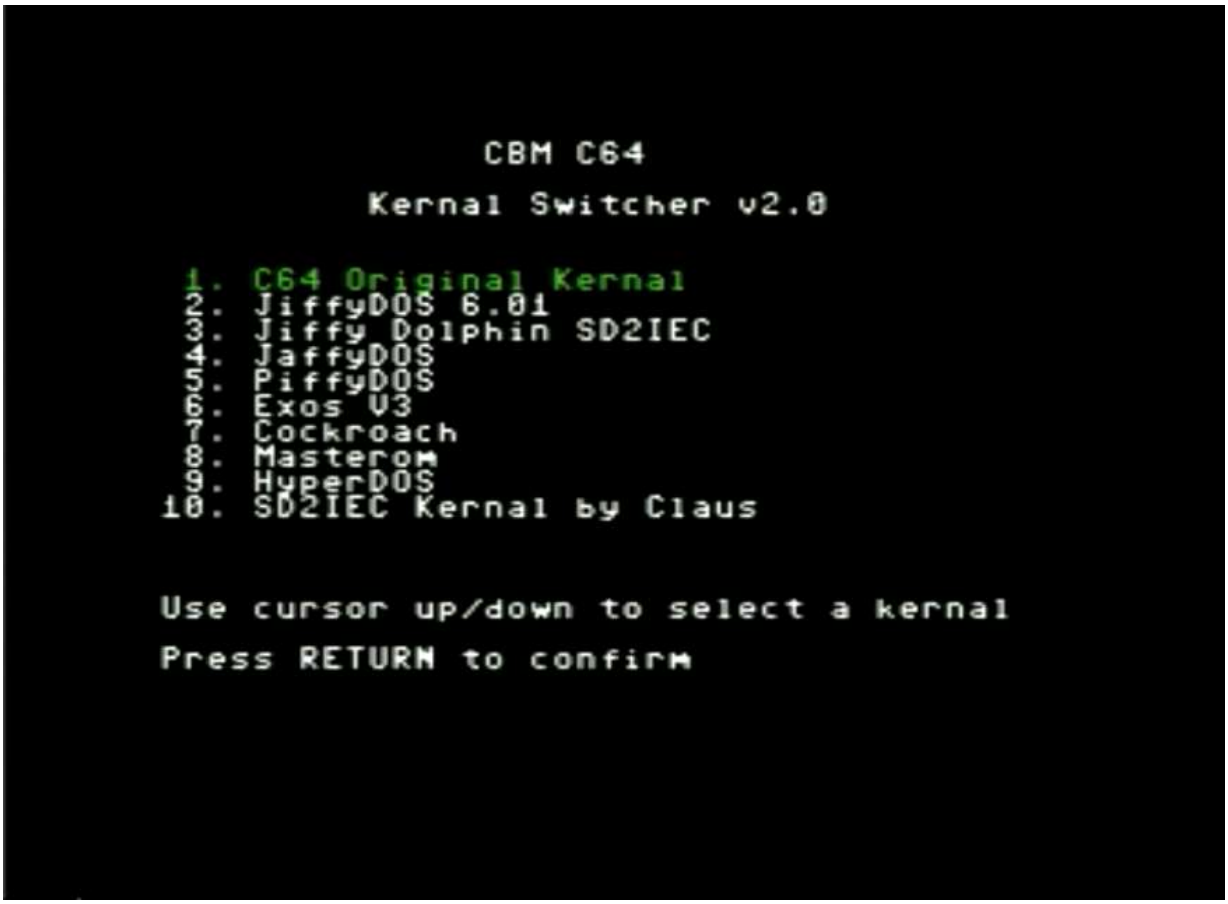
CBM Switchless Multi ROM
for 2364 & 27128 & 27256 ROMS



Application Info

Table of Contents

Basics

Logic levels

Digital electronics rely on binary logic to store, process, and transmit data or information. Binary Logic refers to one of two states -- ON or OFF. This is commonly translated as a binary 1 or binary 0. A binary 1 is also referred to as a HIGH signal and a binary 0 is referred to as a LOW signal.

The strength of a signal is typically described by its voltage level.
How is a logic 0 (V_{IL}) or a logic 1 (V_{IH}) defined?

