LEARN

Flashing to memory using a Flashcart

With 8BitCADE

Compatible with:

8BitCADE XL

Contents

Written by the 8BitCADE Team

Support@8bitcade.com

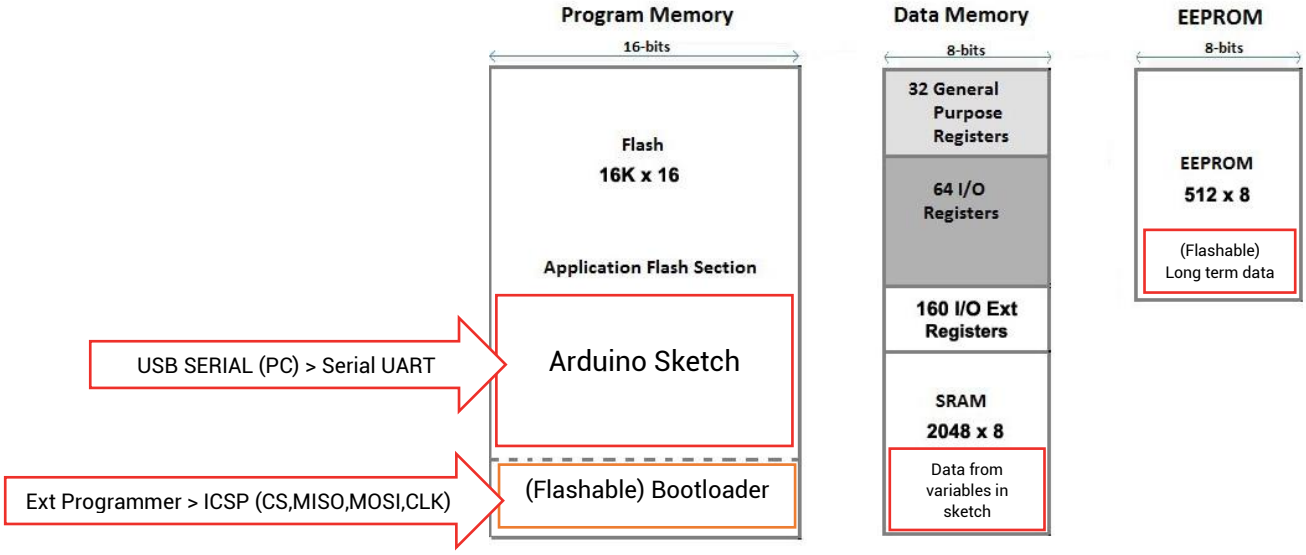
Version 1

ATmega 32U4 has three types of memory

The chip contains 32 kilobytes of internal flash memory, 1 kilobyte of EEPROM and 2.5 kilobytes of SRAM. The flash and EEPROM memory are rewritable electronically, using the Arduino IDE. The information is written to the memory and remains (called volatile) after the chip is powered down (switched off) but the SRAM is a memory which only saves information while power is supplied (called non-volatile) and when the power removed all the information is erased. The program memory contains your sketch you loaded via the Arduino IDE and the bootloader, which is a short, protected program that runs when you turn the chip on, or press the reset button. Its main function is to wait for the Arduino software on your computer to send it a new program which it then writes to memory. The bootloader is an important sketch and is positioned at the end of Program Memory to protect it and can only be rewritten using ICSP (In Circuit Serial Programming).

When we refer to "boot loading" the **ATmega 32U4** chip, we are talking about using a special device (called an In-Circuit-Serial-Programmer or ICSP) to replace the bootloader software.

Overview of the Different Types of Memory of an AVR Chip



Program Memory - 32KB nonvolatile memory. This is used for storing your **sketch** (program) and the bootloader. **This code remains in the memory of the chip even if the power is removed from the circuit.** The program or actual sketch code is loaded via an FTDI chip (a UART Bridge) converting your PC USB serial to TTL serial the AVR chip can read.

Boot Flash Section (Bootloader) enables you to program the Arduino board easily using a PC attached via a USB connector. Its protected and is only accessible via special pins (SPI) if you want to reprogram it using ICSP.

