The GRUB Switch - Quickstart Guide

Ruediger Willenberg
Version 1.2
May 25, 2021

This guide expects that you know how to use the Linux command-line in a console window and are familiar with commands to change directories, list files and start bash scripts. You will also need *super user* access for some steps, so make sure you have **`sudo`** rights.

Refer to the detailed documentation if these quick steps do not work on your platform.

Table of Contents

1 Downloading and installing GRUB Switch files

The GRUB Switch scripts and files are designed to be installed and used under Linux. There are two ways to obtain the files:

- Via **git clone**:
  - Go to your preferred working directory
  - Clone the repository with

    ```
    git clone  https://github.com/rw-hsma-fpga/grub-switch.git
    ```

  - All GRUB Switch files are now available in the path

    ```
    ./grub-switch
    ```

- Via **download**:
  - Download to your preferred directory: https://github.com/rw-hsma-fpga/grub-switch/archive/refs/heads/master.zip
  - Unzip with GUI-tool or on the command line with

    ```
    unzip grub-switch-master.zip
    ```

  - All GRUB Switch files are now available in the path

    ```
    ./grub-switch-master
    ```

2 Configuring your boot choices

To configure your *GRUB Switch* boot choices, enter the bash script directory with

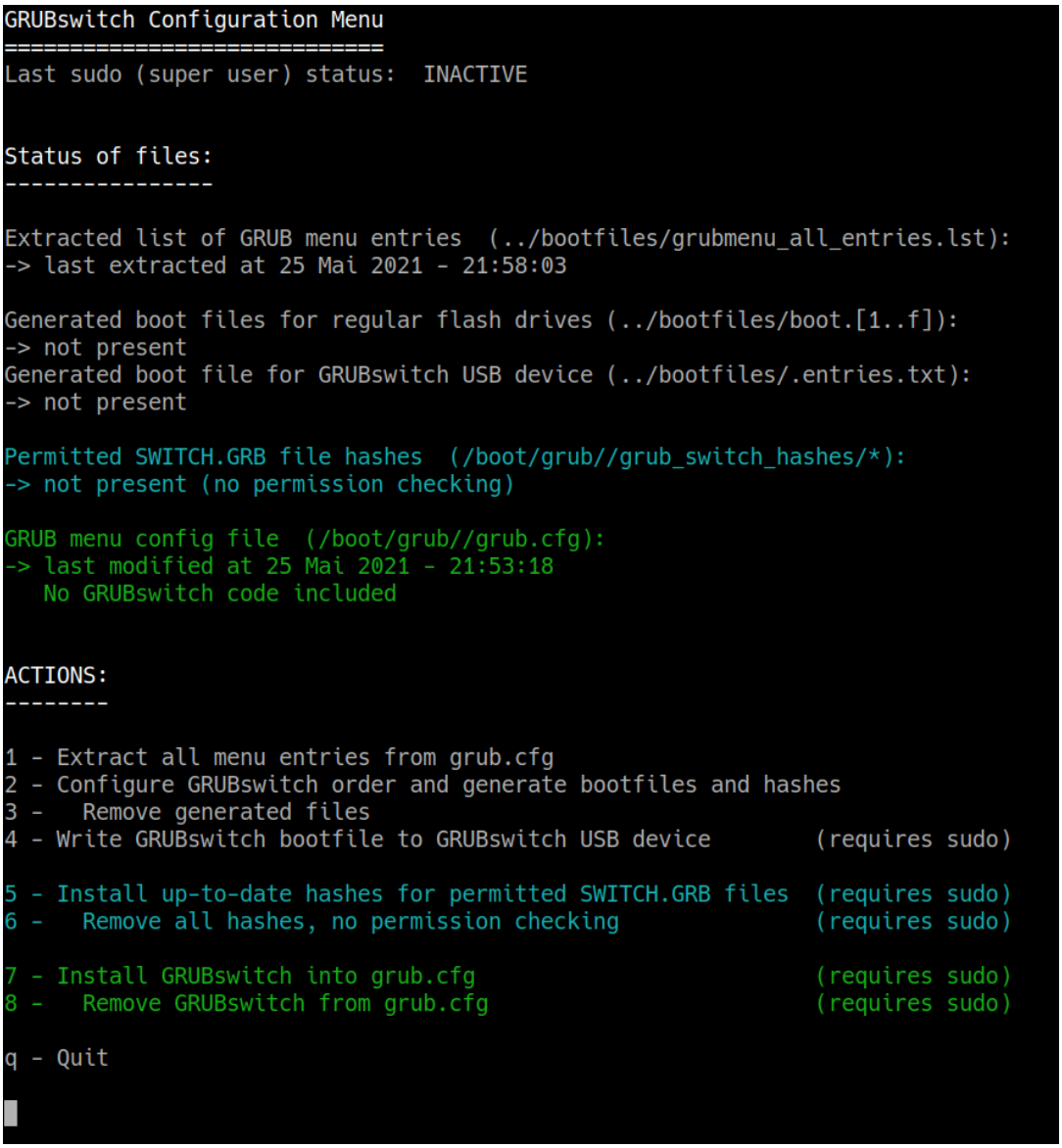
`cd grub-switch/1_shell_scripts/`

(exact path depends on your installation location and current working directory)

Start the configuration tool with

`./CONFIGURE_GRUBswitch.sh`

(we don't recommend starting the script with *sudo*, because all generated files would then be owned by *root*)



The main screen should look as above (depending on your console theme).

The tool lists on top if *sudo* rights, which are needed for some actions, have been established previously.

Below that, the script lists the status of important files either required or generated by this tool.

In the lower half, the tool offers a menu of actions to take by pressing a corresponding number key.

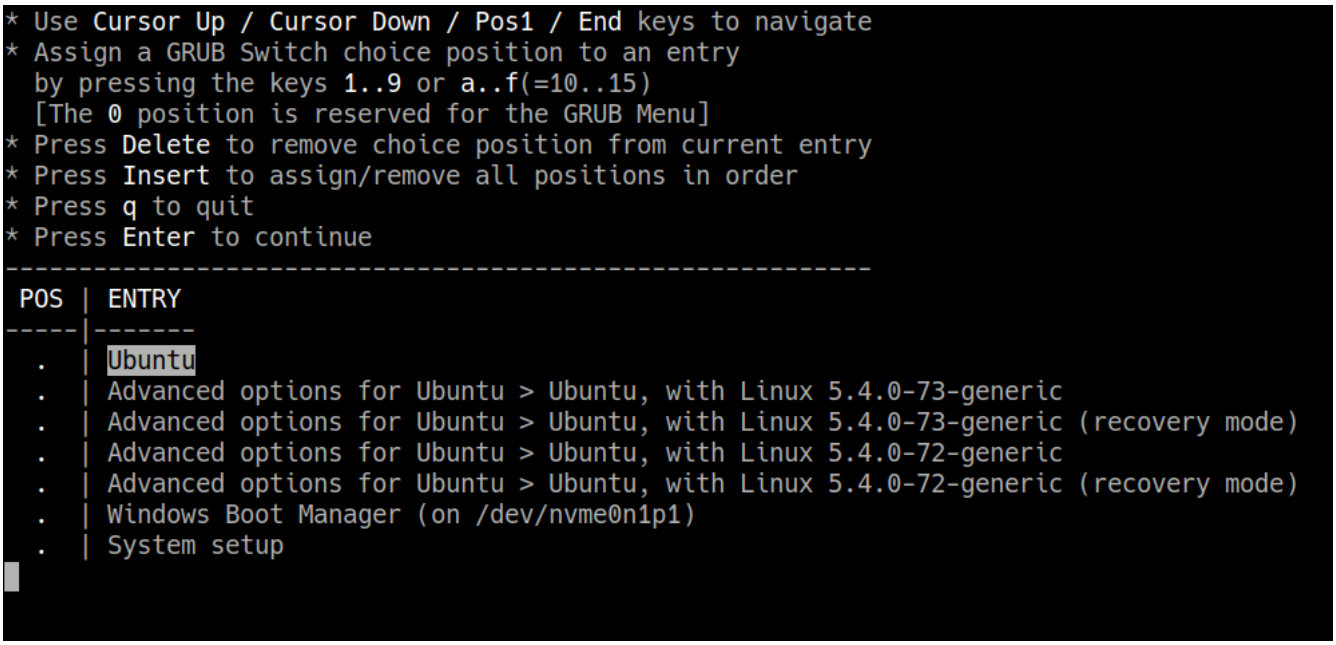
2.1 Extracting a list of GRUB menu entries

By choosing option **1**, the tool will parse GRUB's menu configuration file and generate a local text file **`grub-switch/bootfiles/grubmenu_all_entries.txt`** that lists all menu entries, including ones in submenus, and is the basis for picking your custom boot choices.

After returning to the main screen, the status of **`grubmenu_all_entries.txt`** should reflect the recent extraction date.

2.2 Picking and ordering your boot choices

Choose menu option **2** to start the actual *GRUB Switch* configuration.



