the file.

To read a file from the cartridge, you will first need to know its size. You can input it manually using the ROM Size and RAM Size menus, or use the Autodetect button after selecting the cartridge type. Select if you'd like to read from ROM or RAM using the Target IC dropdown, then press Read From Cartridge to load the data into memory.
At the end of the process, you can use the File/Save menu to save the dumped data into a file.

To verify that everything worked correctly, insert the Squareboi into your Game Boy of choice, make sure the MBC DIP switch is set to the appropriate mode for your ROM, and turn it on. It should now play the same as an original cartridge.

How Does It Work?

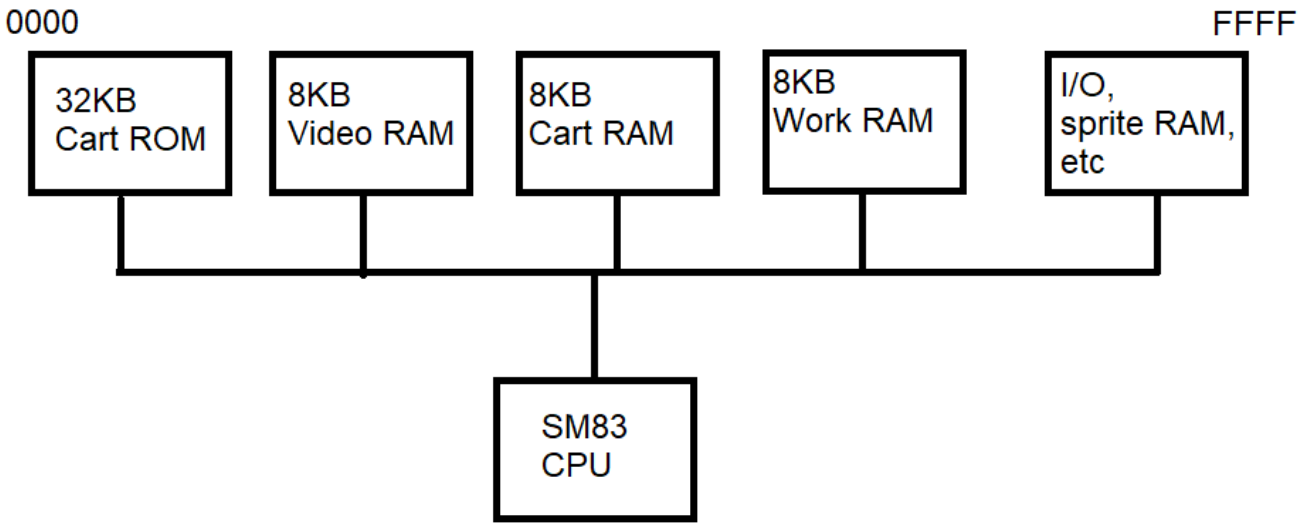
Game Boy Cartridge Theory Of Operation

The Game Boy is, in essence, a very traditional 8-bit computer. The Sharp SM83 CPU (mostly a clone of the famous Z80 microprocessor) has a 16-bit address bus, giving it 64KB of address space, to which all memory and I/O are mapped. The cartridge has access to this memory bus, as well as a series of control signals.

Here is the pinout of the cartridge connector:

1	VCC
	CLK
	/WR
	/RD
	/CS
	(A0-A15)
	(D0-D17)
	/RST
	AUDIO/VIN
32	GND

There are two memory areas of special interest: The first 32KB are mapped to cartridge ROM, and another 8KB sector is mapped to cartridge RAM.

