# Pokémon Automation Sword/Shield Scripts (ver. 20200927)

Discord Server: https://discord.gg/cQ4gWxN

This is the manual for Pokémon Automation's in-house Arduino programs for Pokémon Sword/Shield.

This manual assumes:

1. You already know how to use your Arduino/Teensy.
2. You are somewhat familiar with brianuuuSonic's programs or other similar programs online.
3. You are familiar with the mechanics of:
    a. Den Exploits: frame-skipping, VS-glitch (aka Y-COMM glitch), hosting, etc...
    b. Breeding: step-counts, Masuda Method, flame-body, parent compatibility, etc…

If this is your first time using Arduino for Nintendo Switch, watch this video by brianuuuSonic. This manual is not for beginners and will not teach you how to setup your Arduino/Teensy.

If you have questions, feel free to join our Discord server (linked above) for help.

## Table of Contents

## Program List:

**General Tools:**

- TurboA – Endlessly mash A.
- MultiGameFossil – Revive fossils. Supports multiple saves so you can go afk for longer than 5 hours.
- MassRelease – Mass release boxes of Pokémon.
- SurpriseTrade – Surprise trade away boxes of Pokémon.
- TradeBot – Surprise trade with a code for hosting giveaways.
- ClothingBuyer – Buy out all the clothing in a store.

**Date-Spam Farmers:**

- WattFarmer – Farm watts from a wishing piece beam. (6.9 seconds/fetch, 1 million watts/hour)
- BerryFarmer – Farm berries.
- LotoFarmer – Farm the Loto ID.
- StowOnSideFarmer – Farm the Stow-on-Side items dealer.

**Den Hunting:**

- BeamReset – Reset a beam until you see a purple beam.
- EventBeamFinder – Drop wishing pieces until you find an event den.
- DaySkipperJPN – A day skipper for Japanese date format. (7600 skips/hour)
- DaySkipperEU – A day skipper for EU date format that. (~7500 skips/hour)
- DaySkipperUS – A day skipper for US date format that. (~7100 skips/hour)
- DaySkipperJPN-7.8k – An faster, but less reliable Japanese date skipper. (7800 skips/hour)

**Hosting:**

- DenRoller – Roll den to the N'th day, SR and repeat.
- AutoHostRolling – Roll N days, host, SR and repeat.
- AutoHostAirplane – Softlock Hosting: Host, airplane disconnect, repeat.
- AutoHostFriendSearch – Softlock Hosting: Host, friend-search disconnect, repeat.
- AutoHostSleep – Softlock Hosting: Host, sleep disconnect, repeat.
- AutoHostMultiGame – Run AutoHostRolling across multiple game saves. (Up to 16 dens!)
- FriendDelete – Mass delete/block all those unwanted friends from the above.

**Egg Hatching:**

- EggFetcher / EggFetcher2 – Fetch eggs without hatching them.
- EggHatcher – Hatch eggs from boxes.
- EggCombined – Fetch and hatch eggs at the same time. (maximum efficiency)
- EggSuperCombined – Mass release breedjects, then run EggCombined.

**The Forbidden Programs:**

- FastCodeEntry – Force your way into raids by entering 8-digit codes in under 1 second.
- GodEggItemDupe – Mass duplicate items with the God Egg.

## What are these Programs?

Pokémon Automation (PA)'s programs are a large set of Arduino programs for Pokémon Sword/Shield.

The categories which these programs cover are:

- Date-spam farming: watts, berries, loto ID.
- Den-hunting: date skipping
- Auto-hosting
- Egg hatching
- Other misc. tools

Functionally, there is a lot of overlap with brianuuuSonic's AutoController programs. While the AutoController programs are much more user-friendly for beginners, the programs here (by PA) are more expert-oriented and are focused on speed, reliability, and ease of large-scale unattended automation.

| **Program** | **PA's SwSh Scripts (this package)** | **brianuuuSonic's Autocontroller** |
|---|---|---|
| Day Skipper (JPN)<br>(the original "7k skipper") | 7619 skips/hour | 6500 skips/hour (fast mode)<br>6134 skips/hour (default mode) |
| 7.8k JPN Skipper (experimental) | 7827 skips/hour | |
| Day Skipper (EU) | 7541 skips/hour (year 2020)<br>7533 skips/hour (year 2021) | 5006 skips/hour (fast mode)<br>4786 skips/hour (default mode) |
| Day Skipper (US) | 7180 skips/hour (year 2020)<br>7173 skips/hour (year 2021) | |
| Watt Farming | 6.9 seconds/fetch | 9.1 seconds/fetch |
| Berry Farming | 13.1 seconds/fetch | 16.0 seconds/fetch |
| Loto ID | 18.4 seconds/attempt | 20.5 seconds/ attempt |
| Den Rolling | 16.4 seconds/skip | 18.0 seconds/skip |
| Sustained Egg Hatching<br>(5120 step) | 1500 eggs/day<br>(2 touches/day) | 1300 eggs/day<br>(~6 touches/day) |
| 8-Digit Code Entry | ~0.5 seconds | up to 6.3 seconds |

Performance and efficiency goes beyond the raw benchmarks shown above. For example:

- PA's fossil program can revive fossils across multiple game saves to allow overnight runs without the program finishing and the Switch idling. This allows fossils to be run 24/7.
- PA has combined egg fetching+hatching that is almost as fast as manual hatching. More importantly, it can be run 24/7 with little user planning and intervention.

The programs here also emphasize reliability. For example:

- The 7k day skippers are self-correcting and will recover from errors that would normally cause other day skippers to stop working.
- The auto-hosts have multiple safeguards in place to minimize the chances of several catastrophic failure scenarios which have been observed in other programs:
    - Hard-locking or destroying the den.
    - Unintentional link trading or surprise trading.

## Hardware and Recommendations

If are new to automation and are interested in getting started, you will need some hardware.

**Required Hardware:**

First, you will need a board to run the programs. Any of the following are known to work. You can buy them from other retailers such as Amazon.

- Arduino Uno R3 (https://store.arduino.cc/usa/arduino-uno-rev3)
- Teensy 2.0 (https://www.pjrc.com/store/teensy.html)
- Teensy++ 2.0 (https://www.pjrc.com/store/teensypp.html)

Then you will need a way to connect the board both to your computer and to your Switch dock.

- For Arduino, any USB-A male to USB-B male cable will do. (example)
- For Teensy, you will need a mini-USB male to USB-A male cable or adapter. (example)

**Required for Switch Lite:**

If you have a Switch Lite or you want to use the Arduino/Teensy while undocked, you will also either a USB-A female to USB-C adapter or a USB hub/dock that supports charging:

- USB-A female to USB-C male adapter. (example)
- Portable Dock for Nintendo Switch. (example) – also works for Switch Lite

The adapter will let you connect the Arduino/Teensy directly to the Switch, but you will not be able to charge at the same time. A USB hub or portable dock will let you use the Arduino/Teensy and charge at the same time.

**Optional Hardware:**

If you will be using the FastCodeEntry program or if you are a heavy user in general, then it is strongly recommended to get a USB-A male-to-female cable with a power switch on it. (example)

If you intend to change programs very often and will be running the Switch next to a computer, then you will want a USB switch. (example) These are like USB-only KVM switches. In fact a regular KVM switch will also work. With this, you no longer have to constantly plug/unplug USBs which will eventually wear them down.

### Arduino or Teensy?

As a first time user, you will need to pick a board. Arduino is the most common choice due to price and availability, but Teensy has some major advantages.

**Pokemon Automation recommends Teensy 2.0.**

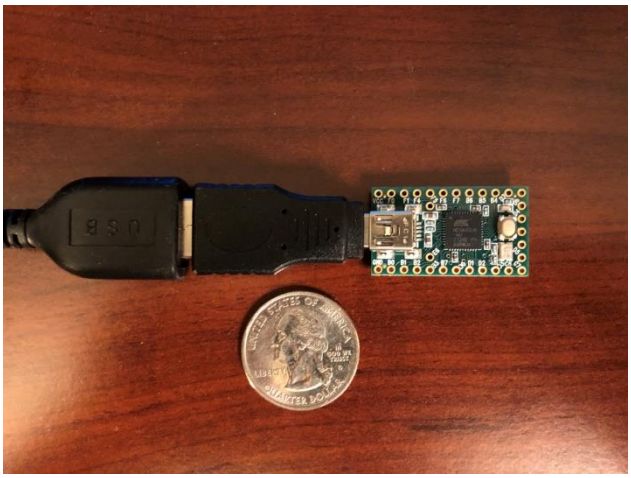
Arduino Uno R3 is ***not*** recommended.

Teensy++ 2.0 is also fine, but overkill in terms of capability.

## Teensy is Easier to Use:

Teensy 2.0 and Teensy++ 2.0 are much easier use. To put them into flash mode, you simply push the white button. By comparison for Arduino, you need to short two pins with a metal object. If you change programs a lot, this can get very tiring.

Teensy is much smaller than Arduino. When used with an adapter, it effectively becomes a small naked USB flash drive which you can easily plug into the computer or the Switch dock. On the other hand, Arduinos are much larger and will require dealing with an extra cable.



## Arduino Uno R3 has Insufficient Memory:

This is a problem that we have run into recent version of this package. The Arduino Uno R3 runs out of memory.

The Arduino Uno R3 only has 512 bytes of usable memory.

This is because the programs for Switch automation run on the ATmega16U2 USB controller chip instead of the larger (removable) ATmega328P chip. Or to put it another way, all Switch automation programs that run on Arduino Uno R3 are hacks that are improperly using the Arduino.

In short, 512 bytes of memory is a very low limit. Prior to version 20200911, the AutoHostMultiGame program overran this limit. This caused the program to crash for Arduino users. While this has since been rectified with memory optimizations, it is only a short-term fix.

Future versions of this package may include even bigger and more complicated programs that will require more than 512 bytes of memory. Such programs will not be able to run on Arduino Uno R3. Thus if you want to future proof yourself, get Teensy instead of Arduino.

Teensy 2.0 and Teensy++ 2.0 have 2.0KB and 2.5KB of memory respectively. This is more than enough.

## How to Use

This section assumes you already know what .hex files are as well as how to load them into the Arduino/Teensy.

### Configuring the Programs:

Each of the programs can be configured by editing the respective .c file. This process is the same as brianuuuSonic's AutoController v3.1.0 programs.

For example, if you want to run the 7k date-skipper to 5000 skips, open up "DaySkipperJPN_7k.c" in a text editor. Edit the "skips" variable to 5000. Then save the file.

```
100     //
101     //   2. Trapping Error: This is a more serious error that eventually leads to
102     //      the program getting stuck toggling the time sync. These are not
103     //      self-recoverable and will only be fixed when the periodic auto-recovery
104     //      routine is run. Trapping errors typically cause the program to fall
105     //      hundreds of skips short since all skips that are supposed happen while
106     //      it is "trapped" are dropped.
107     //
108     //  If the hour, the month, or the timezone has changed, it means that the
109     //  program made a trapping error and recovered from it.
110     //
111
112
113     //  The maximum number of skips to perform.
114     //  The actual # of skips will be lower if any errors are made.
115     const uint32_t skips = 200000;
116
117
118
119     ////////////////////////////////////////////////////////////////////////////////
120     ////////////////////////////////////////////////////////////////////////////////
121     //  Advanced Settings (you shouldn't need to change these)
122
123     //  Run auto-recovery every this # of skips.
124     //  At 500, the overhead is approximately 0.5%.
125     const uint16_t CORRECTION_SKIPS      =   500;
126
127
```

## Building the Hex Files:

There are several ways to build .hex files that you need to load into the Arduino/Teensy.