5V TTL Logic levels, maximum input LOW voltage (V_{IL}) is 0.8 V,
minimum input HIGH voltage (V_{IH}) is 2 V.

AVR MCU Logic levels, maximum input LOW voltage (V_{IL}) is 1.5 V,
minimum input HIGH voltage (V_{IH}) is 3 V.

Active-Low and Active-High

When working with ICs and microcontrollers, you'll likely encounter pins that are active-low and pins that are active-high. Simply put, this just describes how the pin is activated. If it's an active-low pin, you must "pull" that pin LOW by connecting it to GND (ground). For an active high pin, you connect it to your HIGH voltage.

For example, let's say you have a flash-memory chip that has an output enable pin, OE. If you see the OE pin anywhere in the datasheet with a line over it like this, $\overline{\mathrm{OE}}$, then that pin is active-low. The OE pin would need to be pulled LOW in order to enable the output. If, however, the OE pin doesn't have a line over it, then it is active high, and it needs to be pulled HIGH in order to enable the output.

Bit shift

A bit shift moves each digit in a number's binary representation left or right.
The left shift operator is written as "<<", the right shift operator is written as ">>".
When shifting left, the most-significant bit is lost, and a 0 bit is inserted on the other end. When shifting right, the least-significant bit is lost and a 0 bit is inserted on the other end. A single left shift multiplies a number by 2, a single right shift divides a number by 2. Bit shift is faster than multiplying and dividing.

Example:

Binary	0b0010 << 1 → 0b0100;	0b1010 >> 1 → 0b0101
Decimal	2 << 1 → 4;	10 >> 1 → 5

AVR MCU Firmware

Arduino IDE

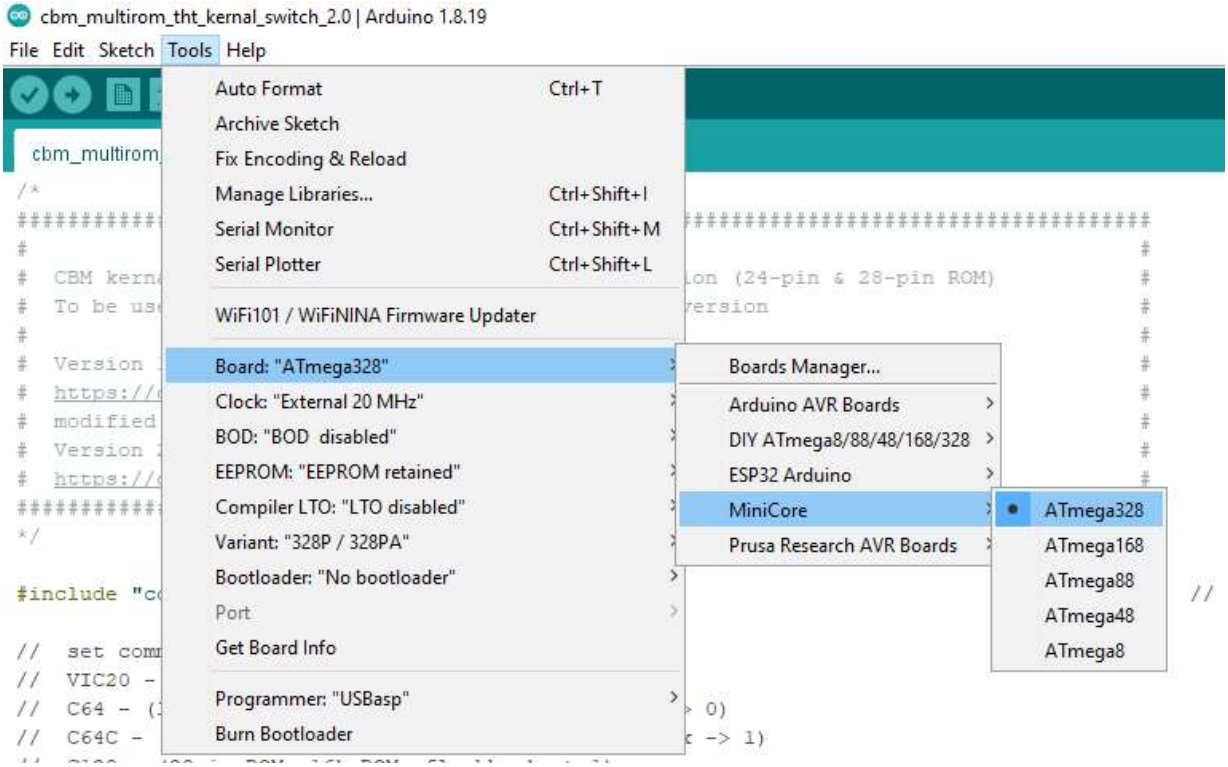
Requirements can be found in the General-Info.pdf.