The tool will show a complete list of boot choices in the order extracted from the GRUB configuration. Moving up and down with the cursor keys, you can use number keys (and the hexadecimal letters **a** to **f** representing 10..15) to assign choices a position in the *GRUB Switch* selection order.

```
* Use Cursor Up / Cursor Down / Pos1 / End keys to navigate
* Assign a GRUB Switch choice position to an entry
  by pressing the keys 1..9 or a..f(=10..15)
  [The 0 position is reserved for the GRUB Menu]
* Press Delete to remove choice position from current entry
* Press Insert to assign/remove all positions in order
* Press q to quit
* Press Enter to continue
-------------------------------------------------------
 POS | ENTRY
-----|-------
  2  | Ubuntu
  .  | Advanced options for Ubuntu > Ubuntu, with Linux 5.4.0-73-generic
  3  | Advanced options for Ubuntu > Ubuntu, with Linux 5.4.0-73-generic (recovery mode)
  .  | Advanced options for Ubuntu > Ubuntu, with Linux 5.4.0-72-generic
  .  | Advanced options for Ubuntu > Ubuntu, with Linux 5.4.0-72-generic (recovery mode)
  1  | Windows Boot Manager (on /dev/nvme0n1p1)
  4  | System setup
```

Not all entries have to be chosen and assigned a numerical position. The *Enter* key continues to the next screen.

```
Switch positions chosen assigned:
---------------------------------
0 : GRUB Menu (Fixed)
1 : Windows Boot Manager (on /dev/nvme0n1p1)
2 : Ubuntu
3 : Advanced options for Ubuntu>Ubuntu, with Linux 5.4.0-73-generic (recovery mode)
4 : System setup
5 :
6 :
7 :
8 :
9 :
a :
b :
c :
d :
e :
f :
-----------------------------------------
* Press Enter to confirm selection
* Press Backspace <- to change configuration
* Press q to quit
```

The next screen shows the selected choices and position assignments. They can be confirmed with the *Enter* key; the *Backspace* key returns to the previous screen for further changes.

```
-------------------------------------------------------
Set boot choice display time:
* Press Cursor Up / Cursor Down to change +/- 10 seconds
* Press Cursor Right/ Cursor Left to change +/- 1 second
* Press Enter to confirm selection
* Press q to quit
-------------------------------------------------------

Show boot choice for 005 seconds
```

It is possible to display the chosen boot option for a number of seconds, which can be adjusted in the next tool screen. If you keep or set the value to 0 seconds, there will be no choice display. The operating system selected by *GRUB Switch* will be booted instantly.

```
-------------------------------------------------------
Choose highlight colors for boot choice display:
* Press Cursor Right/ Cursor Left to change
* Press Enter to confirm selection
* Press Backspace <- to go back and change display time
* Press q to quit
-------------------------------------------------------

Current choice is white/red.
See example below:


   Booting FluxCapOS 1.21
   continues in 005 seconds


(Caution: Not all Linux terminals can display the
 complete range of colors that GRUB supports)
```

The last configuration screen allows choosing a highlight color (foreground/background) for the choice display. Enter finishes the boot configuration. A final screen shows the output written into the configuration file **`grub-switch/bootfiles/.entries.txt`**, which is used to configure the switchable USB storage device.

```
The following data was written to ../bootfiles/.entries.txt:

### #1 specifies display time for subsequent entries
### #2 specifies display colors for subsequent entries
### Uncommented lines are GRUB switch choices 1..15 (0x1..0xF)
### An empty line leads to the GRUB menu, as does choice 0
#1 005
#2 white/red
Windows Boot Manager (on /dev/nvme0n1p1)
Ubuntu
Advanced options for Ubuntu>Ubuntu, with Linux 5.4.0-73-generic (recovery mode)
System setup
```

2.3 Securing possible boot choices through hash checks

TBD

2.4 Activating The GRUB Switch in GRUB

TBD

3 Adding GRUB Switch to your GRUB installation

Aa

4 Using regular USB flash drives for boot choice

Picking and configuring boot entries with option **2** in the **CONFIGURE_GRUBswitch.sh** script has generated directories and files like these in your directory **grub-switch/bootfiles** :

`bootfiles/boot.1/SWITCH.GRB`

`bootfiles/boot.2/SWITCH.GRB`

`bootfiles/boot.2/SWITCH.GRB`

You will find **boot.*** dirs according to the positions you assigned during configuration. There won't be a **boot.0** dir because that choice is reserved for the GRUB menu. Each directory holds a **SWITCH.GRB** file that sets the variable for the corresponding boot choice when called by GRUB. You can open it with a text editor, though I don't recommend editing it by hand.

You will now need as many USB flash drives as you have picked boot choices (excluding the GRUB menu).

- Make sure each flash drive is formatted with a FAT file system (FAT12/16/32, exFAT, VFAT), not with NTFS or any other file system.
  You do not need to re-format an existing drive, or even delete its files (although you might want to for reasons of cleanliness, data protection, privacy, etc.).
- Copy each **SWITCH.GRB** file to a separate flash drive and mark the drive with the corresponding number, OS name or whatever helps you tell them apart.
- Before powering on your computer or rebooting, plug in the flash drive representing your boot choice.
- If you want the GRUB menu to appear, make sure none of the drives are plugged in.
- If multiple of your drives are plugged in by accident, boot choice will depend on the ordering of USB connections in your BIOS, which you might not be able to tell easily.

All further chapters are not relevant if you stick with this method to make your boot choice.

5 Using hardware switches and a programmable USB board for boot choice

If you want to make your boot choice through an electromechanical switch, you need to use a programmable USB board, flash it with our firmware and connect a suitable switch to it.

Our firmware is designed for boards with the Microchip/Atmel ATmega32u4 chip, which can act as a USB device. All related files can be found in **`grub-switch/2_usb_device`**

5.1 Supported boards

Currently the project includes pre-built firmware images for the following products:

- PJRC Teensy 2.0
- Arduino Micro
- SparkFun Pro Micro (3V and 5V versions) and various clones
- Adafruit ItsyBitsy 32u4 (3V and 5V versions)
- DFRobot Beetle
- Our self-designed, non-commercial Custom Hardware PCB (see 5.2.3)

It is fairly simple to adapt the firmware to another board with the ATmega32u4 and build a firmware image. See the detailed documentation for instructions.

5.2 Programming the boards with GRUB Switch firmware

On the Linux command-line, go to the **`grub-switch/2_usb_device/prebuilt_images`** directory, where you can find the suitable image and programming scripts.

5.2.1 Teensy 2.0

Teensy 2.0 has a proprietary bootloader which works with its own GUI programming tool, Teensy Loader. You can find precise installation and download instructions here:

https://www.pjrc.com/teensy/loader_linux.html

The corresponding image is **`prebuilt_images/GRUB_SWITCH_TEENSY20.hex`**

5.2.2 Arduino-compatible boards

Arduino Micro, **ItsyBitsy**, **Pro Micro** and **Beetle** all use a bootloader that is compatible with Arduino IDE, however the IDE does not have an option for downloading any random *.hex file.

We therefore require a command-line tool called **avrdude**, which is available in many Linux repositories. Under Ubuntu, for example, you can install it with

```
sudo apt-get install avrdude
```

To simplify the use of **avrdude**, we have provided a script in **`prebuilt_images`**. After plugging in your board, it can be started with:

```
sudo ./program_grubswitch_arduino.sh CORRECT_FIRMWARE.hex
```

(The use of *sudo* might not be required on your installation)

The script will ask you to put the connected board into *bootloader mode*, which for the **Arduino Micro** and **ItsyBitsy** can be accomplished by pushing the reset button twice.

The **Pro Micro** does not have a reset button; you have to short the **GND** and **RST** pads twice in short order with a wire.

The **Beetle** has a group of 6 tiny contact pads on the bottom side under its USB connector:

```
1 3 5
# 0 0
0 0 0
2 4 6
```

Pad number 1 is square, not round.

Short the round pads number 5 and 6 twice in short order to put **Beetle** into *bootloader mode*

After putting your board into *bootloader mode*, you have between 5 and 10 seconds to press a keyboard key for the programming script to continue; any longer, and the board leaves *bootloader mode* again.

After successful firmware programming, the board identifies as a FAT12 storage device. Depending on your Linux GUI, you might get a device notification.

NOTE: There are separate firmware images for the 3V and 5V versions of both ItsyBitsy and Pro Micro, as they need different oscillator configurations. Downloading the wrong firmware will not cause harm, the firmware will just not function. The board can be brought into *bootloader mode* again, and reprogrammed with the correct firmware.

5.2.3 THE GRUB SWITCH custom USB hardware Rev.B

Our custom USB device design relies on the *DFU bootloader* which is factory-built into each ATmega32u4 chip. There is an open source command-line tool called **dfu-programmer** which needs to be installed for firmware falshing. Under Ubuntu Linux, it can be installed with:

```
sudo apt-get install dfu-programmer
```

To simplify the use of **dfu-programmer**, we have provided a script in **`prebuilt_images`**. After plugging in your board, it can be started with:

```
sudo ./program_grubswitch_dfu.sh GRUB_SWITCH_CUSTOMHW_REVB.hex
```

(The use of *sudo* might not be required on your installation)

Before or after starting the script, the chip needs to be brought into *bootloader mode* by pushing or shorting the **PROG** button. Unlike the Arduino boards, there is no bootloader timeout. Pressing a key on the keyboard will start programming.

After successful firmware programming, the board identifies as a FAT12 storage device. Depending on your Linux GUI, you might get a device notification.

5.3 Writing the GRUB Switch boot configuration onto your storage device

Together with the **`boot.*`** directories, option **`2`** of the **`CONFIGURE_GRUBswitch.sh`** script has also written a single text file holding the complete list and order of boot choices:

```
bootfiles/.entries.txt
```

You can open with a text editor to see the ordered list of boot choices, however it is not recommended to edit the file by hand. Because the filename starts with a period, it is only visible if you use the -**`a`** option with **`ls`**:

```
ls -la bootfiles/
```

If you mount your USB storage device automatically through the GUI, you can just copy the file onto the device and unmount.

Alternatively, you can use option **`4 - Write GRUBswitch bootfile to GRUBswitch USB device`** in the **`CONFIGURE_GRUBswitch.sh`** script. This requires giving the *sudo* password because the mounting and unmounting of the device is being forced.

NOTE: Your USB device is not a regular flash drive. **`.entries.txt`** is the only file that will be accepted for writing, and only if the **Write-Protect** jumper is not set on the board (see wiring section). The only other (read-only) files on the device are:

`SWITCH.GRB` – The actual boot setting script read by GRUB. If there are no switch connections on the board yet, this will be empty because that is the default leading to the GRUB menu.

`.bootpins.txt` – This file (also hidden without **`ls -a`**) shows the choice pins currently connected to GND as ones. It is only included for testing purposes.

5.4 Electrical connections to the GRUB Switch boards

This section shows the various ways to connect switches to each board to control the boot choice.