### Method 1: Run the BuildAll-xxxx.cmd file. (Windows only)

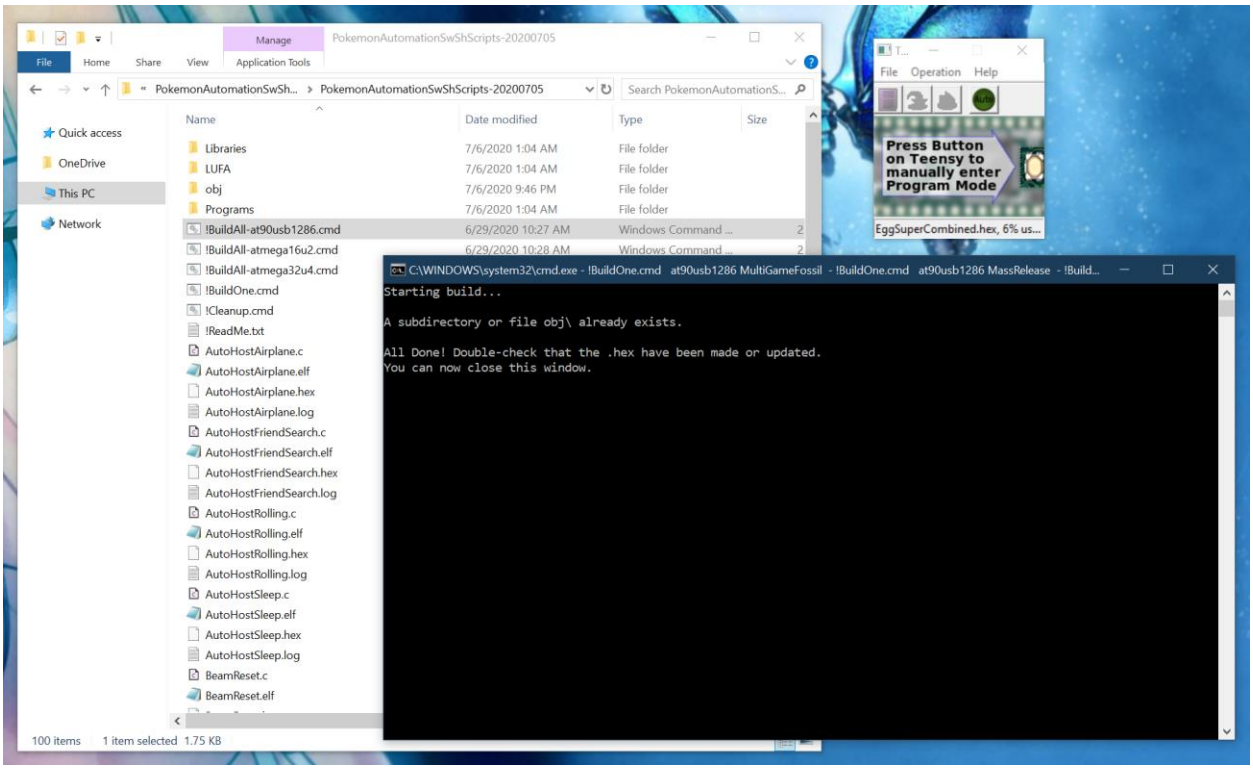
This method only works for Windows users. Mac and Linux users must use Method 2.

First, you must install WinAVR. (https://sourceforge.net/projects/winavr/files/)

There are then 3 BuildAll scripts corresponding to the 3 different MCU types.

- `!BuildAll-at90usb1286.cmd` (for Teensy 2.0++)
- `!BuildAll-atmega16u2.cmd` (for Arduino UNO R3)
- `!BuildAll-atmega32u4.cmd` (for Arduino Micro, and Teensy 2.0)

If you double-click on it, it will build every single program in the package. It should look similar to this:




Depending on how fast your computer is, it may take several seconds or longer to run. Once complete, it will have created .hex files for all the programs. It will tell you if there are any errors.

Errors are logged into the respective .log file for the program (for example “WattFarmer.log”). If you don’t see any .log files, it’s because you have file extensions disabled. In that case, the file will just be named “WattFarmer”.

If you are comfortable with shell scripting and you don’t want to keep rebuilding all the programs, you can edit “Scripts/BuildAll.cmd” to build only a subset of the programs.

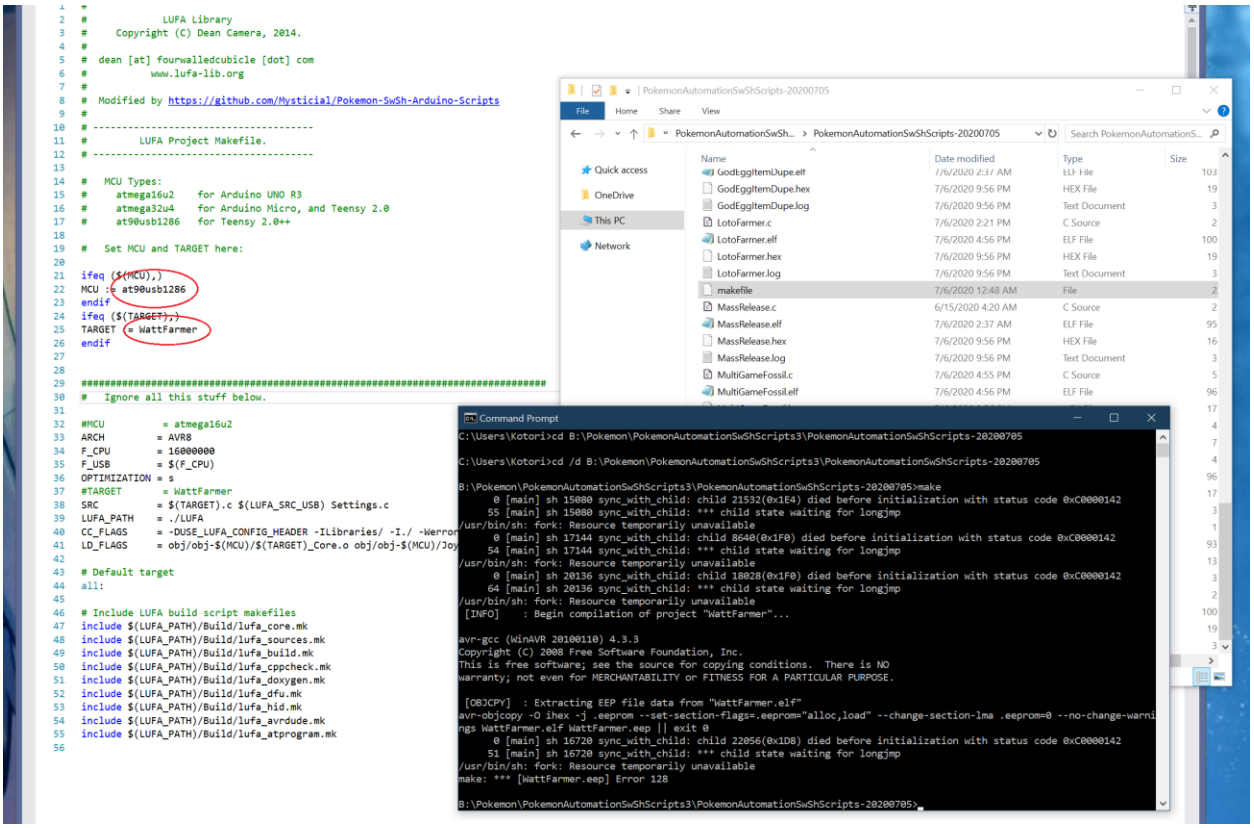
## Method 2: Use the makefile.

This method is almost identical to brianuuuSonic's AutoController v3.1.0 programs. Edit the makefile and run "make" to build the .hex file.

This method does not require Windows and will work for Linux and Mac if you have the right environment setup.

- **Windows:** You again must have WinAVR installed. (https://sourceforge.net/projects/winavr/files/)
- **Mac:** First install homebrew. Then run "brew tap osx-cross/avr" to install AVR.

Open the file "makefile" and edit the "MCU" and "TARGET" manually. Then save the file and run "make" from the command line.



## Load the Program:

Once you have built the .hex file, you must load it into your Arduino or Teensy using FLIP or Teensy Loader respectively.

Refer to brianuuuSonic's video for how to do this as he explains it fairly well for both Arduino and Teensy on Windows.

For Arduino, you will need FLIP: https://www.microchip.com/developmenttools/ProductDetails/flip

For Teensy, you will need Teensy Loader: https://www.pjrc.com/teensy/loader.html

## Run the Program:

On the Switch, navigate to the starting location for the program. Most programs that enter the game will start from the Grip menu. Others will start elsewhere. Refer to the appropriate section in this manual for each program.

Nearly all programs will start with a 5 second delay during which the LEDs will flash for 5 seconds before the program begins running. These 5 seconds give time for KVM setups to switch USB outputs from the computer to the Switch. This can be configured in Settings.c.

If the LEDs do not flash (stays on or off), it means that the device has failed to connect to the Switch.

Some programs will finish, others will run indefinitely. Programs that finish will turn on the LEDs on the Arduino/Teensy to indicate that it is done.

All programs can be safely interrupted at any time by simply unplugging or turning off the Arduino/Teensy. In the end, the Arduino/Teensy is just a controller – one that doesn't fit in your hand as well as a normal controller or a joy con would.

## While a Program is Running:

When running these Arduino programs, do not do anything that might slow down or jitter the UI or graphics. Many of these programs are very optimized and timing-sensitive. Thus, any unexpected delays or slowdowns in the Switch can cause button presses to be missed and/or mis-timed.

**While a program is running:**

- Do not dock/undock the Switch.
- Do not plug/unplug the HDMI cable.
- Do not switch video inputs on the TV.
- Do not turn on/off the TV.
- Do not plug/unplug the AC power.
- Do not let the Switch overheat. Keep it well ventilated, do not lie it flat on a flat surface since it blocks the vents on the back.

All of these actions will jitter the Switch and introduce timing glitches.

**Recommendations:**

- If you don't want to leave the screen on, dim it all the way down before you start the program.
- Alternatively, use a throw-away monitor/TV that you don't care about.
- Alternatively, start the program with the monitor/TV off or disconnected. All programs that finish will turn on the LEDs to indicate it is done.
- If you absolutely need to mess with the displays while the program is running, do it during long animations or when no buttons are being pressed. (i.e. when an egg is hatching, or when a raid is starting)

The worst time to mess with the displays is during settings navigation.

For programs with fast button presses, it is also recommended to disconnect from the internet and turn on airplane mode if applicable. This reduces network traffic-induced jitter.

## History

This project was originally started by Mysticial back in March 2020 as a stand-alone project to add a bunch of functionality that was missing from v3.1.0 of brianuuuSonic's AutoController programs. Since then, the project has been handed over to the Pokémon Automation discord server which now continues its development. Mysticial, the author of first few programs remains part of the development team – but only as a secondary/advisory role.

As of this writing, Pokémon Automation's Arduino programs have been developed independently. With the exception of the LUFA library and the low-level USB descriptors, this project shares no code with and bears no resemblance to any other Arduino Switch programs.

## License

# TurboA

TurboA does exactly what its name implies - it endlessly presses A. The purpose of this program is to grind the Digging Duo and Cram-o-matic.

## Instructions:

- Casual mode is off.
- You are standing in front of what you want to mash A in front of.
- Start the program in the Change Grip/Order Menu.

## Required Parameters:

- None. It is safe to run this program as is without changing anything.

Once the program starts, it will enter the game and start mashing A at approximately 14 times/second.

Stand in front of either the Digging Duo. For the Cram-o-matic, make sure you have already selected your ingredients. When running this on the Cram-o-matic, you need to keep an eye to make sure you stop the program when you run out of ingredients. Otherwise, it will keep churning through different items depending on what you have next in your inventory.

**Useful Tips:**

- It takes about 16 hours and 3 million watts to dig the common fossils from 0 -> 960+.
- Rare fossils are 1/5'th the rarity of the common fossils.

## Options:

### Dodge the System Update Window:

```
const bool DODGE_SYSTEM_UPDATE_WINDOW   =   true;
```

When set to true, the program will attempt to dodge the system update window if it's there. If it's not there, the button presses dodge the window may land inside the game instead. This is harmless in most cases, but is worth keeping in mind when using this program for other things.

## MultiGameFossil

MultiGameFossil is a more advanced fossil reviving program that can revive fossils across multiple game saves. Nevertheless, this program can still be used for a single game only.

### Motivation:

The purpose of a multi-game fossil program is to allow you to go afk for a longer period of time.

Reviving fossils runs at approximately one revive/18.7 seconds. This comes out to about 5 hours for a full game of 960. Unfortunately, 5 hours is not enough to last overnight or a work day. Thus in order to prevent idling and to increase the efficiency of fossils reviving, you must use multiple game saves to get past the 32 box limit of 960 revives.

Thus with 2 games with empty boxes, you can run fossils for 10 hours continuously without needing any attention. This will be enough to last overnight and maybe a day of work. Using even more games will give more flexibility when you aren't able to revive 960 fossils at a time.

### Instructions:

- All participating saves must have casual mode off.
- All participating saves must have their text speed set to fast.
- All participating saves must have "Send to Boxes" set to "Automatic".
- All participating saves must have nicknaming off.
- All participating saves must have at least one of every fossil.
- All participating saves must have the Pokédex entry for the Pokémon they are reviving.
- All participating saves must have spoken to the professor before.
- All participating saves must be saved in front of the professor.
- Each participating save must have enough fossils to run its specified batch.
- If you are reviving from one game version only (Sword or Shield), it must be the 1st slot in the games list.
- If you are reviving from both Sword and Shield, they must be in slots 1 and 2 of the games list.
- You must have no games running.
- Start the program in the Change Grip/Order Menu.

If starting the game requires checking the internet (because it is digital on a non-primary Switch), you will need to open up Settings.c and change `START_GAME_REQUIRES_INTERNET` to true.

Once started, the program will iterate the list specified by "`BATCH`" in the order they are listed. For each entry, it will enter the game and revive the specified fossil the specified number of times. Once it has finished reviving in that game, it will save the game and move on to the next game. If it is the last entry in the list, it will not save the same and will return to the Switch Home to idle.

### Required Parameters:

- `BATCH`: You must set "BATCH" to the desired list of games that you wish to use along with which fossil to revive and how many.

## Safety Recommendations:

- Because this program involves entering games, there is a chance that an error can cause it to enter the wrong game. If any of the games are holding dens, it is recommended to save them facing away from the den. The button presses for reviving fossils can easily destroy a den.

## Single Game Usage:

This is the simplest use case and is what is recommended for first-time users.