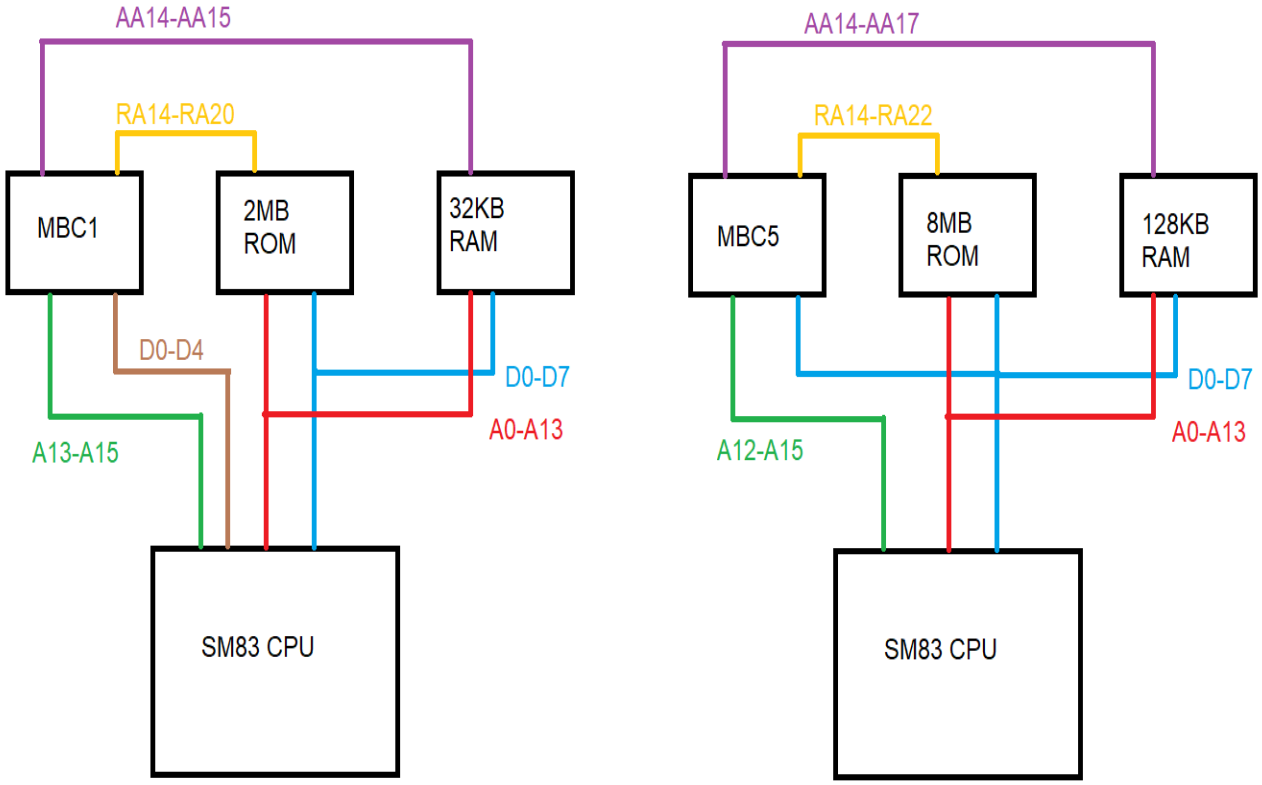


This presents a technical challenge: Most games do not fit in 32KB of storage, and an 8MB cartridge (the largest size officially supported)

would require 22 bits of address for its capacity, not the 15 available within the cart ROM address space.

The solution to this problem was to divide cartridge storage into 16KB Banks, and add a separate IC between the CPU and cartridge ROM. When accessing a new memory bank, the CPU would first write the highest bits of the address to that separate chip, which would then forward them to the ROM chip.
This address-managing chip is known as the MBC, “Memory Bank Controller.” Reproducing its functionality is a large part of Squareboi's function.

Over the lifetime of the Game Boy, many different models of MBC were made, both officially and unofficially, with various supported memory sizes and capabilities (Real Time Clock circuits, Rumble functionality, etc). The Squareboi project implements the two most common ones, MBC1 and MBC5, as they are capable of supporting the majority of available ROMs.
Here is a simplified view of how the various buses are connected for those models:



Note that the MBC also controls some of the cartridge RAM address lines – This allowed developers to implement save files larger than 8KB, or include extra RAM for demanding games within the cartridge itself. Though the MBC5 can support up to 128KB of RAM, the majority of games used 32KB or less.

