The GRUB Switch - Quickstart Guide

Ruediger Willenberg

Version 1.2
May 22, 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

aa

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 a programmable USB board and hardware switches 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`	Pad number 1 is square, not round.
`# O O`	Short the round pads number 5 and 6 twice in
`O O O`	short order to put **Beetle** into *bootloader mode*
`2 4 6`	

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, output fixed to low).

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]
                                       .-----------o=#   #          ////
                                       |             #   #||    ////
                                       |       .---o=#   #||////         [0] Menu
                                       |       |     #   #||
                                       |   .---|---o=#   #
                                       |   |   |     #####               [2]
                                       |   |   |
                                       |   |   |
.--------------------------------------+---+---+-----------------------.
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
|  12  11  10  9   8   7   6   5   4   3   2  GND RST RX  TX  SS  MOSI |
| PD6 PB7 PB6 PB5 PB4 PE6 PD7 PC6 PD4 PD0 PD1         PD2 PD1 PB0 PB2  |
|                                                                      |
|                                                                      |
|=========#                                                            |
| USB     #                                                            |
|         #                   ARDUINO MICRO                            |
| MICRO-B #                                                            |
|=========#                                                            |
|                                                                      |
|                                                                      |
| PC7         PF7 PF6 PF5 PF4 PF1 PF0                         PB3  PB1 |
|  13 3V3 AREF A0  A1  A2  A3  A4  A5         5V0 RST GND VIN MISO SCK |
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
'--------------+---+---------------------------------------------------'
               |   |
               |   |
                 ?
             /WR_PROT
```

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 output fixed to low).

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 from 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.

```
   .-----------------------------------------o [11/0xB]
   |   .-----------------------------------o [10/0xA]
   |   |   .------------------------------o [9]
   |   |   |   .-------------------------o [8]
   |   |   |   |   .--------------------o [7]
   |   |   |   |   |   .----------------o [6]          x-----o
   |   |   |   |   |   |   .------------o [5]         /      |
   |   |   |   |   |   |   |   .---------o [4]       /       |
   |   |   |   |   |   |   |   |   .------o [3]     /        |
   |   |   |   |   |   |   |   |   |   .---o [2]   /         |
   |   |   |   |   |   |   |   |   |   |   .-o [1]           |
   |   |   |   |   |   |   |   |   |   |   |   o [0] Menu    |
   |   |   |   |   |   |   |   |   |   |   |                 |
   |   |   |   |   |   |   |   |   |   |   |   .-------------'
   |   |   |   |   |   |   |   |   |   |   |   |
.--+---+---+---+---+---+---+---+---+---+---+---+-----------------------.
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
|  12  11  10  9   8   7   6   5   4   3   2  GND RST RX  TX  SS  MOSI |
| PD6 PB7 PB6 PB5 PB4 PE6 PD7 PC6 PD4 PD0 PD1         PD2 PD1 PB0 PB2  |
|                                                                      |
|                                                                      |
|=========#                                                            |
| USB     #                                                            |
|         #                    ARDUINO MICRO                           |
| MICRO-B #                                                            |
|=========#                                                            |
|                                                                      |
|                                                                      |
| PC7         PF7 PF6 PF5 PF4 PF1 PF0                         PB3  PB1 |
|  13 3V3 AREF A0  A1  A2  A3  A4  A5         5V0 RST GND VIN MISO SCK |
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
'--------------+---+---------------------------------------------------'
               |   |
               |   |
                 ?
             /WR_PROT
```

BINARY MODE:

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

The Write Protect against modifying boot entry configurations is activated
by tieing/jumpering pin PF6 to GND or PF7 (adjacent output fixed to low).

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 tying the four pins PD1..PC6 [LSB..MSB] to GND/0V
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 interpretes a pin tied to GND as a '1', so all four
connections mean choice 15/0xF, while no connection means 0.

```
                .---------------.
                |               |
             .--+ 2    ---    4 +--.
             |  |     /   \     |  |
             |  | C  | --> |  C +--|-----------.
             |  |     \   /     |  |           |
          .--|--+ 8    ---    1 +--|-------.   |
          |  |  |               |  |       |   |
          |  |  '---------------'  |       |   |
          |  '---------------------|---.   |   |
          |                        |   |   |   |
          '--------------------.   |   |   |   |
                               |   |   |   |   |
.------------------------------+---+---+---+---+-----------------------.
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
|  12  11  10  9   8   7   6   5   4   3   2  GND RST RX  TX  SS  MOSI |
| PD6 PB7 PB6 PB5 PB4 PE6 PD7 PC6 PD4 PD0 PD1         PD2 PD1 PB0 PB2  |
|                                                                      |
|                                                                      |
|=========#                                                            |
| USB     #                                                            |
|         #                    ARDUINO MICRO                           |
| MICRO-B #                                                            |
|=========#                                                            |
|                                                                      |
|                                                                      |
| PC7         PF7 PF6 PF5 PF4 PF1 PF0                         PB3  PB1 |
|  13 3V3 AREF A0  A1  A2  A3  A4  A5         5V0 RST GND VIN MISO SCK |
|  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   |
'--------------+---+-----------+---+-----------------------------------'
               |   |           |   |
               |   |           '---'
                 ?              /BM
             /WR_PROT
```