Example:

```
const Batch BATCH[] = {
    {1, 3, Arctozolt, 960},

    //  DO NOT DELETE THIS
    {0},
};
```

In the example above, the program will enter the 1st game using the 3rd user profile and revive 960 Arctozolts.

More about user profile #'s.

Once it is done, the program will return to the Switch Home to idle. The game will not be saved. Check if you have a shiny (see below), and reset accordingly if not.

## Multi-Game Usage:

This is the more complicated usage scenario. It will require the use of backup saves.

```
const Batch BATCH[] = {
    {1, 3, Arctozolt, 960},
    {2, 6, Dracovish, 960},

    //  DO NOT DELETE THIS
    {0},
};
```

In this example, the program will:

1. Enter the 1st game as user #3, revive 960 Arctozolts, and save the game.
2. Enter the 2nd game as user #6, revive 960 Dracovishes, and return home without saving.

This program will take about 10 hours to run.

The 1st parameter is the game slot position at the start of the program. It must be 1 or 2. If you are only using one version of the game (Sword or Shield, but not both), then set these all to 1.

When reviving from both Sword and Shield, the games will swap position each time the program starts the 2nd game. This program knows this and will keep track of their positions. Thus, the slot that you specify is when the program starts, not the slot it will be at when the program reaches that point.

## Checking if you have Revived a Shiny:

Once you have revived a ton of fossils, you need to check if there are any shinies. There are generally 3 ways to do this:

1. Manually check them one-by-one.
2. Check your Pokédex for the shiny sprite.
3. Save the game, enter Pokémon Home, and filter by shiny.

#1 is pretty self-explanatory. Seriously, don't do this. It's time-consuming and error-prone. You won't want to miss a shiny because you're impatient and were checking them too quickly.

#2 is a very fast way to check if you have a shiny, but it only works if you don't already have the Pokédex entry for it. Thus it's only good for the 1st time you revive the shiny. If you don't have the shiny sprite in the Pokédex before running the program, but you do afterwards, then you have revived a shiny!

#3 will always work, but requires Pokémon Home and the use of backup saves. Save the game if you haven't already, then immediately enter Pokémon Home. Pokémon Home lets you filter by shiny – thus you can quickly check if you have revived any shinies. If not, you need to close Home without saving and reload the backup save to recover the fossils.

## Managing Backup Saves:

When reviving over multiple game saves, all but the last game will be saved. This means that the fossils will have been consumed. In order to recover them, you must use backup saves.

If you don't know what backup saves or how to use them, look it up online. We won't re-iterate it here. But as a reminder, you can load a backup save by pressing UP+X+B on the title screen.

The idea here is to hard-save in front of the professor before running the program. That way when you reload a backup save, you will reload the game from just before the reviving began. If you don't hard-save, you will reload further back – most likely the fly point in front of the Route 6 tent since flying will overwrite the backup save. This is fine too so hard-saving isn't really necessary.

Hard-saving can be done either by trading, or saving the game through Pokémon Home.

Common Pitfalls:

- If you forgot to load the backup save and thus have a game full of non-shiny fossils, DO NOT fly! Flying will overwrite the backup save causing you to lose all the fossils that were used.
- When exiting Pokémon Home, you MUST select "exit without saving changes". If you exit saving changes, it will hard-save and wipe the backup save even if you made no changes!

## MassRelease

MassRelease will release entire boxes of Pokémon. The main use case is getting rid of breedjects and non-shiny fossils.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must in the box system.
- The cursor must be over the 1st Pokémon in the box. (top-left corner)
- The cursor must be red. (not blue or green)
- Start the program in the Change Grip/Order Menu.

Once started, this program will release the specified number of boxes consecutively starting from the current box.

### Box Setup:

- Place entire boxes filled with Pokémon to be released consecutively.
- The last box must be full. This program does not tolerate partial boxes.

### Required Parameters:

- `BOXES_TO_RELEASE`: You must specify the number of boxes to release.

### Safety Recommendations:

See Maximizing Switch Stability.

### Options:

#### # of Boxes to Release:

```
const uint8_t BOXES_TO_RELEASE  =   2;
```

This is the number of boxes to release.

#### Dodge the System Update Window:

```
const bool DODGE_SYSTEM_UPDATE_WINDOW   =   false;
```

When set to true, the program will dodge the system update window. Do not set this option if the system update window is not present. Unlike other programs, MassRelease will not be able to tolerate the extra button presses if they land in the box system.

# SurpriseTrade

SurpriseTrade will surprise trade entire boxes of Pokémon. The main use case is gathering trainer IDs for Loto farming and to farm hacked Pokémon+items surprise trades.

This program is similar to brianuuuSonic's BoxSurpriseTrade, but is also tolerant to errors when changing boxes.

## Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Nicknaming needs to be off. (in case of a trade evolution)
- If you are trading online, make sure you have a strong internet connection.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- You should be backed into a corner where you cannot travel down or right.
- Your location should be safe from getting attacked by wild Pokémon.
- If you wish to surprise trade online, be connected to the internet before you start the program. Thus you will need to be fast when starting the program or you will be disconnected.
- Start the program in the Change Grip/Order Menu.

Once started, this program will surprise trade the specified number of boxes consecutively starting from the current box.

When the program reaches the end of a box, it will pause for a longer period before moving the next box. This is to ensure that any errors from a preceding trade will have cleared out.

## Box Setup:

- Place boxes* of Pokémon to be surprise traded consecutively.
- You must be on the first box.
- Do not have any untradable Pokémon in these boxes. (no eggs or fused Pokémon)

*The boxes do not need to be full. Empty slots are safely skipped.

## Safety Recommendations:

- It goes without saying that this is a very dangerous program. Don't run this program unattended unless the game is completely free of Pokémon you want to keep.

## Corner Cases:

**Known Corner Cases:**

- If a box slot is empty, it will be safely skipped.
- If a trade takes too long, the program will self-recover.*
- If no trading partner is found, the program will self-recover.*

- If a trade results in a new Pokedex entry, the program will self-recover.*
- If a trade results in an evolution, the program will self-recover.*
- If a trade results in an evolution and a new move, the program should recover*, but it may take more than one loop.
- This program will tolerate the above errors even at the end of a box.

*In these cases, self-recovery will mean skipping the next trade.

**Untested Corner Cases:**

- Untradable Pokémon: eggs, fused, illegal, etc...
- You get disconnected from the internet while the program is running.

## Options:

### # of Boxes to Trade:

```
const uint8_t BOXES_TO_TRADE    =   2;
```

This is the number of boxes to trade away.

### Wait for Partner Delay:

```
const uint16_t INITIAL_WAIT     =   30 * TICKS_PER_SECOND;
```

Wait this long for a partner to be found before continuing.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Misc. Timings:

```
const uint16_t TRADE_ANIMATION  =   23 * TICKS_PER_SECOND;
const uint16_t EVOLVE_DELAY     =   30 * TICKS_PER_SECOND;
```

These are pretty self-explanatory. You shouldn't need to change them.

## TradeBot

TradeBot is a coded version of SurpriseTrade. The main use case is for hosting giveaways.

The program can sustain a trade every ~103 seconds. Thus it takes almost an hour to process a box.

This program is experimental and is more suited for high-demand and high-quantity giveaways. As with SurpriseTrade, this program will skip Pokémon if no trade happens. Thus there will be many untraded Pokémon. Likewise, you will need many Pokémon to sustain a giveaway that lasts for a long period of time unattended.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Nicknaming needs to be off. (in case of a trade evolution)
- If you are trading online, make sure you have a strong internet connection.
- The cursor in the menu is over the "Pokémon" option.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- You should be backed into a corner where you cannot travel down or right.
- Your location should be safe from getting attacked by wild Pokémon.
- If you wish to trade online, be connected to the internet before you start the program. Thus you will need to be fast when starting the program or you will be disconnected.
- Start the program in the Change Grip/Order Menu.

Once started, this program will trade the specified number of boxes consecutively starting from the current box.

When the program reaches the end of a box, it will pause for a longer period before moving the next box. This is to ensure that any errors from a preceding trade will have cleared out.

### Box Setup:

- Place boxes* of Pokémon to be traded consecutively.
- You must be on the first box.
- Do not have any untradable Pokémon in these boxes. (no fused Pokémon)

### Safety Recommendations:

- It goes without saying that this is a very dangerous program. Don't run this program unattended unless the game is completely free of Pokémon you want to keep.

*The boxes do not need to be full. Empty slots are safely skipped, but you will leave the partner waiting for a long time before disconnecting them without trading anything.

## Corner Cases:

**Known Corner Cases:**

- If no trading partner is found, no trade happens and the current Pokémon is skipped.
- If the trade partner is too slow to pick a Pokémon or confirm the trade, the trade will be canceled.
- If the trade partner cancels at any time, the program should back out without issues.
- If a trade results in an evolution, the program will self-recover.*
- If a trade results in an evolution and a new move, the program should recover*.
- This program will tolerate the above errors even at the end of a box.

*In these cases, self-recovery will mean skipping the next trade.

**Untested Corner Cases:**

- If a box slot is empty, it should be safely skipped.
- If a trade results in a new Pokedex entry, the program should self-recover.
- Untradable Pokémon: fused, illegal, etc...
- You get disconnected from the internet while the program is running.

## Options:

### Trade Code:

```
const char* TRADE_CODE          =   "1280 0000";
```

Pretty self-explanatory.

The program will skip non-digit characters. So the space or hyphen separator is optional.

### # of Boxes to Trade:

```
const uint8_t BOXES_TO_TRADE    =   2;
```

This is the number of boxes to trade away.

### Search Delay:

```
const uint16_t SEARCH_DELAY     =   20 * TICKS_PER_SECOND;
```

Wait this long to search for a trade partner before canceling (if no partner found) or proceeding to trade.

### Confirm Trade Delay:

```
const uint16_t CONFIRM_DELAY    =   10 * TICKS_PER_SECOND;
```

Once you select a Pokémon to trade, give the user this long to confirm the trade before you cancel the trade.

### Internet Connection Delays:

```
const uint16_t TRADE_START          =   10 * TICKS_PER_SECOND;
const uint16_t TRADE_COMMUNICATION  =   20 * TICKS_PER_SECOND;
```

Increase these if your internet connection is slow.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Misc. Timings:

```
const uint16_t TRADE_ANIMATION  =   23 * TICKS_PER_SECOND;
const uint16_t EVOLVE_DELAY     =   30 * TICKS_PER_SECOND;
```

These are pretty self-explanatory. You shouldn't need to change them.

## ClothingBuyer

As its name implies, this program will automatically buy clothing.

Note that this program is not particularly efficient. But it will properly handle items that have already been purchased and will eventually clear an entire store regardless of how many items there are in each category.

Preliminary testing suggests that this program takes an hour to buy out the entire Wedgehurst clothing store.

Be aware that this program will change your clothing. So you may need to fix it when you're done.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You are in the menu for buying clothing. (Anywhere is fine.)
- Start the program in the Change Grip/Order Menu.

Note that this program does not stop since it does not know when it is done. Therefore you must stop it manually and check to see if it there is anything left to buy.

### Options:

#### Rotate Categories:

```
const bool CATEGORY_ROTATION            =   true;
```

When set to true, the program will change categories. This allows the program to eventually buy out the entire store, but will make it less efficient when there are categories that are already bought out.

#### Dodge the System Update Window:

```
const bool DODGE_SYSTEM_UPDATE_WINDOW   =   true;
```

When set to true, the program will attempt to dodge the system update window if it's there.

## WattFarmer

WattFarmer will farm watts from a wishing piece beam. It requires activating the VS glitch.

This program runs at 6.9 seconds per fetch. This is about 1 million watts/hour at 2000/fetch.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- You must be offline.
- Airplane mode must be off.
- You must be standing in front of a wishing piece den with watts collected.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

This program will not work if the system update window is present. Instead, it will safely enter a do-nothing loop within the settings.

### Stability Recommendations:

Stand *behind* the den so that the beam is directly in front of your character.

Sometimes, the program will miss a button press which causes the date-spamming to happen in the game instead of the Switch settings. This will cause the character to move downwards and away from the den if you're not standing behind it.

### Options:

#### Fetch Attempts:

```
const uint32_t SKIPS            =   33334;
```

Perform this many fetch attempts. The only reason to set this number is if you're also skipping to a den frame and you don't want to overshoot. But be aware that this program isn't intended to be an accurate date skipper. It may occasionally miss skips which can cause it to fall short.

#### Periodically Save the Game:

```
const uint16_t SAVE_ITERATIONS  =   0;
```

If set to anything other than zero, the program will periodically save the game at the specified interval. In rare cases, the date-skipping can cause the game to crash, thus periodic saves can be enabled to reduce the amount of work that is lost.

## BerryFarmer

BerryFarmer will farm berries from a tree. It requires activating the VS glitch.

This program runs at 13.1 seconds per fetch.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- You must be standing in front of a berry tree.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

This program will not work if the system update window is present. Instead, it will safely enter a do-nothing loop within the settings.

### Stability Recommendations:

Stand *behind* the berry tree so that the tree is directly in front of (and blocking) your character.

Sometimes, the program will miss a button press which causes the date-spamming to happen in the game instead of the Switch settings. This will cause the character to move downwards and away from the tree if you're not standing behind it.