If you are new to Arduino programming, please see at:
https://docs.arduino.cc/learn/starting-guide/getting-started-arduino

Please open the Arduino sketch from the /applications/arduino/firmware/v2.0/source folder with the Arduino IDE.

Then please adjust the AVR MCU you use.

Tools → Board → MiniCore → AVR MCU (depends on what you use)



Necessary settings:

Clock → External 20 MHz or 16 MHz, depends on what you use.
Compiler LTO → LTO disabled.
Variant → depends on what you use.
Bootloader → No bootloader.

If you use ISP programming you still have to set up your Programmer.

All other settings can remain default.

AVR MCU Firmware

Defining the Sketch

The sketch contains many comments and is self-explanatory.
Please define the CBM Computer for which the firmware should be.

```
//  set commodore cbm id, possible ids
//  VIC20 - (24pin ROM, 8k ROM, flashbank -> 0)
//  C64 - (longboard, 24pin ROM, 8k ROM, flashbank -> 0)
//  C64C - (shortboard, 28pin ROM, 16k ROM, flashbank -> 1)
//  C128 - (28pin ROM, 16k ROM, flashbank -> 1)
//  C128DCR - (28pin ROM, 32k ROM, flashbank -> 2)

#define CBMID C64C                                                    // set CBM ID
```

If you don't want the LED to blink, you can disable the LED by commenting it out.

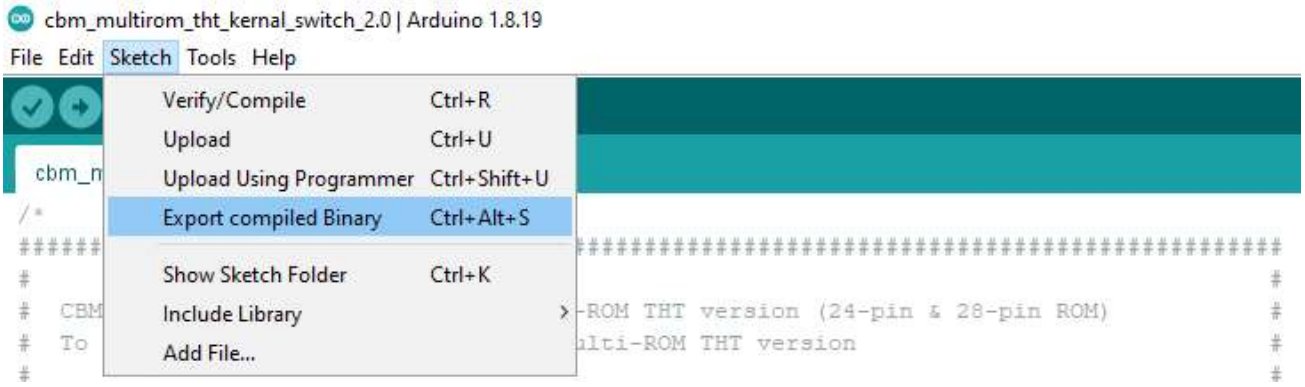
```
// enable LED for blinking
#define LED                                                  // enable LED, commend out to disable

// nothing to setup below
```

These are all settings, nothing more needs to be done.

Compiling

Now please compile the firmware and load it onto your MCU.

If you are programming via ISP, click Sketch → Upload Using Programmer.
For an Universal Programmer or AVRDUDESS click Sketch → Export compiled Binary.



Mini Kernal

CBM prg Studio

Requirements can be found in the General-Info.pdf.