To write to the MBC, the Game Boy pulls the /WR pin low, while A15 is 0. This writes the contents of the data bus to one of the MBC's four registers, determined by the top 4 bits of the address bus.
Here is a summary of the various registers for each model of MBC:

MBC1:

A15-A12	Name	Function
000X	RAM Gate	Enables RAM if it contains 0xA
001X	ROM Register	5 bits of RA (ROM Address) bus
010X	RAM Register	2 bits of AA (RAM Address/ROM Address2) bus
011X	MODE Register	Selects how AA is used

MBC5:

A15-A12	Name	Function
000X	RAM Gate	Enables RAM if it contains 0x0A
0010	ROM Register 1	Lower 8 bits of RA (ROM Address) bus
0011	ROM Register 2	Upper bit of RA (ROM Address) bus
010X	RAM Register	4 bits of AA (RAM Address) bus

The Squareboi's CPLD implements the functionality of both MBC1 and MBC5, selected by the MBC Mode DIP switch.

In addition, it also implements a change necessary for compatibility with FRAM modules: It strobes the RAM Chip Select line on every positive edge of the Game Boy's clock, allowing the FRAM to perform a pre-charge operation between consecutive memory accesses. This is relevant for programs that use the Game Boy's DMA functions, which normally leave Chip Select low through 160-byte operations.

Major ICs Used

The ROM and RAM ICs used in Game Boy Cartridges use industry-standard pinouts and interfaces, so the replacements were chosen based mostly on features and availability. A choice was made to only use 5V components to make debugging easier, but this also caused some limitations.

The SST39SF series is one of very few 5V DIP flash ICs still in production, and is commonly available on most electronics distributors. Its maximum capacity being limited to 512KB is its only drawback.

The M29F series is essentially the only remaining line of 5V parallel flash with capacity above 512KB. The TSOP-48 package makes soldering it a challenge, unfortunately, but prices are relatively low and it is easy to find online.

The FM18W08 (and its functionally identical twin, the FM1808B) are nonvolatile FRAM ICs that closely mimic the behaviour of normal static RAM modules, allowing them to be drop-in replacements for the normal save memory, with superior data retention. They were chosen as a way to avoid the extra complexity of having a battery-backed save system.

The ATF1502ASL is programmable logic device that is low-cost, operates on a single 5V rail, and, being a clone of the now-discontinued Altera MAX 7000 7032sl, can be programmed using the freely available Quartus 2 Software. This makes it both a versatile prototyping platform, and a good introduction to the world of FPGAs and CPLDs. The amount of I/O pins available is also just enough to implement the MBC's full functionality, with a single one left for debug purposes.

CPLD Code

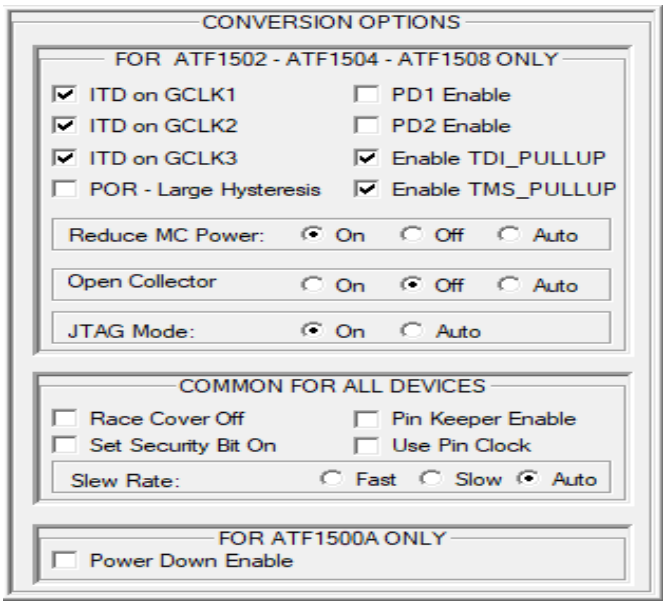
If you wish to edit the CPLD's source code and build your own binary files, you will need a few pieces of software:

>Intel Quartus II, Version 13.0.1 (Do note that this is an older version of the program, as newer ones dropped support for MAX 7000 devices)
https://www.intel.com/content/www/us/en/software-kit/711791/intel-quartus-ii-web-edition-design-software-version-13-0sp1-for-windows.html

>Microchip POF2JED and Microchip ATMISP 7.3
https://www.microchip.com/en-us/products/fpgas-and-plds/spld-cplds/pld-design-resources

The Quartus project can be opened by using Firmware/ATFMBC51_Source/ATFMBC1.qpf, and contains a single VHDL file, ATFMBC1.vhd, which is mostly self-documented.

After compiling, it will produce a programming object file called ATFMBC1.pof on the output_files folder. This file can be converted into a .jed standard programming file using POF2JED with the following options:



Finally, using ATMISP, the .jed file can be converted into a .svf file, that can be written to the CPLD using UrJTAG.

Adding Squareboi Support To Existing Programmers

This section is dedicated to developers of cartridge dumpers and programmers, who might want to support the Squareboi interface with their hardware.

For reading ROM and all RAM operations, the Squareboi behaves transparently like an original cartridge, with two differences caused by the use of FRAM:

- cartridge pin 2 (Clock) must be held Low when reading or writing to RAM
- The RAM IC must be deselected between consecutive accesses, to allow for a pre-charge operation. This can be done by strobing either cartridge pin 2 (Clock) or cartridge pin 5 (CS) High between them.

For ROM Write and Erase operations, the flash IC's Write pin is tied to Cartridge Pin 31 (VIN/AUDIO), together with a pull-up resistor. Operations are performed by writing a sequence of data and addresses specific to the IC being used.

Currently, only Write, Full Chip Erase and Read/Reset operations are implemented. They can be found in the following files:

Firmware/Writeboi_ino/04MBC_ROM_SST.ino

Firmware/Writeboi_ino/05MBC_ROM_M29.ino

In addition, when using the 4MB stackable ROM option, the second flash IC is only active when ROM Address line 21 is High. The standard way to perform this is to add 0x200000 to the address of any commands you intend to perform, to offset them to the start of bank 128.

The current implementation of this procedure can be found in:

Firmware/Writeboi_ino/08MBC_ROM_M29_4MB.ino

For further details on the interfaces used, and other operations available, please check the respective components' datasheets.

Writeboi Serial API

Writeboi implements a basic API for communications with the host computer, and can be easily integrated with your own applications.

The serial link runs at 115200 Baud, and uses the common port for Arduino devices (Often COM3 on Windows, or /dev/ttyUSB0 on Linux).

Commands are performed by sending a newline terminated UTF-8 string via serial, always consisting of three comma-separated decimal integers.
The Writeboi responds with either a UTF-8 string representing a single hexadecimal byte, newline-terminated, or, in the case of chunk read/write operations, a 1KB raw binary data stream.

Examples:

command,address,data – command fields
Unsigned Int, Unsigned Long, Byte – field size

11,0,0 – Read address 0 of DIP IC
27,100,255 – Write FF (255) to address 100 of RAM, on MBC5
59,0,0 – Erase an M29 ROM, on MBC1
80,2048,0 – Read 1KB from ROM on MBC1, starting at address 2048
92,0,0 – Write 1KB to SST ROM on MBC5, starting at address 0

A full list of operations can be found in the file:

Firmware/Writeboi_ino/10Communications.ino

Do note that specific versions of most functions exist for each MBC, and using the wrong one will likely result in an error.

Command Responses:

Single-Byte Read: Returns the requested byte.

Single-Byte Write: Returns “0” if successful, “1” if failed.

Chip Erase: Returns “2” on 1 second intervals while erasing, then “0” if successful. Return “1” if erasing fails.

Chunk Read: Returns a 1KB chunk of raw binary data, then returns “0” to indicate the process has terminated.