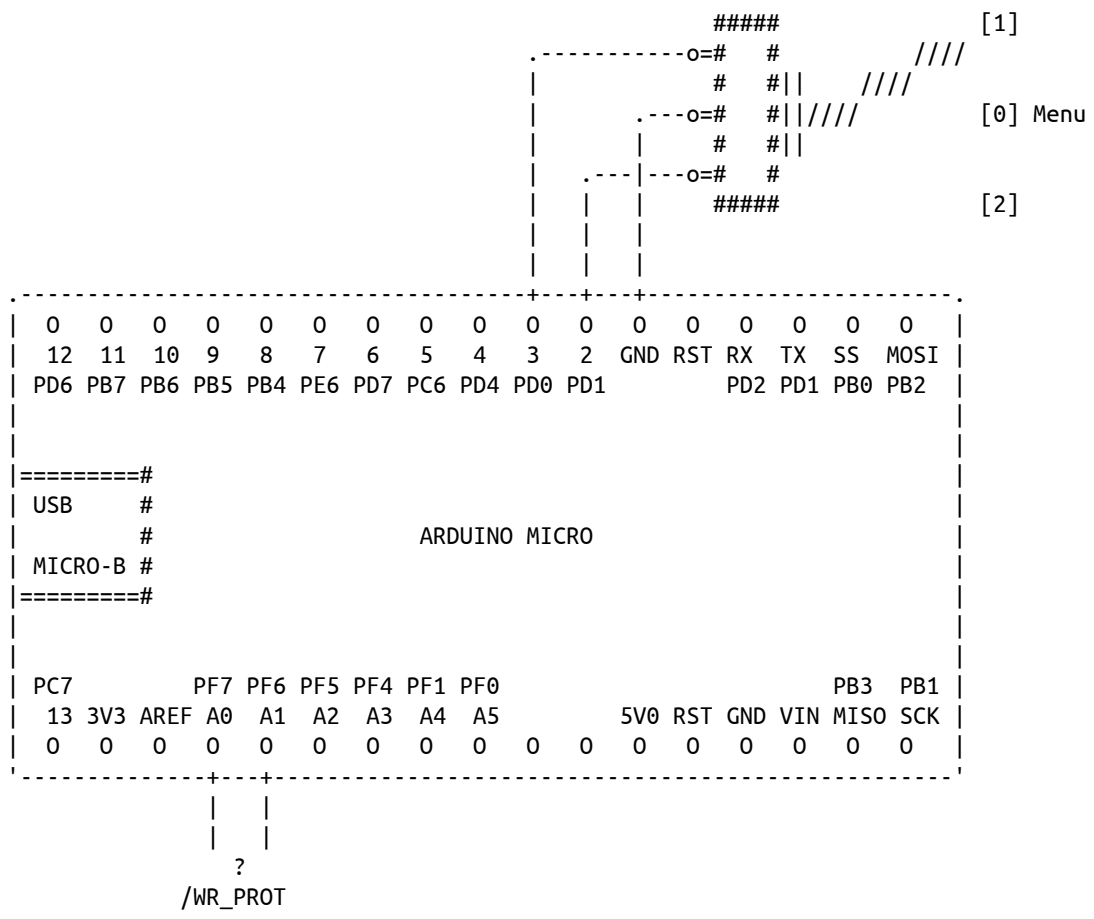
5.4.1 Arduino Micro

```
1-OF-N MODE  with  ON/OFF/ON toggle switch:
-------------------------------------------

To choose this mode, pin PF1 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Using a three-position on/off/on switch permits choosing between two
OS options, while the neutral middle position leads to the GRUB menu
```

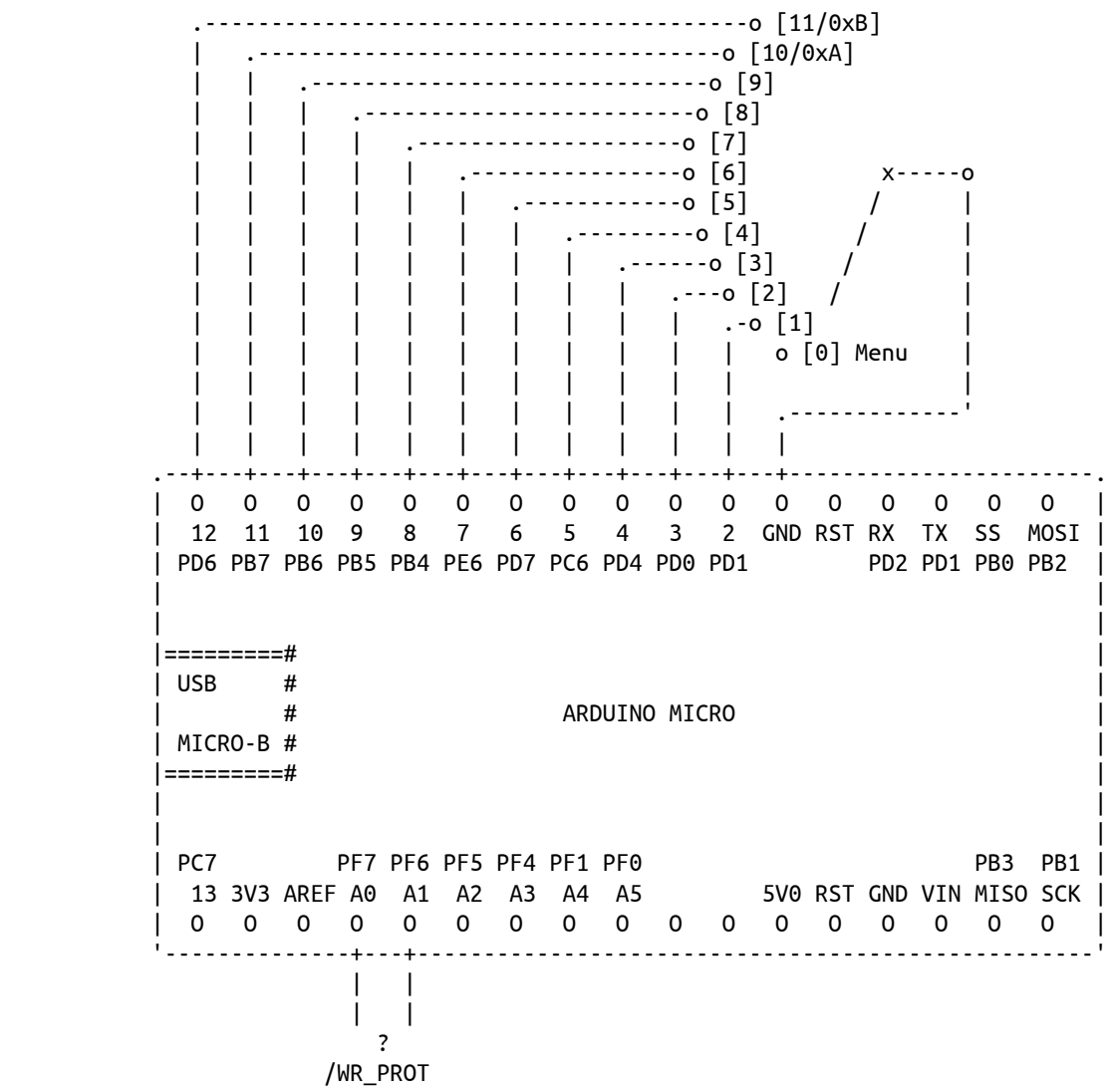


1-OF-N MODE with rotary switch: (Motto: "This one goes to 11!")

To choose this mode, pin PF1 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Common rotary switches have up to 12 positions (some can be limited to fewer
positions with a mechanical keyring). Keeping the first position [0] unconnected
permits choosing the GRUB menu and leaves 11 boot choices, which is the maximum
number of pins sampled. Connecting one of the pins PD1..PD6 to GND
therefore reflects boot choices 1..11 (hexadecimal 0x1..0xB). Accidentally
connecting multiple pins results in the lowest-order entry being chosen.