CBM prg Studio tutorials can be found here:
https://oldskoolcoder.co.uk/cbm-prg-studio-tutorials/

Each CBM Computer has its own mini Kernal source file.
Folder /applications/cbm-prog-studio/mini-kernal/v2.0/……/source.

Please open the project file with the ending ".cbmprj" of the desired CBM Computer using the CBM prg Studio program.

Now you can design your personal Switchless mini Kernal menu in the main.asm file.

Such as the number of Kernal ROM image entries in the Switchless mini Kernal menu.

main.asm | vic.asm

```
1  ; ========================================
2  ; Project     C64 Kernal menu
3  ; Target      Commodore 64
4  ; Comments    modified by xboxpro1
5  ; Author      Retroninja
6  ; ========================================
7  ; Verison history
8  ; 1.0   First version for C64
9  ; 1.1   Fixed bugs
10 ; 1.2   Improved menu routines and added a sprite
11 ; 1.3   Cleaned up code. Clear screen to avoid garbage during reset
12 ; 1.4   Added command for C128DCR
13 ; 1.5   Scan keys once per screen refresh to debounce keys
14 ; 2.0   Renamed switcher, changed colors, changed ascii command, disabled sprite
15
16 CHARSET   2
17 NOLOADADDR
18 GenerateTo c64minikernal.bin
19
20 KRNIMGS = 10     ; number of kernal images in menu/ROM 1..10 where 1 would be quite pointless :)
21 BODCOL  = $00    ; border color (C64 default = $0e) $00 = black
22 BAKCOL  = $00    ; background color (C64 default = $06) $00 = black
23 CHRC    = $01    ; mnutxt color (C64 default = #$0e ) $01 = white
24 CHRCHL  = $05    ; menu highlightedtext color $05 = green
25
```

Also the frame, background and text colors.

```
15
16 CHARSET   2
17 NOLOADADDR
18 GenerateTo c64minikernal.bin
19
20 KRNIMGS = 10     ; number of kernal images in menu/ROM 1..10 where 1 would be quite pointless :)
21 BODCOL  = $00    ; border color (C64 default = $0e) $00 = black
22 BAKCOL  = $00    ; background color (C64 default = $06) $00 = black
23 CHRC    = $01    ; mnutxt color (C64 default = #$0e ) $01 = white
24 CHRCHL  = $05    ; menu highlightedtext color $05 = green
25
```