Chunk Write: Expects a 1KB chunk of raw binary data to be sent immediately after the command. When done writing, returns “0” if the operation succeeds, or “1” if it fails.

Dump Cartridge Header: Returns a 1KB chunk of raw binary data, corresponding to memory bank 0, then returns “0” to indicate the process has terminated. This function is special in that it can be used on both MBC1 and MBC5, to detect the type of cartridge being used.

APPENDIXES

I - CPLD Development Board

Being a PLCC-44 packaged IC, the ATF1502ASL is slightly awkward to use when prototyping with breadboards. The solution to this issue was to develop a basic breakout board with pin headers at the initial stage of the project, until the layout of the mainboard was finalized.

Even now, this board can be useful, both for general prototyping, as well as for programming used or JTAG-disabled CPLDs. It is highly recommended that you build one, if you plan to use a second-hand CPLD for your Squareboi, or if you're having issues uploading firmware to it.

To Program A Used CPLD:

->Follow the same procedure as when uploading firmware to the Squareboi itself, with two differences:

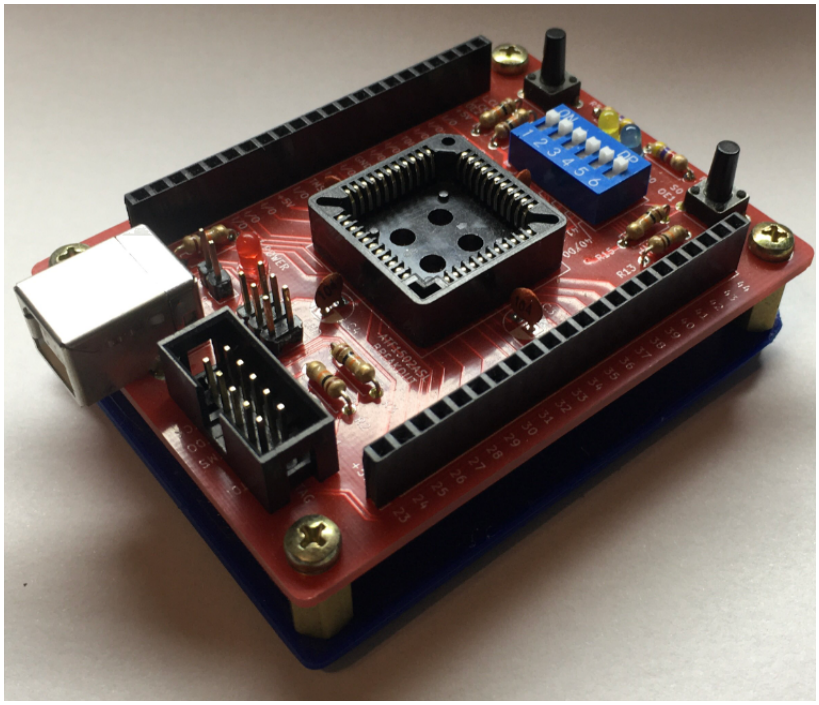
1- Make sure the 6 DIP switches are turned to the ON position, to keep all mandatory inputs in a stable state

2- When powering the board, it is possible to just plug it into an USB port on the computer being used, as it does not block serial communications like the Arduino would

To Program A JTAG-disabled CPLD:

->Follow the same procedure as programming a used CPLD, but, after the board has been powered, apply +12v to pin 44 (OE1) while programming. You could either use a benchtop power supply, or a small boost converter powered off the 5V header.
It might take several tries for programming to work using this method, but it should in theory let you program a chip that's had its JTAG header disabled.

CPLD Development Board



Gerber File Location:
Hardware/KiCad 6/ATF Devboard/ATF_Dev_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

USB Socket: USB B, 90 degree angle socket, through-hole version

Resistors: 20KOhm 1/4W through-hole resistor, 10 units

Capacitors: 100nF ceramic disc capacitor, 2.5mm pin spacing, 4 units

LEDs: Any standard LED with a 2.5mm pin pitch, 3 units, ideally of different colors

Momentary Switches: Any standard 6mm tactile switch, 2 units

Pullup/Pulldown switch: 6-position DIP switch

Programming socket: 10-pin JTAG socket, aka 10-pin shrouded box header. Can be replaced by a 2x5 pin header if necessary.