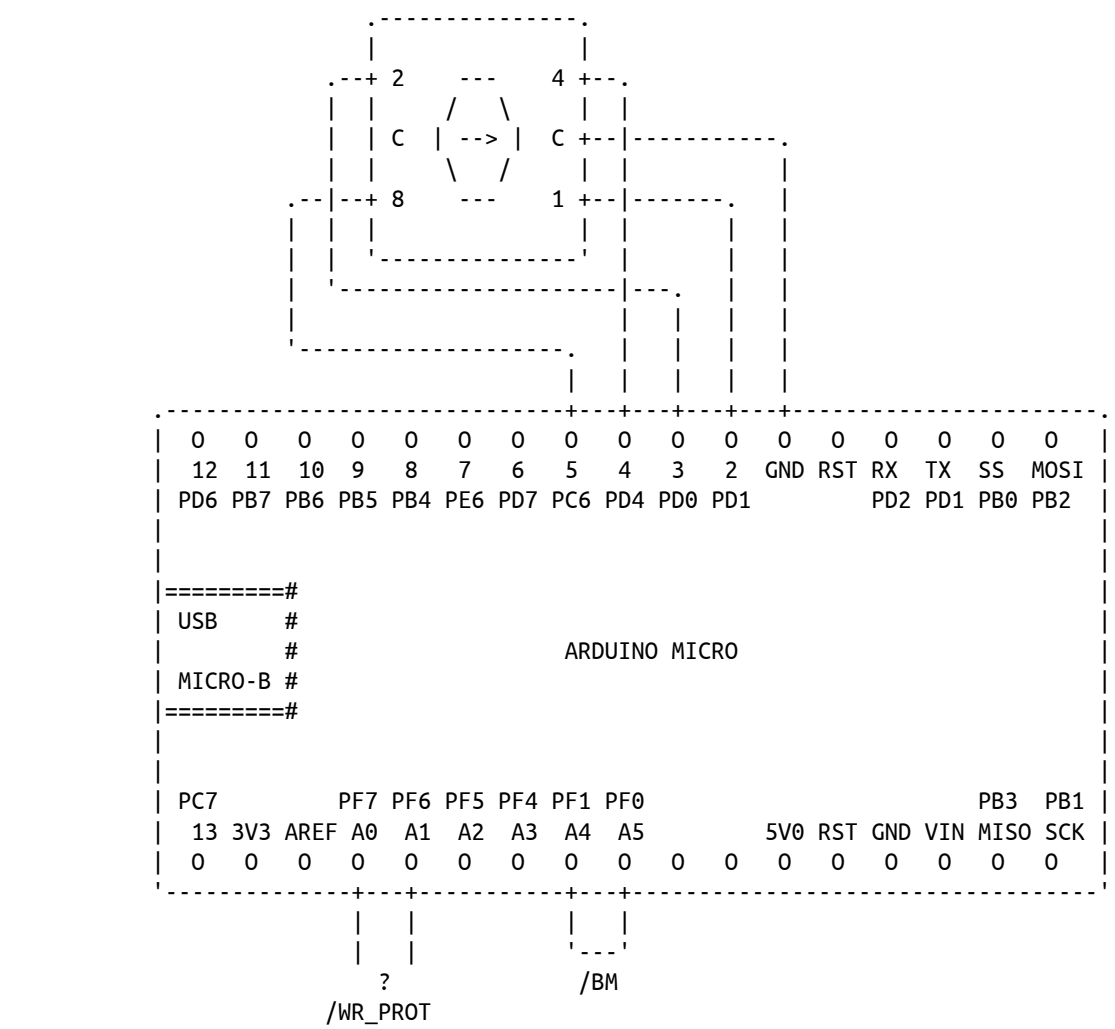


BINARY MODE:

To choose this mode, pin PF1 needs to be tied to GND or adjacent pad PF0.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Common binary-encoding rotary switches have 10 to 16 positions and connect
the corresponding bit values 1,2,4,8 [LSB..MSB] to the 'C'-contacts; using the
device's binary mode means tieing the four pins PD1..PC6 [LSB..MSB] to GND/0V
as to encode the corresponding boot choices 0(GRUB Menu) up to 15(0xF).
Obviously a proper nerd can also use four separate switches to encode the
choice in binary herself!
(Reminder: The device interprets a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.



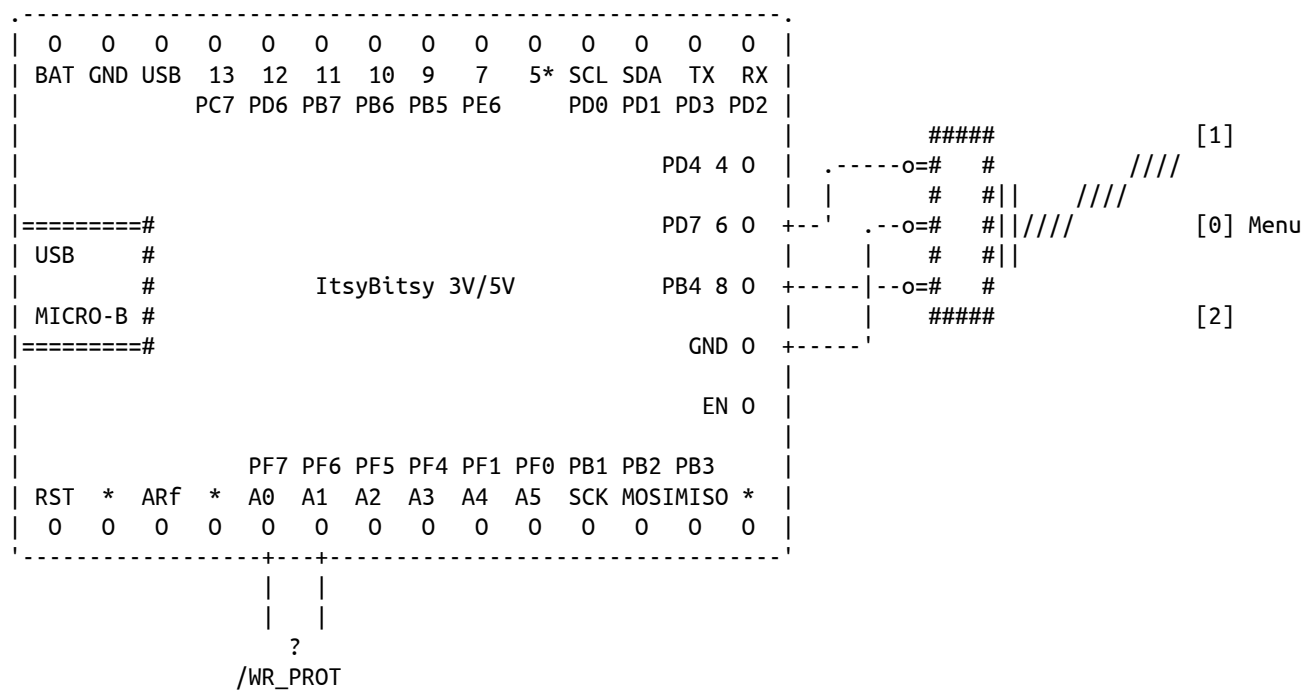
5.4.2 Adafruit ItsyBitsy 3V/5V

```
1-OF-N MODE  with  ON/OFF/ON toggle switch:
-------------------------------------------

To choose this mode, pin PF1 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Using a three-position on/off/on switch permits choosing between two
OS options, while the neutral middle position leads to the GRUB menu
```

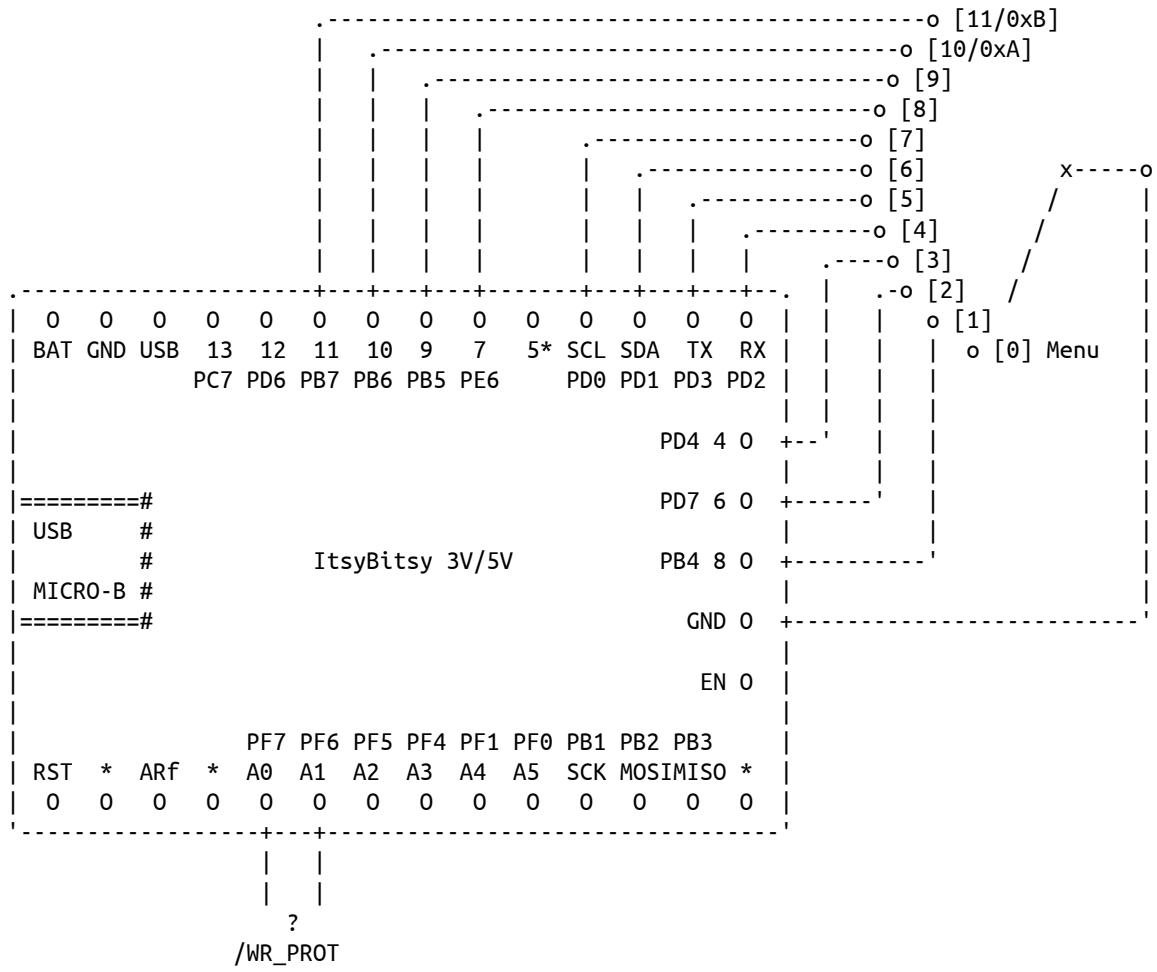


1-OF-N MODE with rotary switch: (Motto: "This one goes to 11!")

To choose this mode, pin PF1 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Common rotary switches have up to 12 positions (some can be limited to fewer
positions with a mechanical keyring). Keeping the first position [0] unconnected
permits choosing the GRUB menu and leaves 11 boot choices, which is the maximum
number of pins sampled. Connecting one of the pins PB4..PB7 to GND
therefore reflects boot choices 1..11 (hexadecimal 0x1..0xB). Accidentally
connecting multiple pins results in the lowest-order entry being chosen.