Mini Kernal

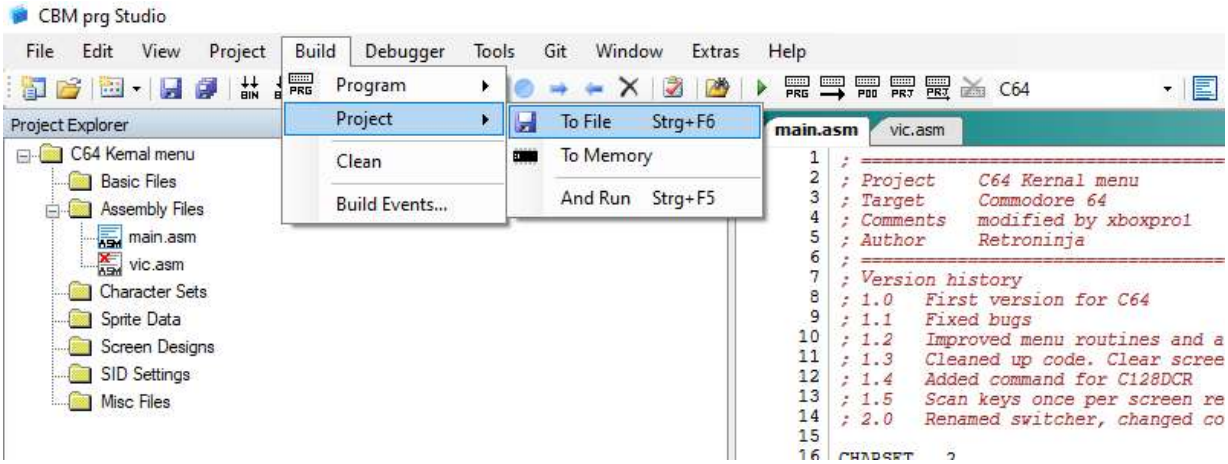
Menu

Design the Kernal ROM entries according to your wishes. The order must match the order in the flash chip memory. The menu Switchless mini Kernel ROM image is always in ROM number 0 and is not entered in the menu.

```
235
236 mnutxt  ; Menu layout
237         ; 6 header rows
238         text '                                        '
239         text '                CBM C64                 '
240         text '                                        '
241         text '          Kernal Switcher v2.0          '
242         text '                                        '
243         text '                                        '
244         ; Up to 10 menu choices. number of shown lines controlled by value in $kernalimages
245         text '  1. C64 Original Kernal                '
246         text '  2. JiffyDOS 6.01                      '
247         text '  3. Jiffy Dolphin SD2IEC               '
248         text '  4. JaffyDOS                           '
249         text '  5. PiffyDOS                           '
250         text '  6. Exos V3                            '
251         text '  7. Cockroach                          '
252         text '  8. Masterom                           '
253         text '  9. HyperDOS                           '
254         text ' 10. SD2IEC Kernal by Claus             '
255         ; 8 footer rows
256         text '                                        '
257         text '                                        '
258         text '                                        '
259         text ' Use cursor up/down to select a kernal  '
260         text '                                        '
261         text ' Press RETURN to confirm                '
262         text '                                        '
263         text '                                        '
264         text '                                        '
```

And you're done, all you have to do is compile the mini Kernal.

Click Build → Project → To File.



Flash-memory Chip

The Excel sheet in the tools folder, macros are harmless but not necessary, show you the memory allocation for the 8k, 16k and 32k ROM modes.
The mode depends on the CBM Mainboard / Kernal ROM chip.

VIC20 and C64 (longboard) normally has an 8k Kernal ROM.
C64C (shortboard) normally has an 16k Kernal ROM. Basic and Kernal in one ROM.
C128 has two 16k ROM chips, one for the C128 mode and one for the C64C mode.
C128DCR has a 32k Kernal ROM.

The memory area is divided into three banks. Bank 0 8k ROM, Bank 1 16k ROM and Bank 2 32k ROM. The memory banks are divided into ROM numbers for each Kernal.

The Switchless mini Kernel ROM image is always in ROM number 0.

For Bank 1 and Bank 2, the kernel is selected in the firmware by bit shifting the ROM number to the left with the bank number.

Bank 0 → VIC20, C64

Kernal	Mini Kernal 8k	CBM Kernal 8k	JiffyDOS 8k	JaffyDOS 8k	Exos V3 8k	HyperDOS 8k
Size	8k	8k	8k	8k	8k	8k
ROM number	0	1	2	3	4	5

Bank 1 → C64C

Kernal	C64 Basic 8k	Mini Kernal 8k	C64 Basic 8k	CBM Kernal 8k	C64 Basic 8k	JiffyDOS 8k
Size	16k		16k		16k	
ROM number	0		2		4	

Bank 1 → C128

Kernal	C128 Mini Kernal 16k	C128 CBM Kernal 16k	C128 JiffyDOS US 16k	C128 JiffyDOS DK 16k
Size	16k	16k	16k	16k
ROM number	0	2	4	6

Bank 2 → C128DCR

Kernal	C64 Basic 8k	C64 Mini-Kernal 8k	C128 Mini-Kernal 16k	C64 Basic 8k	C64 CBM-Kernal 8k	C128 CBM-Kernal 16k
Size	32k			32k		
ROM number	0			4		

For more information, please see RetroNynjah's user guides in the retroninja folder.

Change Source

You can of course change the source codes according to your requirements.
For example, changing the predefined command string of ”magic bytes”.