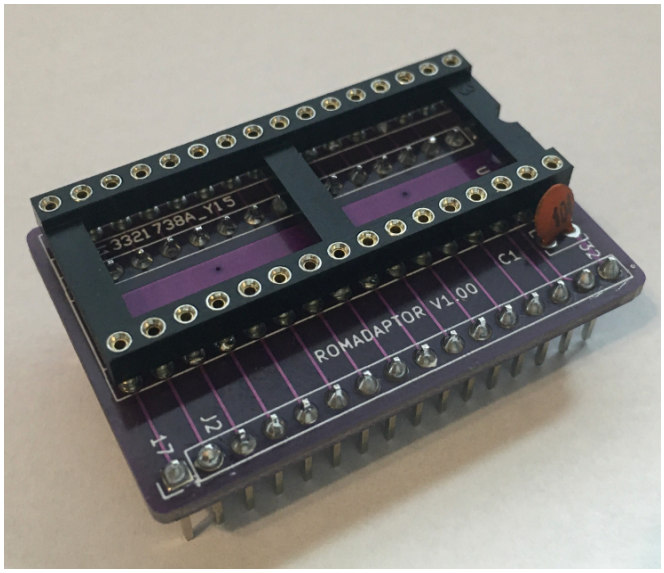
Pin headers:
1x22 2.54mm female pin header, 2 units
and
2x3 2.54mm pin header
and
1x2 2.54mm pin header

If desired, there is also an optional 3D printed backplate that can make the board more convenient to hold.
file: Case/Devboard_Backplate.amf

8 or 12mm M3 screw, 8 units

11mm M3 female-to-female threaded standoff, 4 units

II - ROM Adaptor Board



This board allows a DIP ROM IC to be connected to a Squareboi that has the SMD header instead.
This is a use case that rarely comes up in normal operation, but can be useful when testing a new board if you're having issues with soldering the SMD ICs.

Gerber File Location:
Hardware/KiCad 6/ROMAdaptor/ROMAdaptor_Gerbs.zip

Board printing options: two-layer, standard surface finish, 1.6mm board thickness

Bill Of Materials:

Socket: DIP-32 socket, turned pin, 15.24mm width

Pin Header: 1x16 2.54mm male header, 2 units

Capacitor: 100nF ceramic disc capacitor, 2.5mm pin spacing

Notes:

The footprints for the headers and socket are on top of one another. This means that the pin header must be soldered before the socket, or it will not be possible to access it.

III - Game Compatibility

What follows is a list of games that have been tested with the system. It is currently incomplete, as I am limited to the games I could legally obtain for testing.

Name	ROM Name	MBC	MBC used	Size	RAM needed?	Compatibility	Notes
Tetris	TETRIS	None	MBC1	32KB	No	Full	
Donkey Kong Land	DONKEYKONGLAND	MBC1	MBC1	512KB	Yes	Full	
Pokemon Red	POKEMON RED	MBC3	MBC5	1MB	Yes	Full	
Pokemon Blue	POKEMON BLUE	MBC3	MBC5	1MB	Yes	Full	
Pokemon Yellow	POKEMON YELLOW	MBC5	MBC5	1MB	Yes	Full	
Pokemon Silver	POKEMON_SLV	MBC3	MBC5	2MB	Yes	Partial	No RTC. Game clock may not run, or report that time has been lost on every boot-up.
Pokemon Pinball	POKEPINBALL	MBC5	MBC5	1MB	Yes	Partial	No rumble.
Rayman	RAYMAN	MBC5	MBC5	4MB	No	Full	
Wario Land 3	WARIOLAND3	MBC5	MBC5	2MB	Yes	Full	
Densha de Go! 2	DENSYADEGO2	MBC5	MBC5	8MB	Yes	No	Not enough storage space.