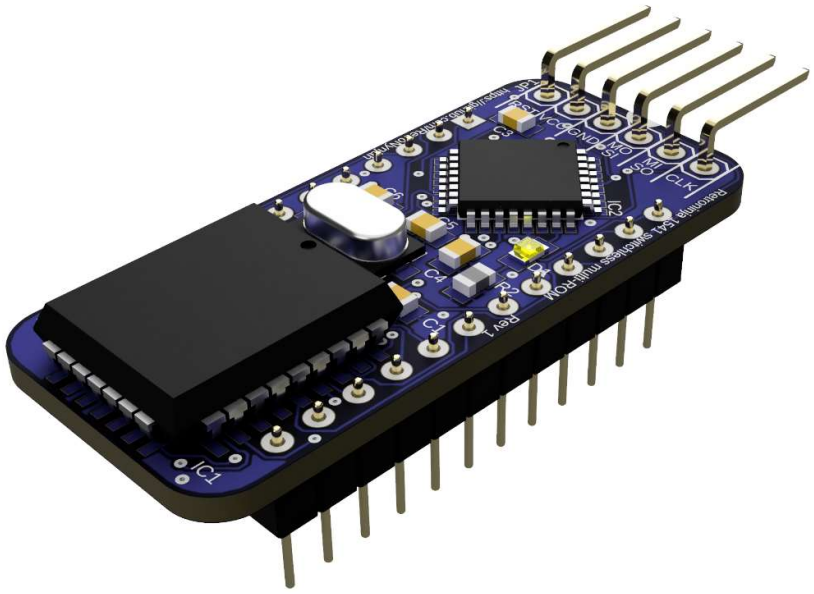
Switchless Multi-ROM for CBM 1541 User's Guide



This microcontroller-based switchless ROM-switcher for the 1541 disk drive lets you switch drive ROM between stock CBM DOS and JiffyDOS using simple basic commands.
It's possible to use other DOS versions too as long as only the upper half of the ROM set ($E000) needs switching as with JiffyDOS.

Usage

The switcher is listening to the data bus for a predefined word and a number. The easiest way to send the word to the drive is to issue a load command to the drive from basic with the word as filename.
The default magic word is #@RNROM where # is the number of the ROM image you want to switch to.
LOAD "1@RNROM",8 switches drive 8 to DOS image 1 and **LOAD "1@RNROM",8** switches to image 2 and so on. The default valid range of numbers is 1-8. It is possible to modify the code to allow bigger numbers but the amount of available ROMs on the market is not that large.

The drive will respond with a file not found error but the programmer will pick up the command, blink the multi-ROM LED to indicate which image was selected, switch the ROM by setting the high address pins accordingly and reset the drive using the newly selected ROM.

The selected ROM will be remembered and used at power-on.

Theory of operation

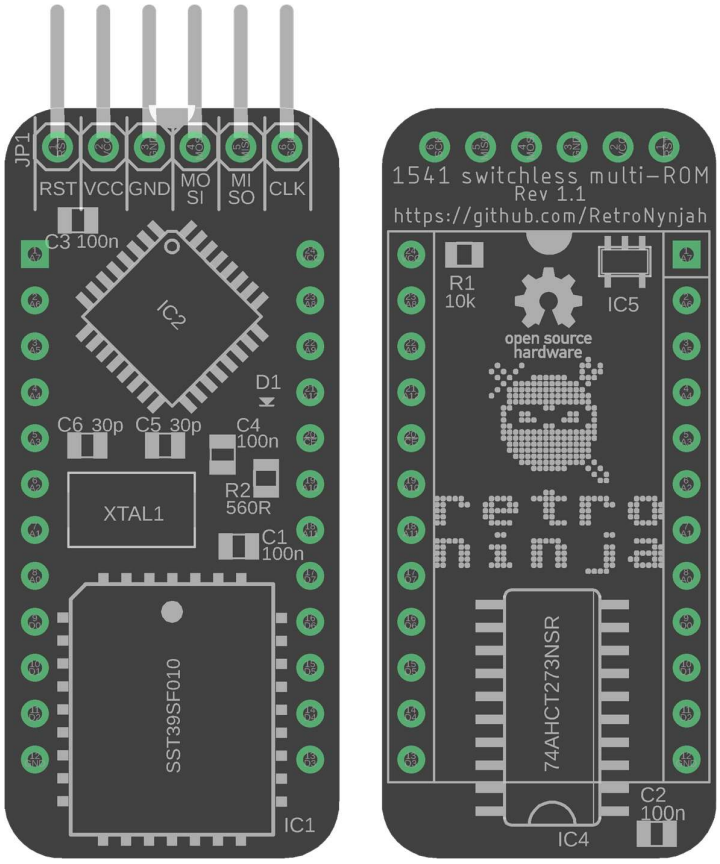
At power on, the microcontroller in the ROM switcher reads address 0 from its internal EPROM to find out which ROM it should select, sets the address A13-A16 on the flash chip and resets the drive to make sure it's booted with the correct image. This can be noticed when the drive is powered on and the drive initializes and stops spinning and then spins up again briefly during the switch reset. It then starts listening for a switch command.