### Options:

#### Fetch Attempts:

```
const uint32_t SKIPS            =   100000;
```

Perform this many fetch attempts. The only reason to set this number is if you're also skipping to a den frame and you don't want to overshoot. But be aware that this program isn't intended to be an accurate date skipper. It may occasionally miss skips which can cause it to fall short.

#### Periodically Save the Game:

```
const uint16_t SAVE_ITERATIONS  =   0;
```

If set to anything other than zero, the program will periodically save the game at the specified interval. In rare cases, the date-skipping can cause the game to crash, thus periodic saves can be enabled to reduce the amount of work that is lost.

## LotoFarmer

LotoFarmer will farm the loto. It requires activating the VS glitch.

This program runs at 18.4 seconds per attempt.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- You must be standing in front of the PC.
- Start the program in the Change Grip/Order Menu.

This program will not work if the system update window is present. Instead, it will safely enter a do-nothing loop within the settings.

### Options:

#### Attempts:

```
const uint32_t SKIPS    =   100000;
```

Perform this many loto attempts. The only reason to set this number is if you're also skipping to a den frame and you don't want to overshoot. But be aware that this program isn't intended to be an accurate date skipper. It may occasionally miss skips which can cause it to fall short.

## StowOnSideFarmer

Farm the Stow-on-Side bargains dealer. It requires activating the VS glitch.

This program runs at 13.9 seconds per purchase.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- You must be standing in front of the Stow-on-Side bargains dealer.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

This program will not work if the system update window is present. Instead, it will safely enter a do-nothing loop within the settings.

### Options:

#### Fetch Attempts:

```
const uint32_t SKIPS            =   100000;
```

Perform this many purchases. The only reason to set this number is if you're also skipping to a den frame and you don't want to overshoot. But be aware that this program isn't intended to be an accurate date skipper. It may occasionally miss skips which can cause it to fall short.

#### Periodically Save the Game:

```
const uint16_t SAVE_ITERATIONS  =   0;
```

If set to anything other than zero, the program will periodically save the game at the specified interval. In rare cases, the date-skipping can cause the game to crash, thus periodic saves can be enabled to reduce the amount of work that is lost.

## BeamReset

Automatically reset a den beam. The purpose is for finding purple beams.

Drop a wishing piece in, check if it's red or purple. Pause, then reset.

### Instructions:

- Casual mode is off.
- Your text speed must be set to SLOW.
- You must stand in front of an empty den with watts collected.
- You must be facing the center of the den.
- You must be saved in the above position.
- Start the program in the Change Grip/Order Menu.

On each beam, the program will go in-and-out of the game several times to flash the beam and grab your attention. Then it will wait in the Home menu for 5 seconds before it resets the game. If you like the beam, you must stop the program during these 5 seconds.

Depending on your Switch, the weather, the type of beam (red or purple), and your alignment with the den, you may not see a beam immediately. In some cases, you may not see a beam for several in-and-outs.

Red beams will show up faster. Purple beams are slower and will appear after the text starts showing up. If you see no beam before the program returns to the home menu the last time, it is likely a purple beam.

### Options:

#### Delay Before Reset:

```
const uint16_t DELAY_BEFORE_RESET = 5 * TICKS_PER_SECOND;
```

Wait in the Home menu for this long before resetting the game.

## EventBeamFinder

Drop wishing pieces between two dens until you find an event den.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Airplane mode must be enabled*. To run docked, undock the Switch, turn on airplane mode, then redock. The Switch will remain in airplane mode after redocking.
- The two dens closest to the Bridge Field daycare must not have any natural beams. Wishing piece beams are okay since they will be despawned.
- The cursor in the menu is over the Town Map.
- You must be in Bridge Field.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

The program will travel back-and-forth between the two dens closest to the Bridge Field daycare center. Each time, it will drop a wishing piece and enter the den to let you view it.

*Airplane mode is required to avoid getting stuck inside a den as well as slowdowns from network traffic/interference.

This program goes back-and-forth between the following two dens:

- **Den 5/46 (Shedinja):** By the trees near the Obstagoon.
- **Den 4/92 (Gmax Orbeetle):** To the right of the Digging Duo.

### Options:

#### Delay Before Reset:

```
const uint16_t WAIT_TIME_IN_DEN = 5 * TICKS_PER_SECOND;
```

View the den for this duration before moving on.

## DaySkipperJPN

This is the original “7k Skipper”. It is a Japanese day skipper that runs at 7619 skips/hour.

When the program finishes skipping, it will continue to enter/exit the date change menu every 15 seconds without changing the date. This prevents the time from advancing naturally and potentially passing midnight which would result in an extra (unintended) day skip.

### Instructions:

- The language for your Switch must be set to Japanese (日本語).
- Set the date to the 1st of a month with 31 days that is not March.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- Make sure you are in a Pokémon center. (The Isle of Armor dojo is not reliable!)
- Start the script anywhere in the Date/Time menu except the time zone menu. It will automatically find the right place.

Most people use the following workflow:

1. Undock the Switch if it is docked.
2. Change the Switch language to Japanese.
3. Set the date to the 1st of a month with 31 days that is not March.
4. Enter the game and go to a Pokémon center.
5. Connect to the internet and start a link battle.
6. When someone is found, turn on airplane mode and keep it on.
7. Mash B to clear all the messages.
8. Confirm that the VS glitch is active by leaving and re-entering the game. The screen should flash when you re-enter the game.
9. Go into settings and date time.
10. Dock the Switch if desired.
11. Start the program.

If the game is a digital copy and the Switch is not the primary, you will not be able to turn on airplane mode for longer skips that take multiple hours. This is because the Switch will periodically check online to see if it’s allowed to keep running the game. Airplane mode prevents this which causes the game to halt.

If this applies to you, either make the Switch the primary, or don’t enable airplane mode. If you opt for the latter, you can expect the day skipper to make a lot more errors.

### Stability Recommendations:

See Maximizing Switch Stability.

This is very important for date skipping. Airplane mode increases stability!

### Common Usage Errors: (Human Errors)

Be careful when setting the starting date ***after*** you activate the Y-COMM glitch. Skipping forward will cause an extra skip to happen.

Do not run this day skipper in the Wild Area even if it is for very few skips where you don't expect to crash the game. Skipping in the Wild Area has been observed to miss LOTs of frames.

Do not run this day skipper in the Isle of Armor Dojo. Apparently, it is not "indoor enough" and may miss skips on some Switches.

## How many Skips are Remaining?

Throughout the skipping process, the program will turn the Arduino/Teensy's LEDs on and off to indicate roughly how many skips are remaining. The faster it flickers the lights, the fewer skips are left.

| **Skips Remaining** | **Flash Interval (seconds)** |
|---|---|
| 0 | LEDs always on |
| < 20 | 1/32 |
| < 50 | 1/16 |
| < 100 | 1/8 |
| < 200 | 1/4 |
| < 500 | ½ |
| < 1,000 | 1 |
| < 2,000 | 2 |
| < 5,000 | 3 |
| < 10,000 | 4 |
| < 20,000 | 5 |
| < 50,000 | 6 |
| < 100,000 | 7 |
| < 200,000 | 8 |
| < 500,000 | 9 |
| >= 500,000 | 10 |

## Errors and Fault Tolerance:

See Day Skipper Errors and Auto-Corrections.

As with all fast day skippers, this skipper is not 100% reliable. It can make errors which cause it to fall short of the target number of skips. See the linked section for more details.

## Options:

### Frame Skips:

```
const uint32_t skips = 10;
```

Skip this many frames. The actual number of skips performed may be less if any errors were made.

This can be as large as 4,294,967,295. But it will take 60+ years to skip that many frames.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Auto-Recover Interval:

```
const uint16_t CORRECTION_SKIPS      =   1000;
```

Perform an auto-recovery every this number of skips. This is the recovery routine that recovers from trapping errors. You shouldn't need to touch this value.

## DaySkipperEU

This is the EU day skipper that runs at 7350 - 7550 skips/hour.

The exact speed will depend on the real life year. For the year 2020, it will run at about 7541 skips/hour.

Note that this skipper is slower than the Japanese date skipper. If you are doing large skips, it is recommended to change the Switch language to Japanese and use the Japanese skipper.

When the program finishes skipping, it will continue to enter/exit the date change menu every 15 seconds without changing the date. This prevents the time from advancing naturally and potentially passing midnight which would result in an extra (unintended) day skip.

### Instructions:

- The language/region for the Switch must have a date layout of dd/mm/year/hour/min.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- Make sure you are in a Pokémon center. (The Isle of Armor dojo is not reliable!)
- Start the script anywhere in the Date/Time menu except the time zone menu. It will automatically find the right place.

Unlike the JPN date skipper, you do not need to set the starting the date. However you do need to set the current year correctly.

Most people use the following workflow:

1. Undock the Switch if it is docked.
2. Change the Switch region to EU.
3. Enter the game and go to a Pokémon center.
4. Connect to the internet and start a link battle.
5. When someone is found, turn on airplane mode and keep it on.
6. Mash B to clear all the messages.
7. Confirm that the VS glitch is active by leaving and re-entering the game. The screen should flash when you re-enter the game.
8. Go into settings and date time.
9. Dock the Switch if desired.
10. Start the program.

If the game is a digital copy and the Switch is not the primary, you will not be able to turn on airplane mode for longer skips that take multiple hours. This is because the Switch will periodically check online to see if it's allowed to keep running the game. Airplane mode prevents this which causes the game to halt.

If this applies to you, either make the Switch the primary, or don't enable airplane mode. If you opt for the latter, you can expect the day skipper to make a lot more errors.

### Stability Recommendations:

See Maximizing Switch Stability.

This is very important for date skipping. Airplane mode increases stability!

## Common Usage Errors: (Human Errors)

Do not run this day skipper in the Wild Area even if it is for very few skips where you don't expect to crash the game. Skipping in the Wild Area has been observed to miss LOTs of frames.

Do not run this day skipper in the Isle of Armor Dojo. Apparently, it is not "indoor enough" and may miss skips on some Switches.

## How many Skips are Remaining?

Throughout the skipping process, the program will turn the Arduino/Teensy's LEDs on and off to indicate roughly how many skips are remaining. The faster it flickers the lights, the fewer skips are left.

| **Skips Remaining** | **Flash Interval (seconds)** |
|---|---|
| 0 | LEDs always on |
| < 20 | 1/32 |
| < 50 | 1/16 |
| < 100 | 1/8 |
| < 200 | 1/4 |
| < 500 | ½ |
| < 1,000 | 1 |
| < 2,000 | 2 |
| < 5,000 | 3 |
| < 10,000 | 4 |
| < 20,000 | 5 |
| < 50,000 | 6 |
| < 100,000 | 7 |
| < 200,000 | 8 |
| < 500,000 | 9 |
| >= 500,000 | 10 |

## Errors and Fault Tolerance:

See Day Skipper Auto-Corrections.

As with all fast day skippers, this skipper is not 100% reliable. It can make errors which cause it to fall short of the target number of skips. See the linked section for more details.

## Options:

### Frame Skips:

```
const uint32_t skips = 10;
```

Skip this many frames. The actual number of skips performed may be less if any errors were made.

This can be as large as 4,294,967,295. But it will take 60+ years to skip that many frames.

### Real Life Year:

```
const uint16_t REAL_LIFE_YEAR   =   2020;
```

Set this to the current year in real life.

More specifically, this needs to be the year that the time-sync will change the date to.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Auto-Recover Interval:

```
const uint16_t CORRECTION_SKIPS      =   500;
```

Perform an auto-recovery every this number of skips. This is the recovery routine that recovers from trapping errors. You shouldn't need to touch this value.

## DaySkipperUS

This is the US day skipper that runs at about 7000 - 7200 skips/hour.

The exact speed will depend on the real life year. For the year 2020, it will run at about 7180 skips/hour.

Note that this skipper is slower and less reliable than the Japanese date skipper. If you are doing large skips, it is recommended to change the Switch language to Japanese and use the Japanese skipper.

When the program finishes skipping, it will continue to enter/exit the date change menu every 15 seconds without changing the date. This prevents the time from advancing naturally and potentially passing midnight which would result in an extra (unintended) day skip.

### Instructions:

- The language/region for the Switch must have a date layout of dd/mm/year/hour/min/AM.
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- Make sure you are in a Pokémon center. (The Isle of Armor dojo is not reliable!)
- Start the script anywhere in the Date/Time menu except the time zone menu. It will automatically find the right place.

Unlike the JPN date skipper, you do not need to set the starting the date. However you do need to set the current year correctly.

Most people use the following workflow:

1. Undock the Switch if it is docked.
2. Enter the game and go to a Pokémon center.
3. Connect to the internet and start a link battle.
4. When someone is found, turn on airplane mode and keep it on.
5. Mash B to clear all the messages.
6. Confirm that the VS glitch is active by leaving and re-entering the game. The screen should flash when you re-enter the game.
7. Go into settings and date time.
8. Dock the Switch if desired.
9. Start the program.

If the game is a digital copy and the Switch is not the primary, you will not be able to turn on airplane mode for longer skips that take multiple hours. This is because the Switch will periodically check online to see if it's allowed to keep running the game. Airplane mode prevents this which causes the game to halt.