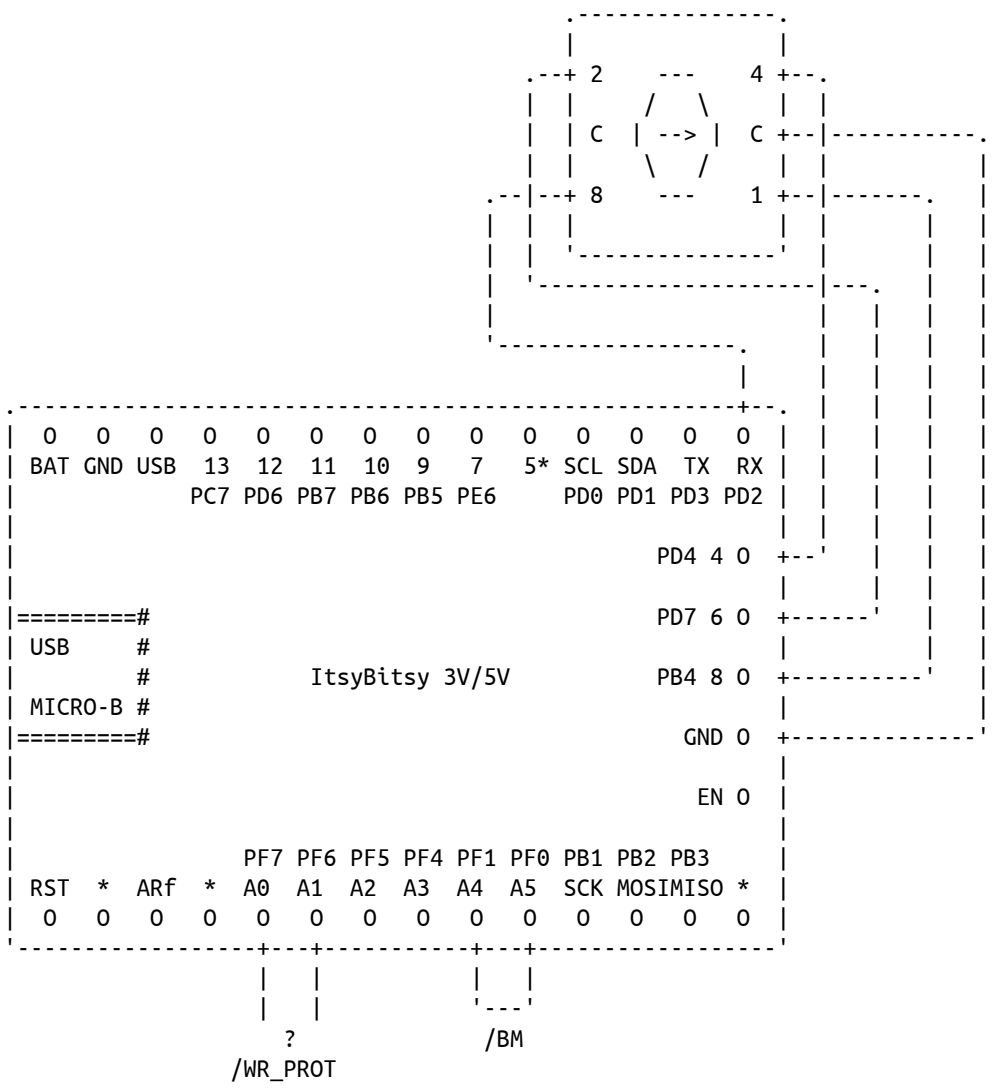


BINARY MODE:

To choose this mode, pin PF1 needs to be tied to GND or adjacent pad PF0.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent pad, driving a low level).

Common binary-encoding rotary switches have 10 to 16 positions and connect
the corresponding bit values 1,2,4,8 [LSB..MSB] to the 'C'-contacts; using the
device's binary mode means tieing the four pins PB4..PD2 [LSB..MSB] to GND/0V
as to encode the corresponding boot choices 0(GRUB Menu) up to 15(0xF).
Obviously a proper nerd can also use four separate switches to encode the
choice in binary herself!
(Reminder: The device interprets a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.



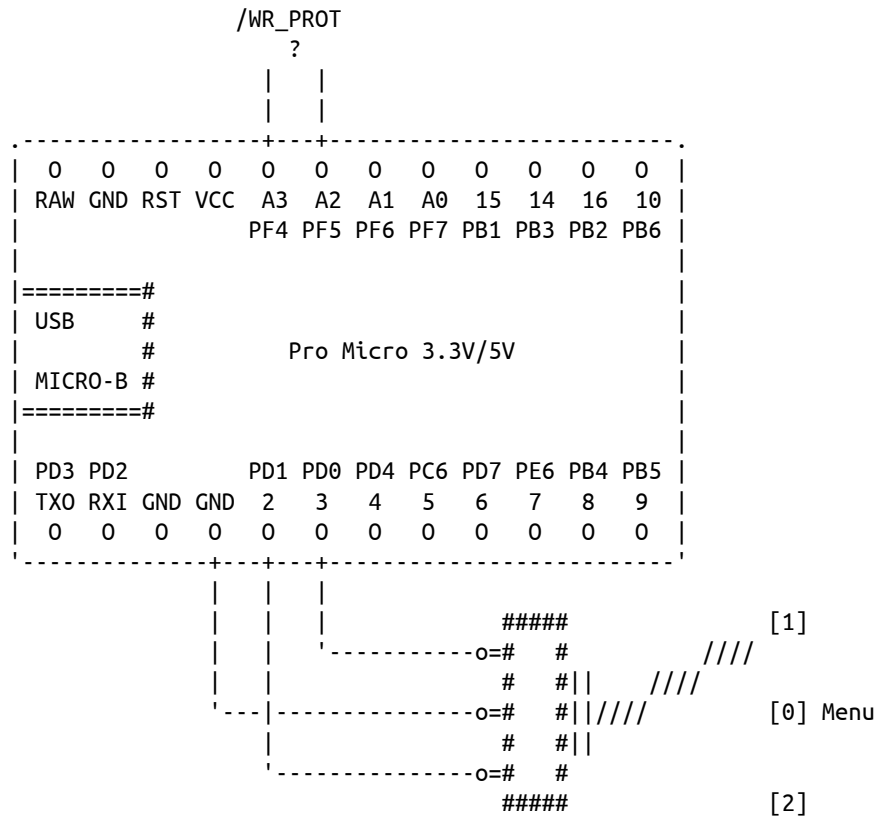
5.4.3 Sparkfun Pro Micro 3V/5V (and clones)

```
1-OF-N MODE  with  ON/OFF/ON toggle switch:
-------------------------------------------

To choose this mode, pin PD2 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF5 (adjacent pad, driving a low level).

Using a three-position on/off/on switch permits choosing between two
OS options, while the neutral middle position leads to the GRUB menu
```

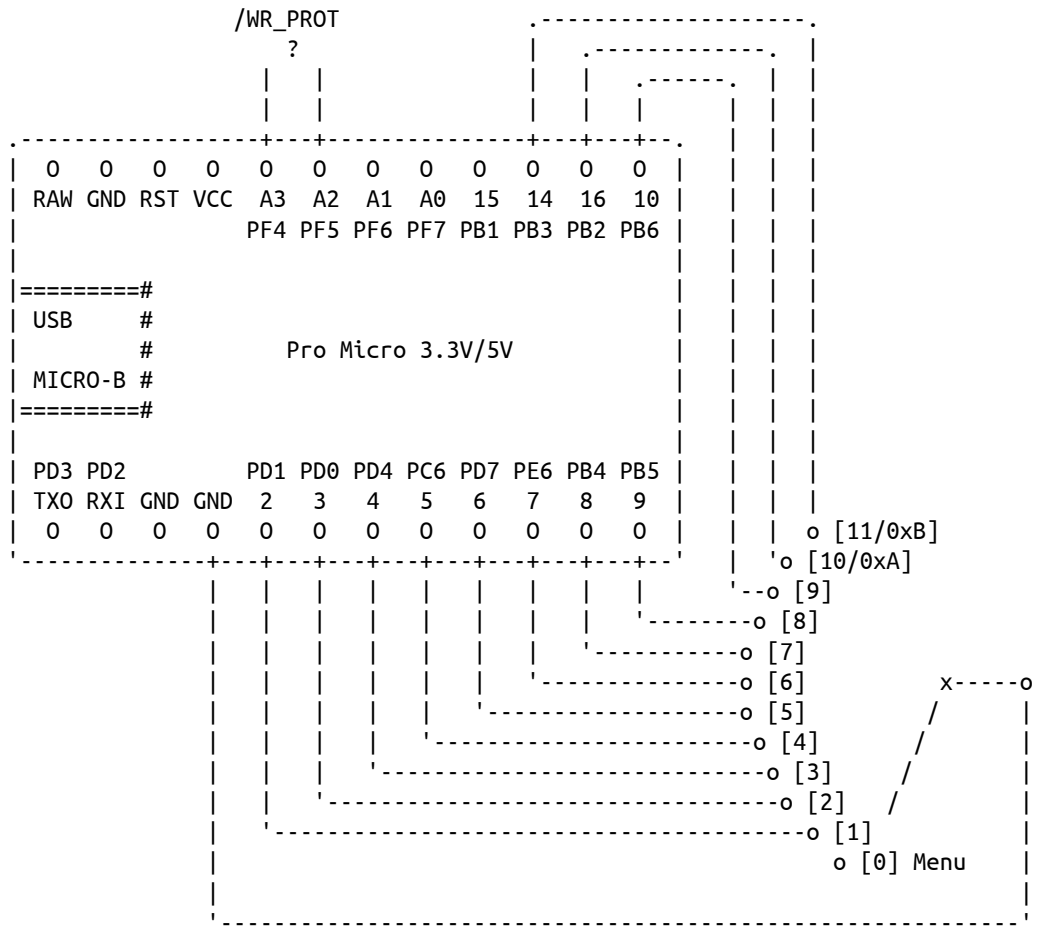


1-OF-N MODE with rotary switch: (Motto: "This one goes to 11!")

To choose this mode, pin PD2 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF5 (adjacent pad, driving a low level).

Common rotary switches have up to 12 positions (some can be limited to fewer
positions with a mechanical keyring). Keeping the first position [0] unconnected
permits choosing the GRUB menu and leaves 11 boot choices, which is the maximum
number of pins sampled. Connecting one of the pins PD1..PB3 to GND
therefore reflects boot choices 1..11 (hexadecimal 0x1..0xB). Accidentally
connecting multiple pins results in the lowest-order entry being chosen.