If you change the predefined command string in the Switchless mini Kernel source code, it also needs to be changed in the AVR MCU source code.

```
225
226
227 ; ascii CBMROM64# command for kernal switcher on data bus
228 cmdasc  byte $43, $42, $4d, $52, $4f, $4d, $36, $34, $23
229 cmdascend
230 ; ascii CBMROM128DCR# command for C128DCR combo switch
231 cmdascdcr
232         byte $43, $42, $4d, $52, $4f, $4d, $31, $32, $38, $44, $43, $52, $23
233 cmdascdcrend
234
```

```
// C64 longboard config
#if CBMID == C64
  byte searchString[] = {0x43,0x42,0x4D,0x52,0x4F,0x4D,0x36,0x34,0x23};   // searchstring is "CBMROM64#"
  #define flashBank 0                                                     // 8k ROM
  #define maxROM  10                                                      // max minikernal ROM entries
#endif
```

There are two PDF files in the tools folder with the ASCII and color codes.
The codes are decimal and they must be converted to hexadecimal.

Programming

Flash memory

To program the flash memory chip you need a USB universal programmer such as the TL866II Plus. A PLCC32-DIP32 adapter is also required.

Tutorial: https://projectiot123.com/2023/05/16/how-to-use-the-tl866ii-plus-programmer/

The easiest way is to first create a flash file with all the Kernal ROM images you want.
Copy all the Kernal ROM images into a folder including a basic ROM.
For example, for an 8K ROM flash file, use the Windows command:
copy /b c64minikernal.bin+c64kernal.bin+jiffydos.bin+exos3.bin+sd2iec.bin flash.bin
For example, for an 16K ROM flash file, use the Windows command:
copy /b basic.bin+minikernal.bin+basic.bin+c64kernal.bin+basic.bin+exos3.bin flash.bin
For example, for an 32K ROM flash file, use the Windows command:
copy /b c64basic.bin+c64minikernal.bin+c128minikernal.bin+
c64basic.bin+c64kernal.bin+c128kernal.bin flash.bin

This is just an example, of course all of your desired Kernal images must be attached.
Use the Universal programmer software to write the flash image to the flash chip.