If this applies to you, either make the Switch the primary, or don't enable airplane mode. If you opt for the latter, you can expect the day skipper to make a lot more errors.

### Stability Recommendations:

See Maximizing Switch Stability.

This is very important for the date skipper. Airplane mode increases stability!

## Common Usage Errors: (Human Errors)

Do not run this day skipper in the Wild Area even if it is for very few skips where you don't expect to crash the game. Skipping in the Wild Area has been observed to miss LOTs of frames.

Do not run this day skipper in the Isle of Armor Dojo. Apparently, it is not "indoor enough" and may miss skips on some Switches.

## How many Skips are Remaining?

Throughout the skipping process, the program will turn the Arduino/Teensy's LEDs on and off to indicate roughly how many skips are remaining. The faster it flickers the lights, the fewer skips are left.

| **Skips Remaining** | **Flash Interval (seconds)** |
|---|---|
| 0 | LEDs always on |
| < 20 | 1/32 |
| < 50 | 1/16 |
| < 100 | 1/8 |
| < 200 | 1/4 |
| < 500 | ½ |
| < 1,000 | 1 |
| < 2,000 | 2 |
| < 5,000 | 3 |
| < 10,000 | 4 |
| < 20,000 | 5 |
| < 50,000 | 6 |
| < 100,000 | 7 |
| < 200,000 | 8 |
| < 500,000 | 9 |
| >= 500,000 | 10 |

## Errors and Fault Tolerance:

See Day Skipper Auto-Corrections.

As with all fast day skippers, this skipper is not 100% reliable. It can make errors which cause it to fall short of the target number of skips. See the linked section for more details.

## Options:

### Frame Skips:

```
const uint32_t skips = 10;
```

Skip this many frames. The actual number of skips performed may be less if any errors were made.

This can be as large as 4,294,967,295. But it will take 60+ years to skip that many frames.

### Real Life Year:

```
const uint16_t REAL_LIFE_YEAR   =   2020;
```

Set this to the current year in real life.

More specifically, this needs to be the year that the time-sync will change the date to.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Auto-Recover Interval:

```
const uint16_t CORRECTION_SKIPS      =   500;
```

Perform an auto-recovery every this number of skips. This is the recovery routine that recovers from trapping errors. You shouldn't need to touch this value.

## DaySkipperJPN-7.8k

This is an experimental JPN day skipper that runs at 7827 skips/hour. Thus it is faster than DaySkipperJPN.

This skipper is less tolerant to errors compared to the other skippers. Don't use this skipper unless your system is very stable or if you're just experimenting.

This skipper works by looping through all 22,280 days from 2000/1/1 to 2020/12/31 then rewinding and starting over. The speed advantage over the other skippers is that it wastes a lot less time rewinding the year or rolling over the day of the month.

However, this slight increase in efficiency comes at a cost. The 7.8k skipper is less tolerant to errors. Unlike the other skippers in this package, the 7.8k skipper cannot always fully recover from every error.

While the 7.8k skipper can recover from minor errors, major errors will cause it to lose track of the date and thus run at a degraded speed of about 7570 skips/hour – which is slightly worse than the regular JPN date skipper. Likewise, once the program enters this "off-track" state, about 3.3% of the skips will be dropped which will cause the program fall significantly short of the target number of skips.

### Instructions:

- The language for your Switch must be set to Japanese (日本語).
- Set the date to January 1st 2000.
- Make sure the time of the day is not during the DST missing hour. (2 - 3AM)
- You must have system time unsynced.
- The VS (Y-COMM) glitch must be active.
- Make sure you are in a Pokémon center. (The Isle of Armor dojo is not reliable!)
- Start the script anywhere in the Date/Time menu except the time zone menu. It will automatically find the right place.

Most people use the following workflow:

1. Undock the Switch if it is docked.
2. Change the Switch language to Japanese.
3. Set the date to January 1st 2000.
4. Enter the game and go to a Pokémon center.
5. Connect to the internet and start a link battle.
6. When someone is found, turn on airplane mode and keep it on.
7. Mash B to clear all the messages.
8. Confirm that the VS glitch is active by leaving and re-entering the game. The screen should flash when you re-enter the game.
9. Go into settings and date time.
10. Dock the Switch if desired.
11. Start the program.

If the game is a digital copy and the Switch is not the primary, you will not be able to turn on airplane mode for longer skips that take multiple hours. This is because the Switch will periodically check online to see if it's allowed to keep running the game. Airplane mode prevents this which causes the game to halt.

If this applies to you, either make the Switch the primary, or don't enable airplane mode. If you opt for the latter, you can expect the day skipper to make a lot more errors.

## Stability Recommendations:

See Maximizing Switch Stability.

This is very important for date skipping. Airplane mode increases stability!

## Common Usage Errors: (Human Errors)

Be careful when setting the starting date ***after*** you activate the Y-COMM glitch. Skipping forward will cause an extra skip to happen.

Do not run this day skipper in the Wild Area even if it is for very few skips where you don't expect to crash the game. Skipping in the Wild Area has been observed to miss LOTs of frames.

Do not run this day skipper in the Isle of Armor Dojo. Apparently, it is not "indoor enough" and may miss skips on some Switches.

## How many Skips are Remaining?

Throughout the skipping process, the program will turn the Arduino/Teensy's LEDs on and off to indicate roughly how many skips are remaining. The faster it flickers the lights, the fewer skips are left.

| **Skips Remaining** | **Flash Interval (seconds)** |
|---|---|
| 0 | LEDs always on |
| < 20 | 1/32 |
| < 50 | 1/16 |
| < 100 | 1/8 |
| < 200 | 1/4 |
| < 500 | ½ |
| < 1,000 | 1 |
| < 2,000 | 2 |
| < 5,000 | 3 |
| < 10,000 | 4 |
| < 20,000 | 5 |
| < 50,000 | 6 |
| < 100,000 | 7 |
| < 200,000 | 8 |
| < 500,000 | 9 |
| >= 500,000 | 10 |

## Errors and Fault Tolerance:

See Day Skipper Errors and Auto-Corrections.

As with all fast day skippers, this skipper is not 100% reliable. It can make errors which cause it to fall short of the target number of skips. See the linked section for more details.

## Options:

### Frame Skips:

```
const uint32_t skips = 10;
```

Skip this many frames. The actual number of skips performed may be less if any errors were made.

This can be as large as 4,294,967,295. But it will take 60+ years to skip that many frames.

### Start Date:

```
const Date start_date = {0, 1, 1};  //  {year-2000, month, day}
```

If you don't want to start on January 1st 2000, you can set your start date here.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Auto-Recover Interval:

```
const uint16_t CORRECTION_SKIPS      =   1000;
```

Perform an auto-recovery every this number of skips. This is the recovery routine that recovers from trapping errors. You shouldn't need to touch this value.

## DenRoller

Roll a den forward by N days, show what it is, then reset.

Use this program to roll for a specific Pokémon in your den.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- You must be standing in front of a wishing piece den with watts collected.
- Your location should be safe from getting attacked by wild Pokémon.
- You must be in the overworld.
- You must be saved in the above position.
- Start the program in the Change Grip/Order Menu.

If starting the game requires checking the internet (because it is digital on a non-primary Switch), you will need to open up Settings.c and change `START_GAME_REQUIRES_INTERNET` to true.

### Options:

#### Frame Skips:

```
const uint8_t SKIPS          =   3;
```

The number of frames to roll. The default is 3. But some people may prefer to save high-value dens more than 3 days back for added safety in case of accidental roll-over.

#### Catchability:

```
//  ALWAYS_CATCHABLE    All pokemon in this den are catchable.
//  MAYBE_UNCATCHABLE   There may be uncatchable pokemon.
const Catchability CATCHABILITY     =   ALWAYS_CATCHABLE;
```

Some dens have uncatchable Pokémon (i.e. Mewtwo or Zeraora). If the den has any such uncatchable Pokémon, you must this to MAYBE_UNCATCHABLE.

#### Delay Before Reset:

```
const uint16_t VIEW_TIME     =   5 * TICKS_PER_SECOND;
```

View the rolled Pokémon for this long before resetting.

## AutoHostRolling

This is the big auto-host program that everybody uses. It repeats the following sequence forever:

1. Roll the den forward by N days.
2. Host the Pokémon.
3. Reset the game.

By setting N to zero, this program becomes a hard-lock auto-host. This is the preferred way to host hard-locked dens as opposed to the soft-lock auto-hosts which have more restrictions and are less safe.

**Optional Features:**

- Host offline using local communication instead of online.
- Automatically accept friend requests.
- Add a delay between raids to make farming more efficient when you can't clear a raid before the next one starts.
- Select a first move for the hosting Pokémon.

**Important Warning:**

All auto-hosts carry a risk of destroying the den that is being hosted. The failure case is an error that causes the program to clear the raid and drop a new wishing piece. Dropping a wishing piece is a hard-save and cannot be reversed via backup save.

If you care about your den:

1. Get rid of all your wishing pieces.
2. Do not run the auto-hosts unattended.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- You must have system time unsynced.
- Make sure you have a strong internet connection. (if hosting online)
- You must be disconnected from internet.
- You must be standing in front of a wishing piece den with watts collected.
- Your location should be safe from getting attacked by wild Pokémon.
- You must be in the overworld.
- You must be saved in the above position.
- Start the program in the Change Grip/Order Menu.

If starting the game requires checking the internet (because it is digital on a non-primary Switch), you will need to open up Settings.c and change `START_GAME_REQUIRES_INTERNET` to true.

By default, the program will start the raid at 2:00. Everybody needs to be ready by then or the raid may fail. This can be changed by modifying "`LOBBY_WAIT_DELAY`".

## Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### Raid Code:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1280 0000";
```

See Raid Code Entry.

### Frame Skips:

```
const uint8_t SKIPS             =   3;
```

This option sets how many frames to skip before hosting. Set to 0 if you're hard-locked on the Pokémon to host. Otherwise, set this to the # of frames behind the target you are saved. In most cases, this will be 3 frames. But sometimes people will save further back on high-value dens as a safety against accidental roll-over.

### Host Online:

```
const char HOST_ONLINE          =   true;
```

Set this to true if you want to host online and false if you want to host locally.

### Start Time:

```
const uint16_t LOBBY_WAIT_DELAY =   60 * TICKS_PER_SECOND;
```

Wait this long in the lobby before starting the raid. Thus the start time is 3:00 minus this value. The default is to wait 60 seconds which starts the raid at 2:00.

### Catchability:

```
//  ALWAYS_CATCHABLE    All pokemon in this den are catchable.
//  MAYBE_UNCATCHABLE   There may be uncatchable pokemon.
const Catchability CATCHABILITY     =   ALWAYS_CATCHABLE;
```

Some dens have uncatchable Pokémon (i.e. Mewtwo or Zeraora). If the den has any such uncatchable Pokémon, you must this to MAYBE_UNCATCHABLE. If the den has uncatchable Pokémon, but it is hard-locked in a way that prevents any of them from showing up while rolling or hosting, then it is safe to leave this as ALWAYS_CATCHABLE.

### Accept Friend Requests:

```
const uint8_t FRIEND_ACCEPT_USER_SLOT       =   0;
```

If set to zero, do not accept friend requests. Otherwise, this is the user profile to accept friend requests for.

Once enabled, the program will automatically accept friend-requests while waiting in the lobby. It will do this multiple times to avoid getting disconnected and to give people more chances to see the code when streaming.

## Additional Raid Delay:

```
const uint16_t EXTRA_DELAY_BETWEEN_RAIDS    =   0 * TICKS_PER_SECOND;
```

If you are farming a Pokémon and the time between raids is too short to join consecutive raids, use this to add time between raids. This extra wait time is done after entering the den, but before entering the code.

## First Move Select:

```
const uint8_t MOVE_SLOT     =   0;
const char DYNAMAX          =   true;   //  Must set to false if you cannot dmax.
```

This is a silly feature that lets you choose a first move before resetting the game.

If "`MOVE_SLOT`" is 0, no first move is selected and the program resets as usual.

Otherwise, the move slot if the slot # for the move (1 = first move). The flag "`DYNAMAX`" will let you dmax/gmax the Pokémon as well.

## Alternate Games:

```
const bool ALTERNATE_GAMES  =   false;
```

If you are able to play both Sword and Shield on the Switch without changing cartridges, this option will let you host both Sword and Shield dens on the same user using the same settings. When set to true, the auto-host will soft-reset into the other game instead of the same game.

Needless to say, Sword and Shield must be the 1st and 2nd game slots in the Switch Home menu.

## Rollover Prevention:

```
const uint32_t TOUCH_DATE_INTERVAL = (uint32_t)4 * 3600 * TICKS_PER_SECOND;  //  4 hours
```

If the den is not rolling (SKIPS = 0), the date will advance naturally. This means that the den being hosted will eventually roll over. To prevent this, the auto-host will periodically touch the date at the specified interval.

Set this value to zero to disable the feature. This option is ignored if (SKIPS > 0) since it is unnecessary.

## Internet Connection Delays:

```
//  Time from "Connect to Internet" to when you're ready to enter den.
const uint16_t CONNECT_TO_INTERNET_DELAY    =   20 * TICKS_PER_SECOND;

//  "Communicating" when entering den while online.
const uint16_t COMMUNICATING_STANDBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Delay from "Invite Others" to when the clock starts ticking.
const uint16_t OPEN_ONLINE_DEN_LOBBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Time from start raid to reset. (when not selecting move)
const uint16_t RAID_START_TO_EXIT_DELAY     =   14 * TICKS_PER_SECOND;

//  This + RAID_START_TO_EXIT_DELAY = time from start raid to select move.
const uint16_t DELAY_TO_SELECT_MOVE         =   32 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the auto-host works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## AutoHostAirplane

Auto-host a softlocked den using airplane mode.