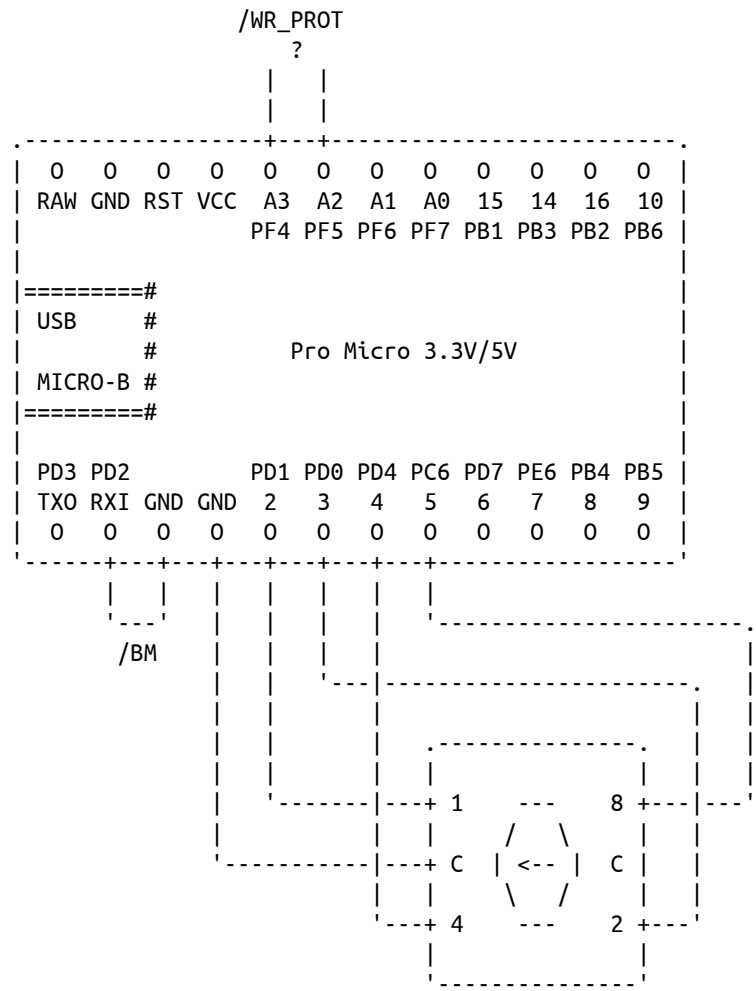


BINARY MODE:

To choose this mode, pin PD2 needs to be tied to the adjacent GND pin.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF5 (adjacent pad, driving a low level).

Common binary-encoding rotary switches have 10 to 16 positions and connect
the corresponding bit values 1,2,4,8 [LSB..MSB] to the 'C'-contacts; using the
device's binary mode means tieing the four pins PD1..PC6 [LSB..MSB] to GND/0V
as to encode the corresponding boot choices 0(GRUB Menu) up to 15(0xF).
Obviously a proper nerd can also use four separate switches to encode the
choice in binary herself!
(Reminder: The device interprets a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.



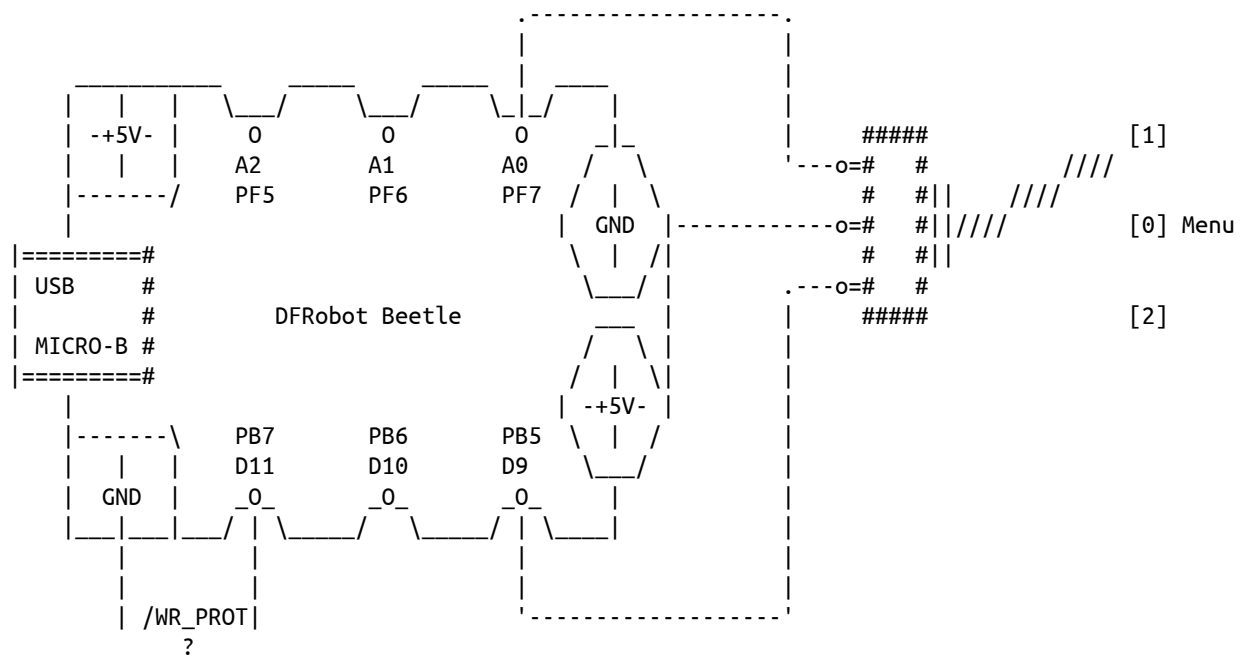
5.4.4 DFRobot Beetle

```
1-OF-N MODE  with  ON/OFF/ON toggle switch:
-------------------------------------------

To choose this mode, pin PB6 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PB7 to GND.

Using a three-position on/off/on switch permits choosing between two
OS options, while the neutral middle position leads to the GRUB menu
```

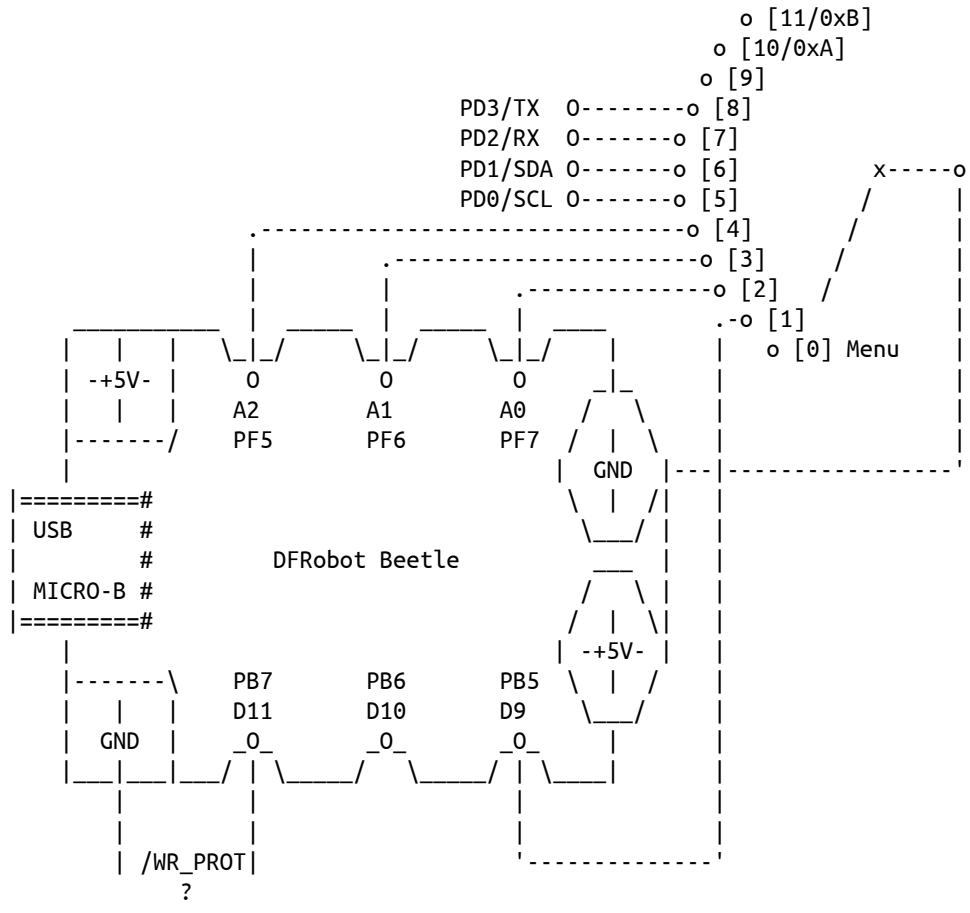


1-OF-N MODE with rotary switch: (Motto: "This one goes to 11!")

To choose this mode, pin PB6 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PB7 to GND.

Common rotary switches have up to 12 positions (some can be limited to fewer
positions with a mechanical keyring). Keeping the first position [0] unconnected
permits choosing the GRUB menu and in principle leaves 11 boot choices, however
Beetle only has 8 accessible connection points left, side pads PB5..BF5 for
positions [1]..[4] and the PCB bottom pads PD0..PD3 for positions [5]..[8]
Connecting one of those pads to GND through the switch reflects the boot
choices 1..8. Accidentally connecting multiple pins results in the lowest-order
entry being chosen.