Even at a clock speed of 16 MHz, reading and analyzing a 1MHz data bus can be a challenge so the data on the 1541 data bus is buffered in the switcher by a 74AHCT273 flip-flop that is clocked by the Write signal on the 6502 MPU in the 1541. By clocking the data with the R/W signal we filter out irrelevant bus traffic and the flip-flop holds the data until the next write cycle. This gives the microcontroller enough time to read the data from the flip-flop.
The microcontroller listens for a predefined sequence of bytes *in reverse order* (MORNR@) and once the sequence is found, it treats the following numeric character as the target ROM image number. The reverse order of the characters on the bus is why the image number must be specified at the beginning of the command.
The microcontroller sets address pins A13-A16 on the flash chip according to image selection and it stores the ROM number at address 0 of its internal EPROM for use at next power-on. It then resets the drive by toggling the IEC reset line.

Building the circuit

The switcher is small and a bit difficult to solder but it can be hand soldered. The reason for the small size is space constraints in some drives. This design started off as a Multi-ROM for the 1541-II which has even less space around the ROM. You can find that version among my other repositories on GitHub.

The Flash chip is the trickiest part to solder and I recommended reflow soldering the top side using stencil and solder paste. Then solder the bottom side by hand and finally solder the pin headers.



Parts

The switcher is designed for the SST39SF010A flash chip but the bigger SST39SF020/SST39SF040 could be used too if no SST39SF010A is available. Remember to duplicate the flash contents to fill all of flash to prevent floating pins from switching to empty flash areas.

The flip-flop needs to be a 74**AHCT**273. LS/HCT/AHC or any other variant will most likely not work.

The 741G04 single gate inverter is needed for inverting the active low write signal from the 6502 to the active high clock input of the flip-flop.

The crystal should be a 16MHz crystal but 20 MHz might work too if you recompile the sketch and burn the fuses accordingly. Not tested.

The LED on the board is optional but flashes a number of times indicating the number of the selected ROM when switching. It is only there for debugging reasons.

The pin headers that goes into the ROM socket should have round machined pins or be of that long flat type that is used for the pass-through pin headers on Arduino shields.
Square pin headers destroy the ROM socket in the drive.

Programming the microcontroller

The microcontroller on the board is an ATmega328P and the code for the controller was created using the Arduino environment. The device can be programmed in circuit from a sketch in the Arduino IDE using an ISP programmer or using the pre-compiled HEX file with AVRDUDE. The fuse settings should be the same as for an Arduino UNO.
lfuse=0xFF, hfuse=0xDE, efuse=0xFD

It's possible to use an Arduino as ISP or use an ISP programmer such as a USBASP. Those techniques are not covered by this document.

Programming the flash chip

Due to space constraints, there are no on-board programming headers for the flash chip so the flash chip must be programmed in a flash/EPROM programmer before soldering it to the board or desoldered for reprogramming.
It could also be possible to make a programming adapter to put on top of the flash chip to program it while it's still soldered to the board but this is not tested.

The high ROM chip in the 1541 is only 64 kbit (8 kilobyte) while the flash chip is 1 Mbit (128 kilobyte). It's highly recommended to fill the entire flash chip with working DOS images to prevent bricking of the drive by selecting a non-existing image. The switcher will not work without a working ROM in the drive that can receive the serial data and put it on the data bus.
If you just use two ROM images, fill the flash with the same two images over and over for safety.

Installation

The circuit can be installed in the drive without soldering by using DuPont cables with test clips that attaches to the R/W pin on the 6502 MPU and to the reset pin on the back of any of the IEC connectors. However, it is recommended to solder the cables to the pins on the bottom of the drive PCB. That makes for a much more robust and transportable solution.
An even more stable solution is to solder the cables to the edge of the switcher PCB instead of using a pin header but that could make it harder to upgrade the firmware in the future.

The edge of the Multi-ROM PCB with the six ISP pins should face the notched end of the ROM socket (Pin 1). You may have to bend a ceramic capacitor out of the way a little before you can insert it.

The clock signal (CLK on the Multi-ROM) connects to the R/W pin (pin 34) on the 6502 MPU.
MOSI on the Multi-ROM connects to IEC reset. That's the center pin on the back of the IEC connectors. The Multi-ROM works without a connection to IEC reset but a manual drive reset would then be needed after switching ROM.

The sketch and a pre-compiled firmware for the Multi-ROM can be found at
https://github.com/RetroNynjah/1541-Switchless-Multi-ROM

Troubleshooting

If the device becomes bricked due to an incorrect DOS image, program the ATmega with a sketch that sets byte 0 to 0x00 or 0xFF in the internal EPROM. Then upload the original Multi-ROM sketch again and you should be back on track.

Switching may not work if you are using a fast loader such as Action Replay in your computer. Try switching the ROM without the fast loader. Switching ROM using JiffyDOS in the C64 is working fine though.

Possible other uses

A second switcher for the low ROM in the 1541 could be used simultaneously if you need to switch both ROMs simultaneously or maybe use a slave 2364 ROM adapter for the low ROM and connect the unused address A13-A16 to the same address pins on the Switchless Multi-ROM using bodge wire or similar.

The switcher could possibly be used for other 2364 ROM switching applications if you can find the right clock signal and strings to listen for.

The switcher could work in other 1541 compatible drives that are using 2364 ROMs but none have been tested so far. The 1541-II switcher emulates a 16kB 27128 EPROM has been tested in a number of compatible drives.