SRAM Memory -2.5 KB volatile memory. This is used for storing data from your sketch variables while the program is running. Once the power is turned off this data is lost. A good example of the type of data might be the position of a player on the screen which needs to be tracked so the position can be updated, like the ball in the game 'Pong'.

EEPROM Memory -1KB nonvolatile memory. This can be used to store data that must be available even after the board is powered down and then powered up again. A good example of data stored here would be high scores of a game, small amount of data which would need to be saved to be displayed for later use.

Compile and upload single games to 8BitCADE using Arduino IDE

Games for Arduboy can be downloaded from the following sources:

- https://community.arduboy.com/c/games
- https://github.com/topics/arduboy-game
- Erwin: Arduboy collections
- http://arduboy.ried.cl/

You can download the source code of the game that you can load to Arduino and upload to the 8BitCADE. See the video on how this is done:

https://8bitcade.com/game/8bitcadexl/

Compile and upload single games to 8BitCADE in hex file format

See the video on how this is done: https://8bitcade.com/game/8bitcadexl/

Hex file is a text file containing binary codes resulting from the compilation of your Arduino program (sketch), but represented in a text file format using two digit hexadecimal numbers 0-9, A-F.

You can get these hex file in different ways.

1. We can download hex files from the different sources we explained above:
2. Alternatively, you can make your own hex file.

Open a game in the Arduino IDE, go to>Sketch >Export Compiled Binary. Your sketch will be compiled, then a copy of the compiled .hex file will be output to the directory of your sketch. Browse on your PC to the sketch folder or in the IDE select Sketch>Show Sketch to view the code.

Folder to see the hex file.

If you installed MR.Blinky homemade package, two versions of the .hex file will be created.

For example, if you compile the picovaders.ino sketch, the following two .hex files will be created.

1. picovaders.ino-arduboy-promicro-ssd1306.hex
2. picovaders.ino with_bootloader-arduboy-promicro-ssd1306.hex **(this one is not needed and can be deleted)**

Arduboy Hex File Uploader to program 8BitCADE with Hex Files

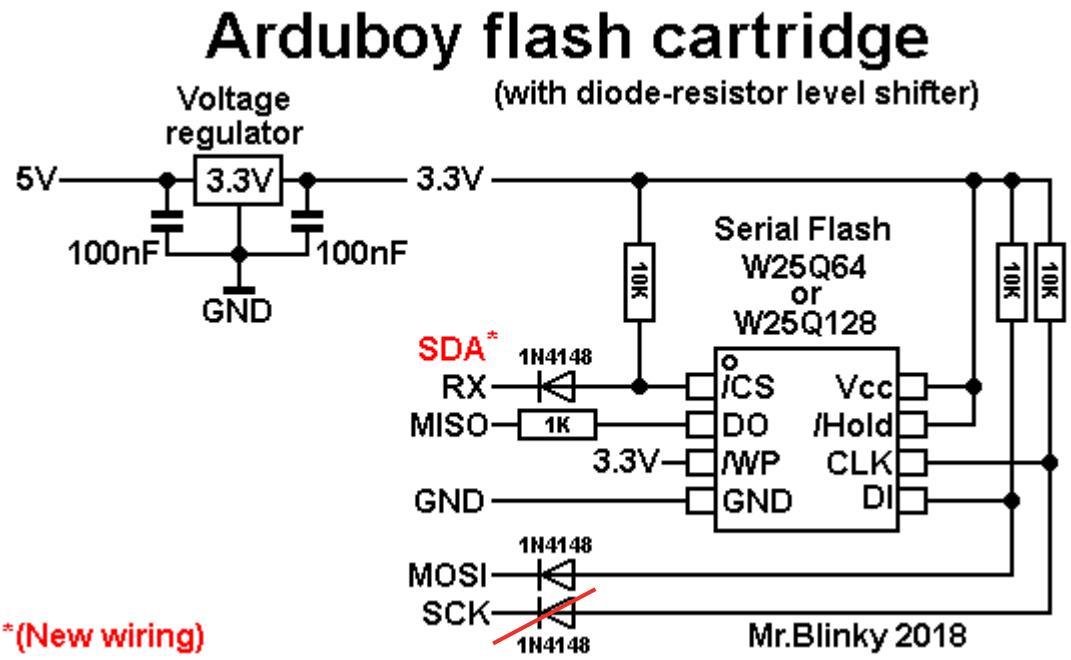
We will only use the first file: picovaders.ino-arduboy-promicro-ssd1306.hex to upload hex file to 8BitCADE. To do this you need to use an uploader. There are many on the internet. For Example, MR.Blinkys uploader is very simple to use. Browse to https://github.com/MrBlinky/Arduboy-Python-Utilities and follow the instructions there to install MR.Blinkys Arduboy Python utilities. If you do not have a python installed, you need to follow the instruction to install python and the required python modules first.