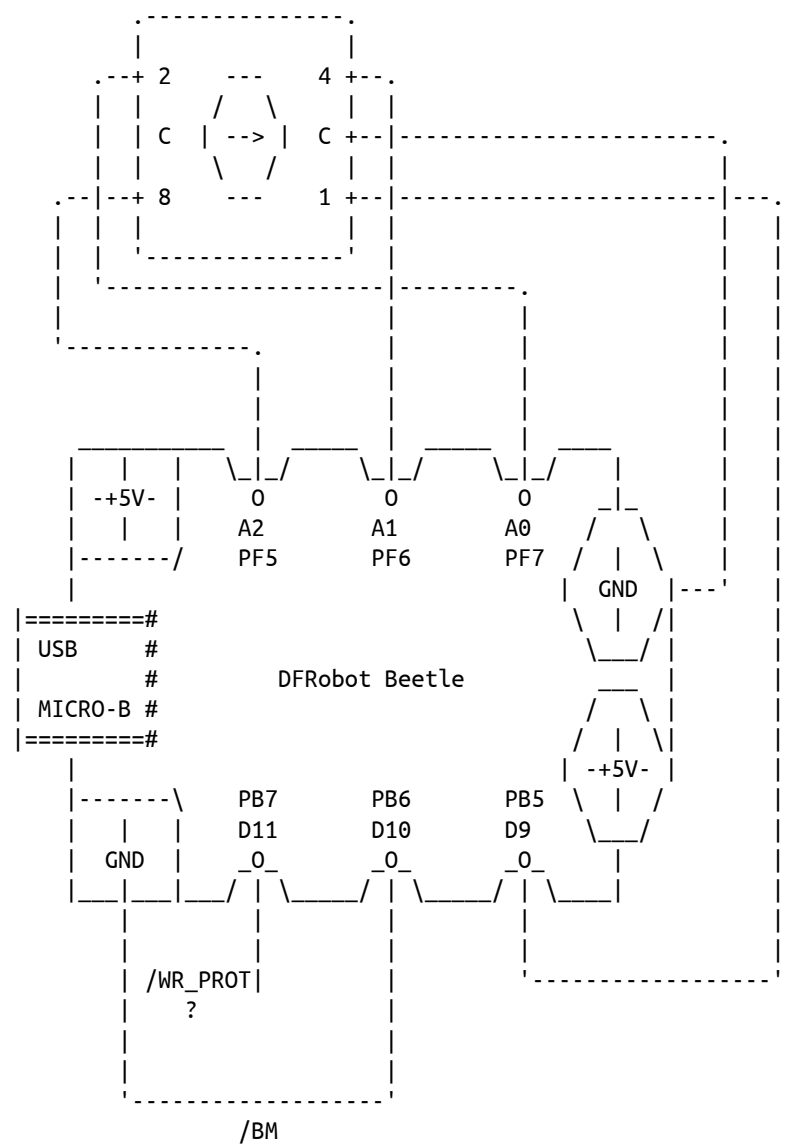


BINARY MODE:

To choose this mode, pin PB6 needs to be tied to GND.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PB7 to GND.

Common binary-encoding rotary switches have 10 to 16 positions and connect
the corresponding bit values 1,2,4,8 [LSB..MSB] to the 'C'-contacts; using the
device's binary mode means tieing the four pins PB5..PF5 [LSB..MSB] to GND/0V
as to encode the corresponding boot choices 0(GRUB Menu) up to 15(0xF).
Obviously a proper nerd can also use four separate switches to encode the
choice in binary herself!
(Reminder: The device interprets a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.



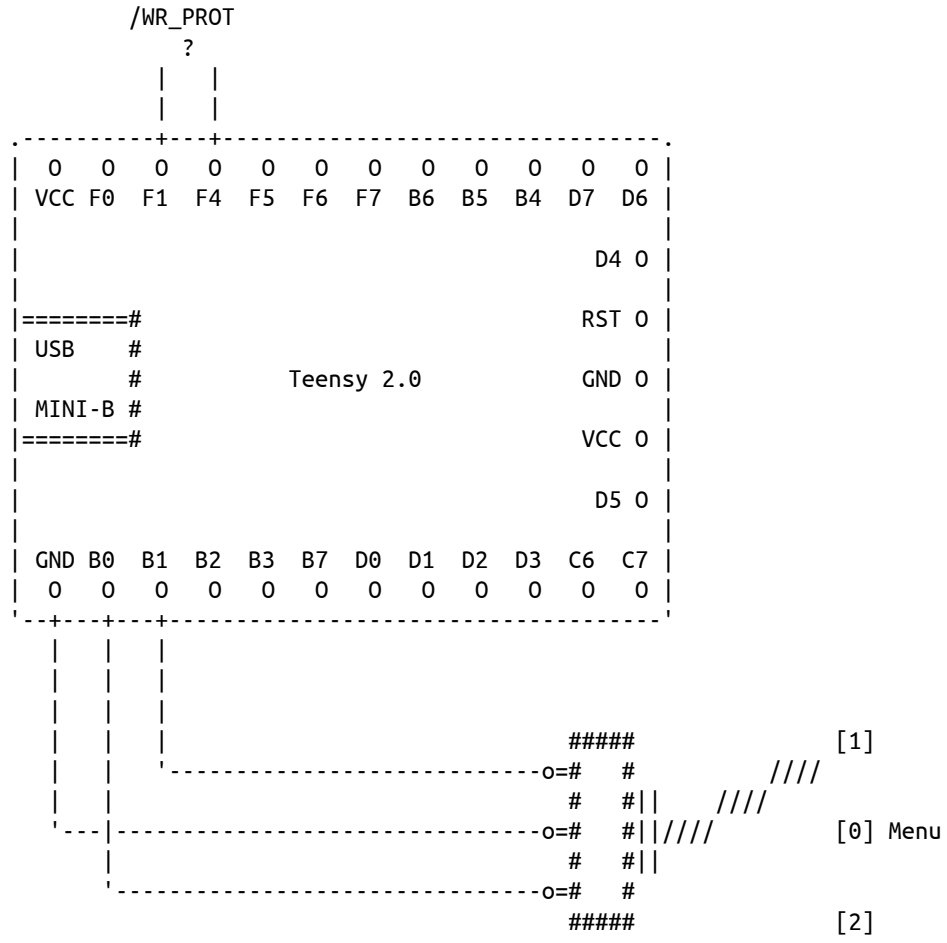
5.4.5 PJRC Teensy 2.0

```
1-OF-N MODE  with  ON/OFF/ON toggle switch:
-------------------------------------------

To choose this mode, pin PB6 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF1 (adjacent pad, driving a low level).

Using a three-position on/off/on switch permits choosing between two
OS options, while the neutral middle position leads to the GRUB menu
```

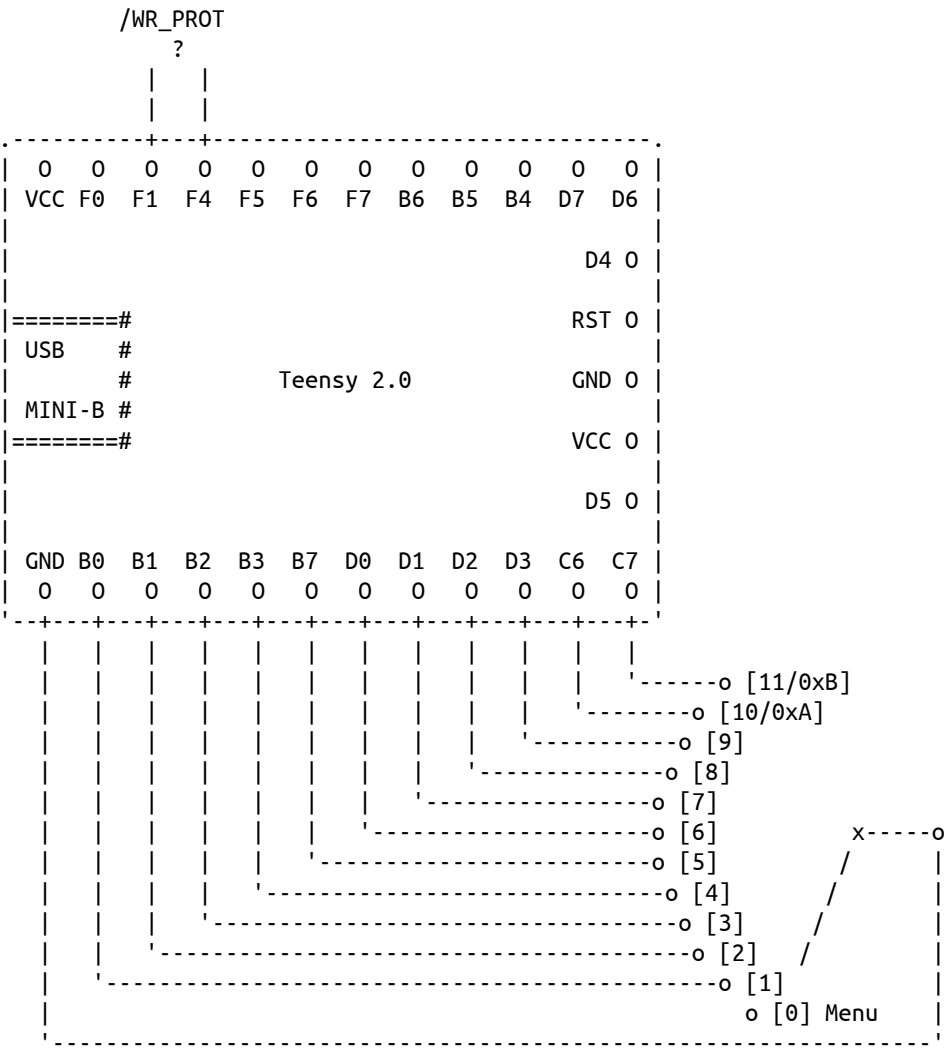


1-OF-N MODE with rotary switch: (Motto: "This one goes to 11!")

To choose this mode, pin PB6 needs to stay UNCONNECTED.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF1 (adjacent pad, driving a low level).

Common rotary switches have up to 12 positions (some can be limited to fewer
positions with a mechanical keyring). Keeping the first position [0] unconnected
permits choosing the GRUB menu and leaves 11 boot choices, which is the maximum
number of pins sampled. Connecting one of the pins PB0..PC7 to GND
therefore reflects boot choices 1..11 (hexadecimal 0x1..0xB). Accidentally
connecting multiple pins results in the lowest-order entry being chosen.