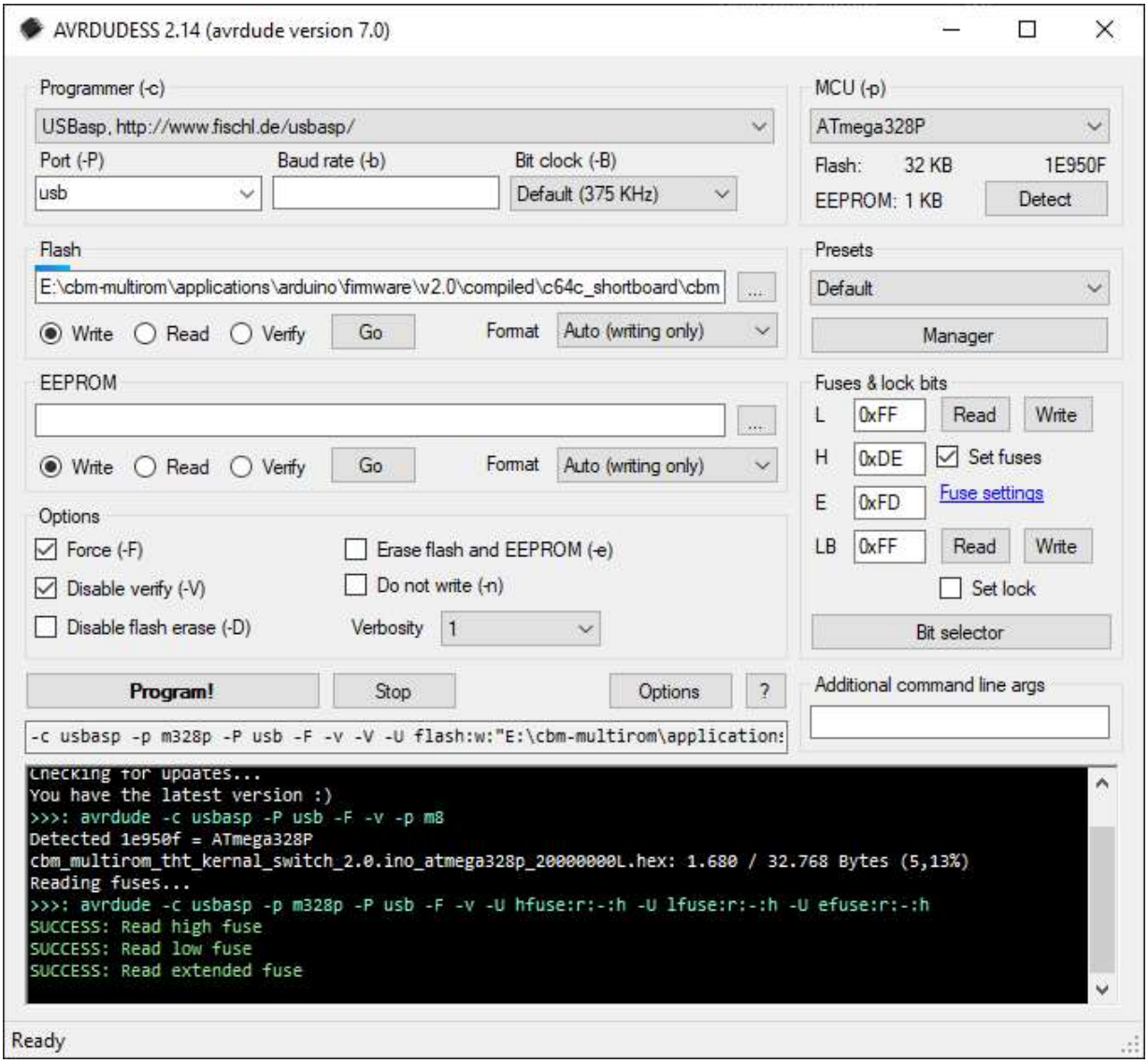
Programming

AVR MCU

You can also flash the AVR MCU with the USB universal programmer.
Or you can use an ISP programmer to flash the MCU.
If you use an ISP programmer, I recommend the program AVRDUDESS.

Requirements can be found in the General-Info.pdf.

With AVRDUDESS you can easily read and write the MCU fuses. There is also a fuse Bit selector and a link to a website where you can get the correct settings for the fuses.



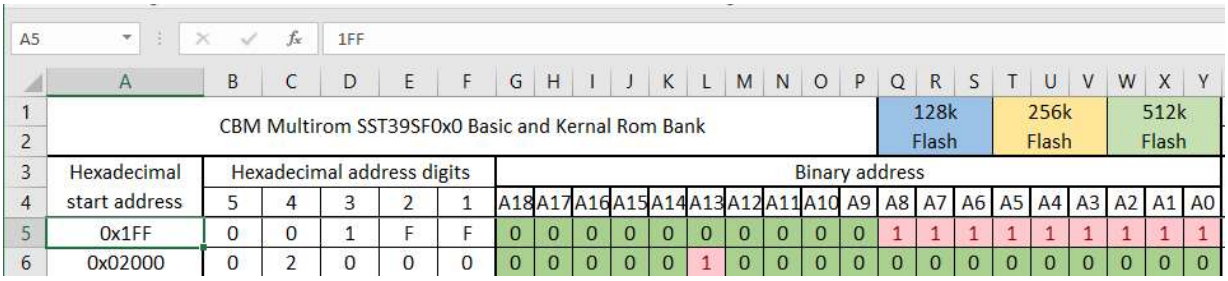
Recommended fuse settings are in the fuses.txt file in the tools folder.

Tools

Excel sheet

In the Excel sheet addresscalc.xlsm in the tools folder you can enter any addresses from 0 to 7FFFF at cell A5.
Please do not write 0x at the beginning, Excel will do that itself.

This causes the address bits A0 to A18 to be set according to the respective address.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	CBM Multirom SST39SF0x0 Basic and Kernal Rom Bank																128k Flash			256k Flash			512k Flash		
2																									
3	Hexadecimal	Hexadecimal address digits					Binary address																		
4	start address	5	4	3	2	1	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
5	0x1FF	0	0	1	F	F	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
6	0x02000	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

The Excel sheet has a sheet protection so that no formula can be deleted accidentally.
The code to remove the sheet protection is 1234.

CircuitVerse

On CircuitVerse you can create simulations with digital components.
https://circuitverse.org/simulator
You can use the MultiromOE.cv file in the tools folder to simulate IC5 the single “AND” Gate for experimental testing.

To use it click: Project → Import file