1. Connect the Arduboy to the USB port of your computer. Switch on the 8BitCADE.
2. Start the shell program in your operating system e.g. the terminal app in Mac OSX or the command prompt in windows to type the following commands to upload the hex file to the 8BitCADE.

3. Taking our previous sketch picovaders.ino as an example. python uploader.py picovaders.ino-arduboy-promicro-ssd1306.hex
4. Once the game is uploaded, the Arduboy will reset and start the game.

Write games to serial flash Chip

Hardware Schematic



1N4148 Diode not required on SCK line

Writing Games to Serial Flash Chip

See the video on how this is done: https://8bitcade.com/game/8bitcadexl/

To write the consolidated game file to serial flash, you need to use MR.Blinky Arduboy Python Utilities again. You should have this already installed if you follow the previous step. Otherwise, browse to https://github.com/MrBlinky/Arduboy-Python-Utilities and follow the instructions there to install MR.Blinky Arduboy Python Utilities.

1. Follow the instructions to install python and the required python modules first.
2. Create the index file for the consolidated game image file to hold as many as 500 games.

We will use the **'Use the flashcart-builder.py'** script to build a consolidated game image files for all the games you want to store into the serial flash of the Arduboy. A 16MB Serial flash can hold as many as 500 games. This script builds a binary flash image from an index file (.csv) and the following 2 files for each game:

A hex files that is the text file containing the hexadecimal codes of the binary images of the compiled Arduboy games. Refer to the **'example-flashcart\flashcart-index.csv** file' for example syntax. This file is included in the package if you click Clone or Download.

b.png graphical image files to be displayed on the bootloader menu on Arduboy so you know which game you will be selecting. Some of the game repositories will have this graphical file together with their game source file or hex file. If you cannot find it, you can create your own by either running the game on the Arduboy emulator (https://felipemanga.github.io/ProjectABE/) and capture the screen. Or just type the name of the game on a power point, then screen capture that. Then the graphics you captured/created down to 128x64 pixel using paint brush in windows, or preview in Mac OSX. The YouTube video also explains how to put things the right place of this.csv index file.

One thing to note, the examples.csv file from MR.Blinkys GitHub is used in Windows PC, backslash\ are used in the pathnames. If you are using a Linux system or MAC OSX you need to change it to /.

To get a quick start, you can download my package of 63 games from:

https://github.com/cheungbx/ArduBaby63games.zip

This package contains the hex files and .png files of the 63 games I have chosen, plus the games.csv index file and the **games-image.bin file that is built using the flashcard-builder.py script.** You can add more games into games.csv and build your own consolidated game binary image file to be written to the serial flash.

You can put max 500 games on the 16M serial flash. I will explain how to make the.csv file using the games.csv that you can download form my GitHub. Even though the.csv file can be opened using excel. DO NOT use excel to open the file. It will corrupt the file. Please use a plain text editor only. You can use notepad in windows.

The first line of the.csv file is the header you can ignore. List;Discription;Title screen;Hex file

The second line point to the graphical image file (must be 128x64 pixel in png file format) for the boot loader menu screen. Bootloader is named in example-flashcarts> arduboy_loader.png.

The games are configured **starting from the third line**. Games are organised into **groups** in the bootloader menu called **categories**. This line is the group title of the list of games for that group e.g. Action Game. It also points to the graphical image file for the group of games.

The beginning denotes group number 1. All games that follows this group will start with this number.