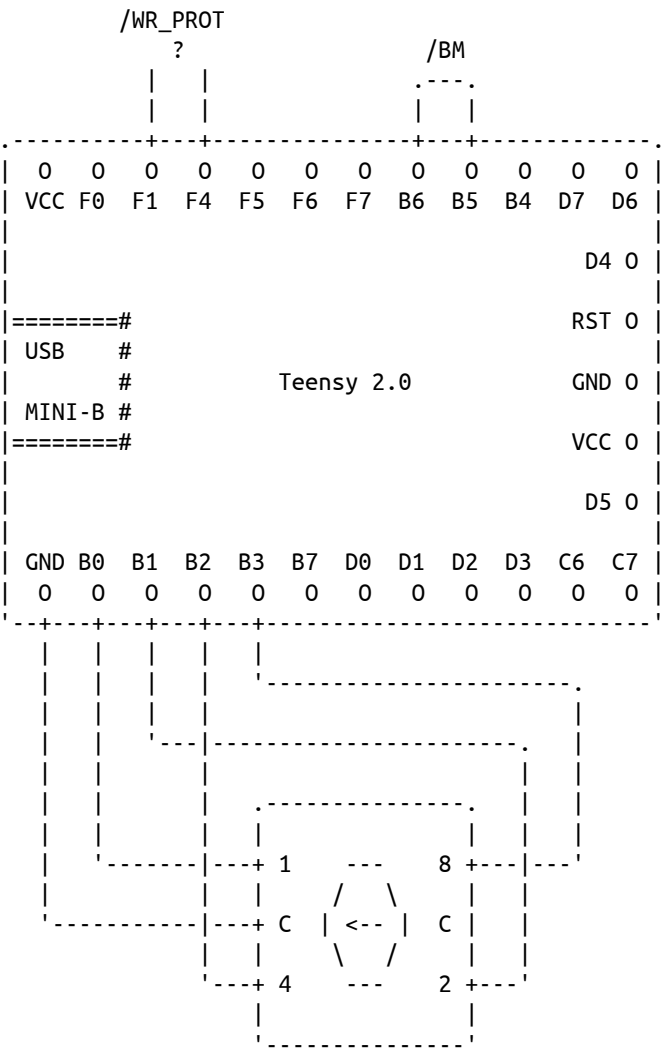


BINARY MODE:

To choose this mode, pin PB6 needs to be tied to GND or adjacent pad PB5.

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF4 to GND or PF1 (adjacent pad, driving a low level).

Common binary-encoding rotary switches have 10 to 16 positions and connect
the corresponding bit values 1,2,4,8 [LSB..MSB] to the 'C'-contacts; using the
device's binary mode means tieing the four pins PB0..PB3 [LSB..MSB] to GND/0V
as to encode the corresponding boot choices 0(GRUB Menu) up to 15(0xF).
Obviously a proper nerd can also use four separate switches to encode the
choice in binary herself!
(Reminder: The device interprets a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.

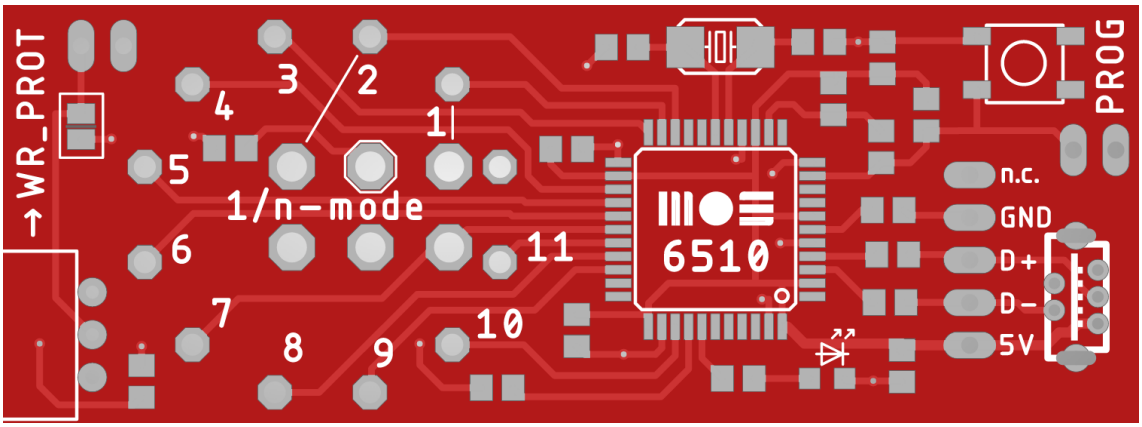


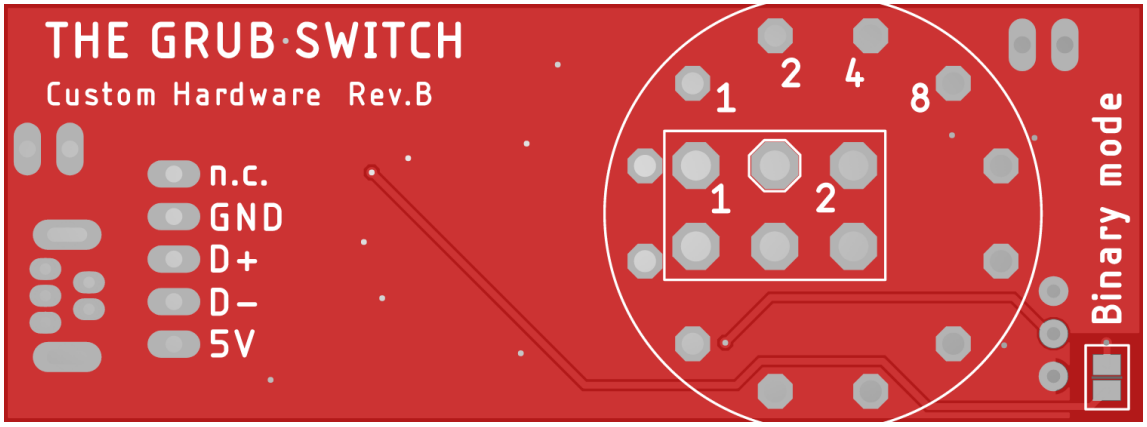
5.4.6 The GRUB SWITCH Custom Hardware Revision B

Our custom PCB has been designed with an eye to directly mounting and soldering two common types of switches for **1-of-n mode** (shorting one single choice pin to ground at a time).

The silhouettes for both a **1-of-12 rotary switch** and a **on/off/on toggle switch** are visible on the PCB side titled with “THE GRUB SWITCH” (the *bottom* layer in the PCB files). The switches are supposed to be *through-hole*-mounted on this side (which means they will be soldered on the side labeled with “1/n-mode”). Therefore, this is also the side facing the front panel of a case.

Because both types of switches are available with a screw thread for front-panel mounting, the PCB and switch can be affixed mechanically to a case.


→WR_PROT
3
2
4
1
5
1/n-mode
6
11
7
8
9
10
6510
PROG
n.c.
GND
D+
D-
5V


THE GRUB·SWITCH
Custom Hardware Rev.B
n.c.
GND
D+
D-
5V
1
2
4
8
1
2
Binary mode

1-of-n mode with ON/OFF/ON switch

The rectangular white silhouette shows the mounting position for a two-channel on/off/on toggle switch (DPDT - ”double-pole/double-throw”). Only the upper channel is actually used; the two channel model was chosen to decrease mechanical stress on solder pads and increase stability.

The pad numbers 1 and 2 indicate the boot choices made through electrical connection to the center pad. Note that the toggle switch lever moves across a fixed pivot, so moving the lever to the left (as seen from the front) actually connects choice 2, while moving the lever to the right chooses boot option 1. The neutral center position leads to the GRUB menu.

The following toggle switch series include DPDT on/off/on models with the required 4.7-4.75mm distance between PCB pins:

APEM Series 5000 E-Switch Series 100

NKK Series M TE-Connectivity Gemini A

(these suggestions should not be considered a commercial endorsement)

1-of-n mode with 1-of-12 rotary switch

The round silhouette indicates the mounting position of a 1-of-12 rotary switch. On the PCB side labeled with “1/n-mode” the 11 switch positions for boot choices of increasing order are indicated; position 0 leads to the menu. Because the switch is mounted on the other PCB side, choices increase by turning the switch clockwise.

The switch’s main connection pin, which is tied to GND, shares a pad with the toggle switch and is indicated by an octagonal silhouette. It is not in the center of the rotary switch, therefore the switch can only be mounted in exactly one position.

Compatible rotary switches with various actuator shapes and lengths, as well as metric and imperial mounting holes and threads, are available from multiple vendors:

C&K A112.**.*.NC.** E-Switch KC52.*.***.**.NP.*

LORLIN CK1049/CK1059 (metric), CK1044/CK1054 (imperial)

(these suggestions should not be considered a commercial endorsement)

Make sure to order the 1x12 variation, as there are also 2x6, 3x4 and 4x3 models. Most switches come with a metal keyring that can be inserted before tightening the screw and that will limit the range of movement to less than 12 steps; so if you only need, say, 4 boot choices including the GRUB menu option, the range of movement can be fixed to 4 steps.

Binary mode

The Custom PCB isn’t designed to hold a binary-encoding switch, but binary mode can be selected anyway by connecting the solder bridge marked with “Binary mode”. The numbers 1, 2, 4, 8 inside the round silhouette indicate the pads to connect to GND to choose the corresponding power of 2. GND level is available at the pad marked with an octagonal silhoutte.

Write-protect

A fully-equipped board should include a slide switch to enable and disable *write-protect* for the boot configuration file. Two alternatives are:

- *Write-protect* can be enabled permanently by connecting the solder bridge marked with a rectangle next to the WR_PROT label.
- Next ot the solder bridge are pads for a 1x2 pin header, which can be jumpered to enable *write-protect* non-permanently.

Further details about the custom PCB can be found in the full documentation.