If you want to host a hard-locked den, use AutoHostRolling instead.

This program does not work while docked! Use AutoHostSleep or AutoHostFriendSearch instead!

This auto-host will always wait out the full 3 min. timer because it is impossible to airplane out when the host is the only player. Because of the need to wait out the timer, it takes about 4 min. for each loop.

**Optional Features:**

- Automatically accept friend requests.
- Add a delay between raids to make farming more efficient when you can't clear a raid before the next one starts.

**Important Warning:**

All auto-hosts carry a risk of destroying the den that is being hosted. The failure case is an error that causes the program to clear the raid and drop a new wishing piece. Dropping a wishing piece is a hard-save and cannot be reversed via backup save.

If you care about your den:

1. Get rid of all your wishing pieces.
2. Do not run the auto-hosts unattended.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Your switch must not be docked.
- Make sure you have a strong internet connection.
- You must be disconnected from internet.
- You must be standing in front of a wishing piece den with watts collected.
- You must be in the overworld.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

### Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

#### Raid Code:

```
const uint8_t RANDOM_DIGITS      =   0;
const char* RAID_CODE            =   "1280 0000";
```

See Raid Code Entry.

## Catchability:

```
const bool CATCHABLE     =   true;
```

If the Pokémon you are hosting is catchable, you must set this to true. If it is not catchable (such as Mewtwo or Zeraora), you must set this to false.

## Accept Friend Requests:

```
const uint8_t FRIEND_ACCEPT_USER_SLOT       =   0;
```

If set to zero, do not accept friend requests. Otherwise, this is the user profile to accept friend requests for.

Once enabled, the program will automatically accept friend-requests while waiting in the lobby. It will do this multiple times to avoid getting disconnected and to give people more chances to see the code when streaming.

## Additional Raid Delay:

```
const uint16_t EXTRA_DELAY_BETWEEN_RAIDS     =   0 * TICKS_PER_SECOND;
```

If you are farming a Pokémon and the time between raids is too short to join consecutive raids, use this to add time between raids. This extra wait time is done after entering the den, but before entering the code.

## Start Raid Early:

```
const uint16_t LOBBY_WAIT_DELAY      =   180 * TICKS_PER_SECOND;
```

This is a dangerous option that lets you start the raid before 3:00. If the raid starts empty, it will not be able to airplane out of the raid. Thus it can clear by itself and destroy the den.

By default, this is set to 180 seconds to wait out the entire 3:00 timer. If you set it less than that, you MUST keep an eye on the program. Be ready to kill the program if a raid goes empty.

## Rollover Prevention:

```
const uint32_t TOUCH_DATE_INTERVAL = (uint32_t)4 * 3600 * TICKS_PER_SECOND;  //  4 hours
```

Prevent the den from rolling over by periodically touching the date at this interval.

Set this value to zero to disable the feature.

## Internet Connection Delays:

```
//  Time from "Connect to Internet" to when you're ready to enter den.
const uint16_t CONNECT_TO_INTERNET_DELAY    =   20 * TICKS_PER_SECOND;

//  "Communicating" when entering den while online.
const uint16_t COMMUNICATING_STANDBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Delay from "Invite Others" to when the clock starts ticking.
const uint16_t OPEN_ONLINE_DEN_LOBBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Time from start raid to reset. (when not selecting move)
const uint16_t RAID_START_TO_EXIT_DELAY     =   14 * TICKS_PER_SECOND;

//  Time from airplane out to return to overworld.
const uint16_t EXIT_WAIT_DELAY              =   32 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the auto-host works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## AutoHostFriendSearch

Auto-host a softlocked den using the friend search method.

If you want to host a hard-locked den, use AutoHostRolling instead.

Unlike AutoHostAirplane, this program will work while docked. However, it needs to be properly configured or it may destroy the den.

This auto-host will always wait out the full 3 min. timer because it is impossible to airplane out when the host is the only player. Because of the need to wait out the timer, it takes about 4 min. for each loop.

**Optional Features:**

- Automatically accept friend requests.
- Add a delay between raids to make farming more efficient when you can't clear a raid before the next one starts.

**Important Warning:**

All auto-hosts carry a risk of destroying the den that is being hosted. The failure case is an error that causes the program to clear the raid and drop a new wishing piece. Dropping a wishing piece is a hard-save and cannot be reversed via backup save.

If you care about your den:

1. Get rid of all your wishing pieces.
2. Do not run the auto-hosts unattended.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Your switch must not be docked.
- Make sure you have a strong internet connection.
- You must be disconnected from internet.
- You must be standing in front of a wishing piece den with watts collected.
- You must be in the overworld.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

### Required Parameters:

- "`USER_SLOT`" must be set properly or the program may fail to disconnect. The consequences of not setting this properly can be the loss of the den.

## Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### Raid Code:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1280 0000";
```

See Raid Code Entry.

### Catchability:

```
const bool CATCHABLE    =   true;
```

If the Pokémon you are hosting is catchable, you must set this to true. If it is not catchable (such as Mewtwo or Zeraora), you must set this to false.

### Accept Friend Requests:

```
const bool ACCEPT_FRIEND_REQUESTS   =   true;
```

Set to true if you wish to accept friend requests.

### User Slot Position:

```
const uint8_t USER_SLOT             =   1;
```

This is the user profile for the user that is currently hosting. This must be set correctly to properly accept FRs and to successfully disconnect from the raid.

Once enabled, the program will automatically accept friend-requests while waiting in the lobby. It will do this multiple times to avoid getting disconnected and to give people more chances to see the code when streaming.

### Additional Raid Delay:

```
const uint16_t EXTRA_DELAY_BETWEEN_RAIDS    =   0 * TICKS_PER_SECOND;
```

If you are farming a Pokémon and the time between raids is too short to join consecutive raids, use this to add time between raids. This extra wait time is done after entering the den, but before entering the code.

### Start Raid Early:

```
const uint16_t LOBBY_WAIT_DELAY     =   180 * TICKS_PER_SECOND;
```

This is a dangerous option that lets you start the raid before 3:00. If the raid starts empty, it will not be able to airplane out of the raid. Thus it can clear by itself and destroy the den.

By default, this is set to 180 seconds to wait out the entire 3:00 timer. If you set it less than that, you MUST keep an eye on the program. Be ready to kill the program if a raid goes empty.

## Rollover Prevention:

```
const uint32_t TOUCH_DATE_INTERVAL = (uint32_t)4 * 3600 * TICKS_PER_SECOND;  //  4 hours
```

Prevent the den from rolling over by periodically touching the date at this interval.

Set this value to zero to disable the feature.

## Internet Connection Delays:

```
//  Time from "Connect to Internet" to when you're ready to enter den.
const uint16_t CONNECT_TO_INTERNET_DELAY    =   20 * TICKS_PER_SECOND;

//  "Communicating" when entering den while online.
const uint16_t COMMUNICATING_STANDBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Delay from "Invite Others" to when the clock starts ticking.
const uint16_t OPEN_ONLINE_DEN_LOBBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Time from start raid to reset. (when not selecting move)
const uint16_t RAID_START_TO_EXIT_DELAY     =   14 * TICKS_PER_SECOND;

//  Time from airplane out to return to overworld.
const uint16_t EXIT_WAIT_DELAY              =   32 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the auto-host works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## AutoHostSleep

Auto-host a softlocked den using sleep mode. Unlike AutoHostAirplane, this program will work while docked. But the way that this program works is hacky enough that it's not clear that it will always work.

If you want to host a hard-locked den, use AutoHostRolling instead.

This auto-host will always wait out the full 3 min. timer because it is impossible to airplane out when the host is the only player. Because of the need to wait out the timer, it takes about 4 min. for each loop.

As a side-effect of the implementation, it will always wait the 32 second "`EXIT_WAIT_DELAY`" in the overworld before it does anything.

Unlike AutoHostAirplane and AutoHostFriendSearch, this program does not have rollover prevention. This is because sleeping the Switch prevents the program from keeping track of time.

**Optional Features:**

- Automatically accept friend requests.
- Add a delay between raids to make farming more efficient when you can't clear a raid before the next one starts.

**Important Warning:**

All auto-hosts carry a risk of destroying the den that is being hosted. The failure case is an error that causes the program to clear the raid and drop a new wishing piece. Dropping a wishing piece is a hard-save and cannot be reversed via backup save.

If you care about your den:

1. Get rid of all your wishing pieces.
2. Do not run the auto-hosts unattended.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Make sure you have a strong internet connection.
- You must be disconnected from internet.
- You must be standing in front of a wishing piece den with watts collected.
- You must be in the overworld.
- Your location should be safe from getting attacked by wild Pokémon.
- Start the program in the Change Grip/Order Menu.

### Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### Raid Code:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1280 0000";
```

See Raid Code Entry.

### Catchability:

```
const bool CATCHABLE    =   true;
```

If the Pokémon you are hosting is catchable, you must set this to true. If it is not catchable (such as Mewtwo or Zeraora), you must set this to false.

### Accept Friend Requests:

```
const uint8_t FRIEND_ACCEPT_USER_SLOT       =   0;
```

If set to zero, do not accept friend requests. Otherwise, this is the user profile to accept friend requests for.

Once enabled, the program will automatically accept friend-requests while waiting in the lobby. It will do this multiple times to avoid getting disconnected and to give people more chances to see the code when streaming.

### Additional Raid Delay:

```
const uint16_t EXTRA_DELAY_BETWEEN_RAIDS    =   0 * TICKS_PER_SECOND;
```

If you are farming a Pokémon and the time between raids is too short to join consecutive raids, use this to add time between raids. This extra wait time is done after entering the den, but before entering the code.

### Start Raid Early:

```
const uint16_t LOBBY_WAIT_DELAY     =   180 * TICKS_PER_SECOND;
```

This is a dangerous option that lets you start the raid before 3:00. If the raid starts empty, it will not be able to airplane out of the raid. Thus it can clear by itself and destroy the den.

By default, this is set to 180 seconds to wait out the entire 3:00 timer. If you set it less than that, you MUST keep an eye on the program. Be ready to kill the program if a raid goes empty.

### Internet Connection Delays:

```
//  Time from "Connect to Internet" to when you're ready to enter den.
const uint16_t CONNECT_TO_INTERNET_DELAY    =   20 * TICKS_PER_SECOND;

//  "Communicating" when entering den while online.
const uint16_t COMMUNICATING_STANDBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Delay from "Invite Others" to when the clock starts ticking.
const uint16_t OPEN_ONLINE_DEN_LOBBY_DELAY  =   8 * TICKS_PER_SECOND;
```

```
//  Time from start raid to reset. (when not selecting move)
const uint16_t RAID_START_TO_EXIT_DELAY     =   14 * TICKS_PER_SECOND;

//  Time from airplane out to return to overworld.
const uint16_t EXIT_WAIT_DELAY              =   32 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the auto-host works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## Implementation Notes:

There are two cases that need to be handled: AC plugged in vs. not plugged in

When the Switch is plugged in, the Arduino continues to receive power and run while the Switch is sleeping. When the Switch is not plugged in, the Arduino is powered off during sleep. When the Switch wakes up, the Arduino is powered back on and restarts the program from the beginning.

Consequently, when the program starts, it doesn't know if it is starting for the first time or if it is resuming from sleep with AC unplugged. Thus the main loop for this version of the softlock auto-host has been rotated so that it starts immediately after disconnecting from the internet. The starting sequence has been arranged so that it can either wake up a sleeping Switch or start the program for the first time.

A potential point of unreliability is waking up without AC and on battery. So far it seems that immediately spamming the HOME button when the Switch sleeps will initiate the wake up sequence before the Switch can power down and reset the Arduino. Otherwise, once the Arduino is powered down, the program stops and cannot issue any more commands to the Switch.

brianuuuSonic's Auto-Host v4.0.1 uses local friend finder to disconnect from the internet. Why isn't this approach used? It has one major drawback - it requires that the user slot is correctly set to a user linked to a Nintendo account. If the user is not linked, the sequence will fail to disconnect from the internet and exit the raid. If left unattended, the raid will likely be cleared - wiping out the beam. Subsequent A button presses will drop a new wishing piece thus destroying the den.

Obviously, if the user is hosting online, the user will be linked. But it is a common error to incorrectly set the user slot. For the rolling auto-host, this error will only prevent auto-FR from working. But for a softlock auto-host, this error can destroy the den if left unattended.

There may be variants of the friend-searching method that are safer. But this is still under investigation.

## AutoHostMultiGame

This is a multi-game version of AutoHostRolling.

Don't use this program until you have used AutoHostRolling and are familiar with all its options. This program is significantly more complicated to setup.

This program does rolling auto-hosts across multiple game saves with different user profiles. In short, you specify a list of user profiles and auto-host settings. The program will then run down this list and auto-host each game accordingly. Once it is has finished hosting everything on the list, it starts from the beginning again.

Other than that, this program has all the same features as AutoHostRolling. Some settings can be customized per game save, others are global and apply to everything.

If you have 8 user profiles and the ability to play both Sword and Shield without changing cartridges, you can theoretically host up to 16 dens with this program.

**Important Warning:**

All auto-hosts carry a risk of destroying the den that is being hosted. The failure case is an error that causes the program to clear the raid and drop a new wishing piece. Dropping a wishing piece is a hard-save and cannot be reversed via backup save.

If you care about your den:

1. Get rid of all your wishing pieces.
2. Do not run the auto-hosts unattended.

### Instructions:

- All participating saves must have casual mode off.
- All participating saves must have their text speed set to fast.
- You must have system time unsynced.
- Make sure you have a strong internet connection. (if hosting online)
- All participating saves must be saved in front of a wishing piece den with watts collected.
- All participating saves should be in a location that is safe from getting attacked by wild Pokémon.
- If you are hosting from one game version only (Sword or Shield), it must be the 1st slot in the games list.
- If you are hosting from both Sword and Shield, they must be in slots 1 and 2 of the games list.
- You must have no games running.
- Start the program in the Change Grip/Order Menu.

If starting the game requires checking the internet (because it is digital on a non-primary Switch), you will need to open up Settings.c and change `START_GAME_REQUIRES_INTERNET` to true.

If any den has uncatchable Pokémon (like Mewtwo or Zeraora), you need to enable another setting to bypass that extra prompt. See Uncatchable Pokémon Prompt. Unfortunately, this cannot be configured on a per-game basis. It is a global option.

## Game List:

This is the important part of the multi-game autohost since it is where you specify the list of games and what settings to host within each game.