1;Action Games;category-screens\Action.png. **Then you add one line for each game within that group.**

Starting with group number 1, **name of the game**, and the **path of the graphic file** for a snapshot of the screen, and the **path of the hex file**. All separated by **“ ”**Add one more “ ” to skip the parameter for the save file.

Example flash cart

List;Discription;Title screen;Hex file;Data file;Save file

0;Bootloader;arduboy_loader.png;;;

1;Action Games;category-screens\Action.png;;;

1;SanSan;Action\Sansan.png;Action\Sansan.hex;;

2;Arcade Games;category-screens\Arcade.png;;;

2;1943;Arcade\Nineteen43.png;Arcade\Nineteen43.hex;;

2;Ardu-Whack;Arcade\Ardu-Whack.png;Arcade\Ardu-Whack.hex;;

3;Platformer Games;category-screens\Platformer.png;;;

3;CastleBoy;Platformer\CastleBoy.png;Platformer\CastleBoy.hex;;

4;Puzzle Games;category-screens\Puzzle.png;;;

4;Hangman;Puzzle\Hangman.png;Puzzle\Hangman.hex;;

4;LATE;Puzzle\LATE.png;Puzzle\LATE.hex;;

4;Minesweeper;Puzzle\Minesweeper.png;Puzzle\Minesweeper.hex;;

5;Racing Games;category-screens\Racing.png;;;

5;Ard-Drivin;Racing\Ard-Drivin.png;Racing\Ard-Drivin.hex;;

6;RPG Games;category-screens\RPG.png;;;

6;Rick & Morty;RPG\Rick-and-Morty.png;RPG\Rick-and-Morty.hex;;

7;Shooter Games;category-screens\Shooter.png;;;

7;Night Raid;Shooter\Night-Raid.png;Shooter\Night-Raid.hex;;The last line has a save file in the parameter which is a cartoon movie.

To build the consolidated game image file, type the command, where games.csv is your game index file. python flashcart-builder.py games.csv. This will create a file named **games-image.bin.** Write the consolidated game image file to 8BitCADE. **We use MR.Blinkys flashcart-writer.py script to write the consolidated game image file to the serial flash memory of the 8BitCADE.**

If you are using my sample games-image.bin file you can type this command. python flashcart-writer.py games-image.bin

If you are using an **SSD1309 OLED** screen instead of the SSD1306 OLED on the standard build, you can patch the screen driver on the fly. To automatically apply the SSD1309 patch to the uploaded image, make a copy of flashcart-writer.py and rename it to flashcart-writer-1309.py. Then type python flashcart-writer-1309.py games-image.bin

Play games from serial flash

To play games from serial flash, switch on the 8BitCADE. If you already have a game loaded, the game will start automatically. Press the reset button on the top of the 8BitCADE once to go to the bootloader menu.

The bootloader menu will be displayed. The RGB LED will light up in sequence. If you see an icon that looks like a USB port displayed instead, that means your serial flash memory chip is not working. Pls check the wiring.

If you do not press any keys within 12 seconds, the game already stored in the ATmega32U4's internal flash memory will be run. To go back from a game to the bootloader menu, just press the Reset button once. You can press the left or right button to scroll through the different category (group) of games. Press the down or up button to scroll through the games within a category (group). Press B button to copy the game from the serial flash memory onto the ATmega32U4s internal Flash memory. The game will start within a second.

Now you have a tiny game console that you can play on the road. I challenge you to collect and load up your 16M Serial flash with 500 games. I haven't seen anyone who's done that yet to fill up the serial flash. If you can do that, do share that consolidated game file with us.

MR.Blinkys GitHub link for python utilities for game upload and serial flash memory operations.

https://github.com/MrBlinky/Arduboy-Python-Utilities

Extract from Blinky Python

Installing dependencies

- Download and install python 2.7.x from https://www.python.org/downloads/ if it is not already installed
- Make sure the option 'Add python.exe to path' is checked on install options (Windows)
- After install run 'python -m pip install pyserial' from command line. For OSX run 'easy_install pyserial' from terminal.

Note: Not all utilities work with Python 3.7.x yet.

Uploader

- Works with both Python 2.7.x **AND** 3.7.x
- Requires pySerial: python -m pip install pyserial

.Hex file and .Arduboy uploader for Arduboy

- Double click the **uploader-create-send-to-shortcut.vbs** for right click Send to upload option(Windows only)

Features

- Supports uploading to Arduboy, DevKit, and homemade Arduboys
- Uploads .hex files, .hex files in .zip and .arduboy files

- Protects unprotected bootloaders from being overwritten by large hex files
- Supports on the fly patching for SSD1309 displays
- Supports on the fly RX and TX LED polarity patching for Arduino /Genuino Micro

Usage:

- Right click a .hex file, .arduboy file or .zip file containing a hex file and choose *Send To* Arduboy uploader
- Drag and drop .hex, .zip or .arduboy files on the **uploader.py** file
- Command line: uploader.py [filetoupload]

SSD1309 display support

To patch Arduboy hex files for use on Homemade Arduboy's with SSD1309 displays, make a copy of **uploader.py** and rename it to **uploader-1309.py** Also make sure you run the **uploader-create-send-to-shortcut.vbs** again to create a **Send To** shortcut for it. Files will be patched on fly, original files will not be altered.

Reverse RX and TX LED polarity support

To patch Arduboy hex files for use with Homemade Arduboy's based on Arduino / Genuino Micro, make a copy of **uploader.py** and rename it to **uploader-micro.py** and run the **uploader-create-send-to-shortcut.vbs** again to create a **Send To** shortcut for it.

EEPROM backup

You can backup your Arduboy's EEPROM by double clicking the **eeprom-backup.py** python script. The backup is saved to a time stamped file in the format **eeprom-backup.py-YYYYMMDD-HHMMSS.bin**

EEPROM restore

You can restore a previously made EEPROM backup simply by dragging the **eeprom-backup.py-YYYYMMDD-HHMMSS.bin** file onto the **eeprom-restore.py** python script

EEPROM erase

Erases the EEPROM content (An erased EEPROM contains all 0xFF's).

Erase sketch

Erases the application/sketch start-up page to keep the bootloader mode active indefinitely. This solves problematic (time sensitive) uploads using Arduino IDE.

Flash cart builder

- Works with both Python 2.7.x **AND** 3.7.x
- Requires PILlow: python -m pip install pillow

Builds a binary flash image from an index file and supporting resource files (.png images and .hex files). Use the **flashcart-writer.py** script to write the output to a flash cart. See the **example-flashcart\flashcart-index.csv** file for example syntax.

example: python flashcart-builder.py example-flashcart\flashcart-index.csv

Flash cart writer

- Works with both Python 2.7.x **AND** 3.7.x
- Requires pySerial: python -m pip install pyserial

Writes a binary flash image to external flash memory of Arduboy FX and Arduboy (clones) with added serial flash memory (Cathy3K v1.3+ bootloader required). Use the **flashcart-builder.py** script to build the image. To automatically apply the SSD1309 patch to the uploaded image, make a copy of **flashcart-writer.py** and rename it to **flashcart-writer-1309.py**.

example: python flashcart-writer.py example-flashcart\flashcart-image.bin
For development purposes external program data and save data can be stored at the end of external flash memory using -d and -s switches.

example: python flashcart-writer.py -d datafile.bin

Flash cart backup

- Works with both Python 2.7.x **AND** 3.7.x
- Requires pySerial: python -m pip install pyserial

Backup your existing flash cart to a binary image that can later be re-written to the Arduboy using the **flashcart-writer.py** script. The backup is saved to a time stamped file in the format **flashcart-backup-image-YYYYMMDD-HHMMSS.bin**

example: python flashcart-backup.py

Image Converter

- Works with both Python 2.7.x **AND** 3.7.x
- Requires PILlow: python -m pip install pillow

Converts .bmp or .png image files to C++ include file. Image width and height can be any size. Tile sheets and sprite sheets with optional spacing can be converted by specifying the width and height and optional spacing in the filename. When an image contains transparency information the converted data will include a sprite mask. Script can convert multiple files in one go by supplying multiple filenames.

example: python image-converter.py tilesheet_16x16.png