```
//  Game List
const RollingAutoHostSlot GAME_LIST[] = {
    {
        .game_slot          =   1,                      //  Game slot at start of program. (must be 1 or 2)
        .user_slot          =   2,                      //  User # to host from.
        .skips              =   3,                      //  # of frame skips to roll before hosting.
        .accept_FRs         =   true,                   //  Accept friend requests.
        .move_slot          =   0,                      //  If 0, don't select a move. Otherwise select this move slot.
        .dynamax            =   true,                   //  If 1st move select is enabled, dynamax as well.
        .post_raid_delay    =   0 * TICKS_PER_SECOND,   //  Extra wait time after each raid.
    },
    {
        .game_slot          =   1,
        .user_slot          =   3,
        .skips              =   0,
        .accept_FRs         =   true,
        .move_slot          =   3,
        .dynamax            =   true,
        .post_raid_delay    =   0 * TICKS_PER_SECOND,
    },


    //  DO NOT DELETE THIS
    {0},
};
```

This example shows 2 games running with different settings. The 1st game runs on user #2 with the usual 3-day roll. The 2nd game runs on user #3 as a hard-locked den with a 1st turn move select with dynamax.

In other words, there is a quite a bit of customization that can be done for each game.

## Per-Game Options:

These options can be customized per game.

### Game Slot Position:

```
.game_slot      =   1,      //  Game slot at start of program. (must be 1 or 2)
```

If you are hosting from one game version only, set this to 1. You don't need to read any further.

If you are hosting from both Sword and Shield, set this to the game slot of the version when the program starts.

For example:

- If you want to host on Sword and Sword is the 1st slot in the game list when the program starts, set to 1.
- If you want to host on Shield and Shield is the 2nd slot in the game list when the program starts, set to 2.

When hosting from both Sword and Shield, the games will swap position each time the program starts the 2nd game. This program knows this and will keep track of their positions. Thus, the slot that you specify is when the program starts, not the slot it will be at when the auto-host reaches that point.

### User Slot Position:

```
.user_slot      =   1,      //  User # to host from.
```

This is the user profile for the user that will be hosted.

### Catchability:

```
.always_catchable   =   true,   //  Pokemon in this den are always catchable.
```

Some dens have uncatchable Pokémon (i.e. Mewtwo or Zeraora). If the den has any such uncatchable Pokémon, you must this to false. If the den has uncatchable Pokémon, but it is hard-locked in a way that prevents any of them from showing up while rolling or hosting, then it is safe to leave this as true.

### Frame Skips:

```
.skips          =   3,      //  # of frame skips to roll before hosting.
```

This option sets how many frames to skip before hosting. Set to 0 if you're hard-locked on the Pokémon to host. Otherwise, set this to the # of frames behind the target you are saved. In most cases, this will be 3 frames. But sometimes people will save further back on high-value dens as a safety against accidental roll-over.

### Accept Friend Requests:

```
.accept_FRs     =   true,   //  Accept friend requests.
```

Set to true if you wish to accept friend requests for the user for this den.

### First Move Select:

```
.move_slot  =   0,      //  If 0, don't select a move. Otherwise select this move slot.
.dynamax    =   true,   //  If 1st move select is enabled, dynamax as well.
```

This is a silly feature that lets you choose a first move before resetting the game.

If "`.move_slot`" is 0, no first move is selected and the program resets as usual.

Otherwise, the move slot if the slot # for the move (1 = first move). The flag "`.dynamax`" will let you dmax/gmax the Pokémon as well.

### Additional Raid Delay:

```
.post_raid_delay    =   0 * TICKS_PER_SECOND,   //  Extra wait time after each raid.
```

If you are farming a Pokémon and the time between raids is too short to join consecutive raids, use this to add time between raids. This extra wait time is done in the Switch Home after closing the game after a raid.

## Global Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

All options here are global and will be applied to all the participating save files.

### Raid Code:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1280 0000";
```

See Raid Code Entry.

### Host Online:

```
const char HOST_ONLINE          =   true;
```

Set this to true if you want to host online and false if you want to host locally.

### Start Time:

```
const uint16_t LOBBY_WAIT_DELAY =   60 * TICKS_PER_SECOND;
```

Wait this long in the lobby before starting the raid. Thus the start time is 3:00 minus this value. The default is to wait 60 seconds which starts the raid at 2:00.

### Accept Friend Requests for Future Games:

```
const uint8_t FR_FORWARD_ACCEPT =   1;
```

While hosting the current den, accept friend requests for a future den.

- If 0, accept friend requests for the current den.
- If 1, accept friend requests for the next den to be hosted.
- If 2, accept friend requests for the den that will be hosted 2 dens later.
- …

The motivation here is that there is a significant delay from when you accept a friend requests to when the person can see the stamp. If you are accepting FRs for the den that's currently being hosted, the user that just got accepted may need to wait until the auto-host loops around the entire list of dens and returns the current one. This can take a very long time if you are hosting a lot of dens.

This option lets you shift the order of the FR accepts so that the accepted FRs become relevant much sooner.

### Rollover Prevention:

```
const uint32_t TOUCH_DATE_INTERVAL = (uint32_t)4 * 3600 * TICKS_PER_SECOND;  //  4 hours
```

If none of the dens are rolling (SKIPS = 0), the date will advance naturally. This means that the dens being hosted will eventually roll over. To prevent this, the auto-host will periodically touch the date at the specified interval.

Set this value to zero to disable the feature. This option is ignored if at least one den has (SKIPS > 0).

## Internet Connection Delays:

```
//  Time from "Connect to Internet" to when you're ready to enter den.
const uint16_t CONNECT_TO_INTERNET_DELAY    =   20 * TICKS_PER_SECOND;

//  "Communicating" when entering den while online.
const uint16_t COMMUNICATING_STANDBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Delay from "Invite Others" to when the clock starts ticking.
const uint16_t OPEN_ONLINE_DEN_LOBBY_DELAY  =   8 * TICKS_PER_SECOND;

//  Time from start raid to reset. (when not selecting move)
const uint16_t RAID_START_TO_EXIT_DELAY     =   14 * TICKS_PER_SECOND;

//  This + RAID_START_TO_EXIT_DELAY = time from start raid to select move.
const uint16_t DELAY_TO_SELECT_MOVE         =   32 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the auto-host works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## FriendDelete

Delete or block from your friend list.

Once you've auto-hosted enough, your friend list is going to fill up. FriendDelete can clean it up.

### Instructions:

Please read these instructions carefully as you can easily delete friends you didn't intend to delete.

1. Mark all the friends you want to keep as "best" friend.
2. Go into Pokémon Sword/Shield. Make sure you're offline.
3. Return to your friends list. All your best friends will be sorted at the top regardless of who's online or not.
4. Scroll all the way to the bottom of your friends list.
5. With the cursor over the last person, start this program. The program will delete your friends one-by-one from the back of the list.
6. Once it has reached your "best" friends, stop the program.

If you know exactly how many friends you wish to delete from the back of your list, you can set the "`friends_to_delete`" parameter accordingly and it will stop by itself when it's done.

### Required Parameters:

- `friends_to_delete`: You must set this parameter to the # of friends you wish to delete. Otherwise, you may end up deleting friends you didn't intend to delete.

### Options:

#### Number of Friends to Delete/Block:

```
const uint16_t FRIENDS_TO_DELETE    =   3;
```

The number of friends to delete or block.

#### Block Instead of Delete:

```
const bool BLOCK_FRIENDS            =   false;
```

Set this to true if you want to block these friends instead of just deleting them.

#### Internet Connection Delays:

```
//  Delay from opening a friend to when you can press buttons.
const uint16_t VIEW_FRIEND_DELAY    =   2 * TICKS_PER_SECOND;

//  Delay to delete the friend.
const uint16_t DELETE_FRIEND_DELAY  =   4 * TICKS_PER_SECOND;
```

```
//  Delay after deleting a friend.
const uint16_t FINISH_DELETE_DELAY  =   2 * TICKS_PER_SECOND;
```

These are all delays related to your internet connection. You may need to adjust these to ensure the program works correctly.

Try with the defaults first. If things don’t work, then try increasing these timings. Likewise, if your internet connection is very fast, feel free to reduce some of these delays.

## Implementation Notes:

There is no way for the program to distinguish "best" friends from regular friends. Therefore you must either tell the program exactly how many to delete, or you must stop it manually.

There is also no way for the program to know when to stop aside from telling it exactly how many friends to delete. When you delete a friend, the cursor automatically moves to the next friend (or the previous if it was your last friend).

Therefore, the only way to feasibly auto-delete friends is to manipulate the sorting order of your friends to group all the ones you want to delete together. Then run the script on that group. Unfortunately, this requires the user to monitor the program while it's running since it's very easy to accidentally delete someone you didn't intend to.

The default sorting order of your friends list is:

1. Best friends who are online.
2. Normal friends who are online.
3. Best friends who are offline.
4. Normal friends who are offline.

However, you can trick it to sort all best friends first by enabling local communication (such as by going offline in Pokemon). This suppresses the online status of your friends when you first load it - thus forcing it to sort only by best friend status. As long as you don't press Y on your friends list or re-enter the friends list while online, it won't re-sort based on online status.

## EggFetcher

Fetch eggs into your boxes without hatching them.

This program is used with EggHatcher to fetch and hatch eggs. Thus you first fetch a bunch of eggs with EggFetcher, then you hatch them with EggHatcher.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- The parents are already deposited at the Route 5 daycare center.
- Your party is full and none are eggs.
- The cursor in the menu is over the Town Map.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

If you are playing in Japanese or Italian, you need to set a flag in Settings.c or this program will not work.

### Strongly Recommended:

- **Masuda Method:** The Pokémon in the daycare are of different languages.
- Have the Oval Charm.
- Have the Shiny Charm.
- Have a fully upgraded bike.

For parents with neutral compatibility (same species or different OT), this program will fetch a box of eggs every ~15 min. or so.

It is okay to let this program fill all your boxes. If there's no room, you simply will not receive any more eggs. No harm is done.

### Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

#### Fetch Attempts:

```
const uint16_t MAX_FETCH_ATTEMPTS = 2000;
```

Make this many attempts to fetch an egg. The only reason to set this option is to avoid over-fetching of eggs and clogging up all your boxes.

## Advanced Settings:

These are advanced settings. You shouldn’t need to touch these unless something isn’t working and you’re trying to debug it yourself.

### Additional Travel Delay:

```
const uint16_t EXTRA_TRAVEL_DELAY = 400;
```

Extra duration to ride the bike before flying back to fetch another egg. You shouldn’t need to change this option.

## EggFetcher2

This is a new version of EggFetcher that is faster. It is currently in beta-testing and will eventually replace the existing EggFetcher.

Fetch eggs into your boxes without hatching them.

This program is used with EggHatcher to fetch and hatch eggs. Thus you first fetch a bunch of eggs with EggFetcher, then you hatch them with EggHatcher.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- The “Send to Boxes” option must be set to “Automatic”.
- Your bike must be fully upgraded. (This is a new requirement!)
- The parents are already deposited at the Route 5 daycare center.
- Your party is full and none are eggs.
- The cursor in the menu is over the Town Map.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

If you are playing in Japanese or Italian, you need to set a flag in Settings.c or this program will not work.

### Strongly Recommended:

- **Masuda Method:** The Pokémon in the daycare are of different languages.
- Have the Oval Charm.
- Have the Shiny Charm.
- Have a fully upgraded bike.

For parents with neutral compatibility (same species or different OT), this program will fetch a box of eggs every ~15 min. or so.

It is okay to let this program fill all your boxes. If there's no room, you simply will not receive any more eggs. No harm is done.

### Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

#### Fetch Attempts:

```
const uint16_t MAX_FETCH_ATTEMPTS = 2000;
```

Make this many attempts to fetch an egg. The only reason to set this option is to avoid over-fetching of eggs and clogging up all your boxes.

## EggHatcher

Hatch eggs from your boxes. This is the counterpart to EggFetcher.

This program will tolerate early hatching eggs as long as they don't hatch immediately to prevent the character from reaching the bridge. Thus EggHatcher can be used to clean up eggs of unknown remaining step-count as long as you keep an eye on it.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- Nicknaming needs to be off.
- Your party must have exactly 1 Pokémon.
- The one Pokémon in your party MUST be a flame body (or similar) Pokémon.
- The menu icons are in their default locations:
    - The Town Map must be in the 2nd row on the far left.
    - The Pokémon option must be in the 1st row, 2nd from the left.
- The cursor in the menu is over the "Pokémon" option.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

### Box Setup:

- Place entire boxes filled with eggs consecutively*.
- You must be on the first box of eggs.

*It is okay to have non-eggs mixed in with eggs. The only requirement is that first row of each box is occupied.

### Required Parameters:

- `BOXES_TO_HATCH`: You MUST set this to the # of boxes you wish to hatch. Otherwise, you either won't hatch all the eggs you want, or the program goes crazy if you run out of eggs.
- `STEPS_TO_HATCH`: You MUST set this to the correct step-count for the Pokémon you are hatching. If this is set too small, the program will fail and may unintentionally start a trade. (see Precautions)

The program will hatch eggs in batches of 5 (one column at a time). Once a box is complete, it moves to the next box. It will continue until it has hatched N boxes where N is specified by "BOXES_TO_HATCH".

### Safety Recommendations:

- Make sure you're offline. (see below)
- Disable the DLC button in case the program gets lost in the menus.

- Don't run this program unattended if any eggs are close to hatching.

As a precaution, it is strongly recommended to be offline. In the event that the eggs do not finish hatching before the program enters the storage system, there is a high chance that it will go into YCOMM and start a trade.

This program is not as reliable when the eggs hatch before the character can travel to the safe location on the bridge. Therefore, it is not recommended to run this program unattended when cleaning up eggs of unknown origin and unknown remaining step-count.

## Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### Boxes of Eggs to Hatch:

```
const uint8_t    BOXES_TO_HATCH      =   3;
```

Hatch this many boxes of eggs.

### Egg Step-Count:

```
const uint16_t   STEPS_TO_HATCH      =   5120;
```

The number of steps needed to hatch the eggs. Look up the value on Serebii.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Safety Time:

```
const uint16_t SAFETY_TIME       =   8 * TICKS_PER_SECOND;
```

Additional time added to the spinning. If any shinies are hatched, they will eat into this safety buffer along with any other unexpected slowdowns. Hatching a shiny takes 2 seconds longer than a non-shiny.

If you see that the program is going into Y-COMM or there is less than 5 seconds of extra spinning after the last egg in the batch, please report this as a bug. As a temporary work-around, you can increase this number.

### Hatch Delay:

```
const uint16_t HATCH_DELAY       =   88 * TICKS_PER_SECOND;
```

Total animation time for hatching 5 eggs when there are no shinies.

## EggCombined

This is where the real stuff begins. This is the combined egg fetching+hatching program.

***Do not*** use this program until you have familiarized yourself with EggFetcher and EggHatcher. EggCombined is harder to use and is very easy to screw up. But if used properly, it can be much more efficient.

Functionally, this program is like EggHatcher, but will also fetch N eggs for each batch of 5 eggs that are hatched. In other words, EggCombined requires an existing stock of eggs to run. It cannot start from nothing.

With proper usage of this program, it is possible to average under 1 minute per hatched egg (5120-step). This seems to come within 20% of manual hatching. The remaining inefficiency comes from the need for fly-backs to reset the view.

Be aware that the complexity of this program makes it inherently less reliable than the regular EggHatcher. Unlike EggHatcher, it does not tolerate early hatching eggs and requires all eggs to hatch at the same time at the specified step count. Furthermore, this program may have issues with fast hatching Pokémon like Magikarp.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- The “Send to Boxes” option must be set to “Automatic”.
- Nicknaming needs to be off.
- The parents are already deposited at the Route 5 daycare center.
- Your party must have exactly 1 Pokémon.
- The one Pokémon in your party MUST be a flame body (or similar) Pokémon.
- The menu icons are in their default locations:
    - The Town Map must be in the 2nd row on the far left.
    - The Pokémon option must be in the 1st row, 2nd from the left.
- The cursor in the menu is over the "Pokémon" option.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

If you are playing in Japanese or Italian, you need to set a flag in Settings.c or this program will not work.

### Box Setup:

- Place entire boxes filled with eggs consecutively*.
- You must be on the first box of eggs.
- All the eggs that you are hatching MUST hatch at the same time and at the specified # of steps. Eggs are not allowed to hatch early.

*It is okay to have non-eggs mixed in with eggs. But all boxes that you intend to hatch must be completely filled.

## Required Parameters:

- `BOXES_TO_HATCH`: You MUST set this to the # of boxes you wish to hatch. Otherwise, you either won't hatch all the eggs you want, or the program goes crazy if you run out of eggs.
- `STEPS_TO_HATCH`: You MUST set this to the correct step-count for the Pokémon you are hatching. If this is set too small, the program will fail and may unintentionally start a trade. (see Precautions)

The program will hatch eggs in batches of 5 (one column at a time). Once a box is complete, it moves to the next box. It will continue until it has hatched N boxes where N is specified by "BOXES_TO_HATCH".

Newly fetched eggs will be dropped one box ahead of the one that's being hatched. These will spill forward to later boxes as necessary, thus it is possible to set "BOXES_TO_HATCH" to more than what you have. But you will need to monitor the program to make sure it fetches eggs quickly enough to keep up with the hatching.

## Safety Recommendations:

- Disable the DLC button in case the program gets lost in the menus.
- As a precaution, it is strongly recommended to be offline. In the event that the eggs do not finish hatching before the program enters the storage system, there is a high chance that it will go into YCOMM and start a trade.

## Failure Cases:

- It is safe to run out of box space. If the egg-fetching wraps around and occupies the current column, the program will fail safely by "re-hatching" the same party over and over again until it's done.
- It is NOT safe to hatch an incomplete column due to running out of eggs. This will lead to an incomplete party being loaded which will swallow up newly fetched eggs. These eggs will not hatch at the correct time which will completely break the program.
- It is NOT safe for eggs to hatch early. If an egg hatches during a fetch attempt, it can put the program into an unexpected state.

## Usage Tips:

There are two primary ways to use this program depending on what kind of risks you're willing to take. The important fact is that the program can fail catastrophically if it runs out of eggs to hatch and attempts to load an incomplete column - and thus an incomplete party.

- **Safe Method:** Hatch only eggs that you already have.
- **Self-sustaining Method:** Hatch eggs that you expect to get.

### Safe Method:

The safe approach is to never set "BOXES_TO_HATCH" to more than the # of boxes of eggs you currently have. This ensures that you don't run out of eggs. Thus an efficient approach is to have half your boxes full of eggs and the other half empty. The program will hatch those eggs and fill up the remaining boxes with new eggs. Then you remove all the hatchlings and repeat.

Since the fetch and hatch rates will not be identical, over time, the # of eggs you have will drift away from half your boxes. So you'll need to correct it with EggFetcher or EggHatcher.

## Self-sustaining Method:

Once you have a feel for the fetch-rate, you can do a more aggressive approach. The design of this program allows it to hatch the eggs that it fetches. Thus you can start off with a relatively small # of eggs and still hatch a full game of eggs. But you need to be careful and make sure that at no point will you run out of eggs.

For parents with neutral compatibility (different OT, different species):

- Less than 5120-step eggs will hatch faster than they can be fetched.
- More than 5120-step eggs will hatch slower than they can be fetched.
- 5120-step eggs have an almost perfect equilibrium.

Thus, this program is easier to use for slower hatching eggs. For fast hatching eggs, it is recommended to use same species (different OT) parents for maximum compatibility. But this is still insufficient so it still requires monitoring.

Empirical testing with the common case (5120-step, neutral compatibility), shows that you can start EggCombined with 3 boxes of eggs and finish 14 hours later with ~3 boxes of eggs and 29 boxes of hatchlings.

## General Notes:

The strength of EggCombined goes beyond the efficiency of fetching+hatching at the same time. It also means that you no longer need to plan ahead on when to fetch and hatch eggs separately. Provided you have a sufficient stock of eggs, you can run the program for any amount of time (with sufficient box space) and interrupt it whenever. Your stock of eggs will still be there (maybe slightly different in quantity). Just fix the partially hatched box, (optionally) clean out the hatchlings, and you're ready to start the next run.

If all 32 boxes are available for hatching, you won't have to move eggs around (except for the incomplete box between runs). You can just let the "frontline" of eggs cycle across all the boxes making laps every 32 boxes.

With proper management, you can run EggCombined almost 24/7 with minimal effort. The only necessary interruptions are to periodically clear out the hatchlings and to make adjustments to maintain the desired stock of eggs.

# Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

## Boxes of Eggs to Hatch:

```
const uint8_t    BOXES_TO_HATCH      =    3;
```

Hatch this many boxes of eggs.

### Egg Step-Count:

```
const uint16_t  STEPS_TO_HATCH      =   5120;
```

The number of steps needed to hatch the eggs. Look up the value on Serebii.

### Fetches per Batch:

```
const uint8_t MAX_FETCHES_PER_BATCH     =   6;
```

For each batch of eggs, fetch new eggs up to this many times.

This parameter works by dividing the incubation time by the # of fetch attempts and doing a fetch at this interval. Therefore increasing this number will decrease the time between fetches which also decreases the chance of successfully fetching an egg. The optimal value is difficult to determine and is dependent on many factors.

For fast hatching eggs like Magikarp, you may need to decrease this number to as low as 1 or 2. This implies that it's impossible for fetching to keep pace with hatching as is the case with manual hatching anyway.

The program will automatically reduce this number if it determines that it is too large to function correctly.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Safety Time:

```
const uint16_t SAFETY_TIME      =   8 * TICKS_PER_SECOND;
```

Additional time added to the spinning. If any shinies are hatched, they will eat into this safety buffer along with any other unexpected slowdowns. Hatching a shiny takes 2 seconds longer than a non-shiny.

If you see that the program is going into Y-COMM or there is less than 5 seconds of extra spinning after the last egg in the batch, please report this as a bug. As a temporary work-around, you can increase this number.

### Early Hatch Safety:

```
const uint16_t EARLY_HATCH_SAFETY   =   5 * TICKS_PER_SECOND;
```

Eggs will not hatch early by more than this period.

### Hatch Delay:

```
const uint16_t HATCH_DELAY      =   88 * TICKS_PER_SECOND;
```

Total animation time for hatching 5 eggs when there are no shinies.

## Appendix: Fetch/Hatch Rates

This section dumps some empirical data on the relative fetch/hatch rates for different settings. You can use these tables to help plan those really long self-sustaining runs. All values are with the Oval Charm.

**How to read these tables:**

Each table entry has two numbers:

- Min Boxes: The minimum number of boxes of eggs you need to hatch 32 boxes.
- 32-Box Flow: The expected net change in the number of boxes of eggs after hatching 32 boxes.

Example:

- Parents: Rowlet (3840 steps) + Ditto (different OT)
- Fetches/Batch: 6

The table shows:

- Min Boxes: 10
- 32-Box Flow: -5

This means that you need to have at least 10 boxes of eggs in order to safely hatch 32 boxes. When it is done hatching 32 boxes of Rowlets, you can expect to have lost about 5 boxes of eggs.

### Neutral Compatibility Parents: (same species ***or*** different OT)

| **Step Count** | **5 Fetches/Batch** | **6 Fetches/Batch** |
|---|---|---|
| 3840 | | Min Boxes: 10<br>32-Box Flow: -5 |
| 5120 | | Min Boxes: 3<br>32-Box Flow: -1 |

### High Compatibility Parents: (same species ***and*** different OT)

| **Step Count** | **5 Fetches/Batch** | **6 Fetches/Batch** |
|---|---|---|
| 3840 | | Min Boxes: 5<br>32-Box Flow: -3 |
| 5120 | Min Boxes: 5<br>32-Box Flow: -3 | Min Boxes: 3<br>32-Box Flow: +3 |

## EggSuperCombined

Run MassRelease, then run EggCombined.

***Do not*** use this program until you are familiar with EggCombined since this is literally the same thing, but with an extra mass-release step. Therefore it is even more tricky to use correctly and safely.

EggSuperCombined is the most optimized of the egg programs in this package. When used properly, it can allow 24/7 hatching while requiring the user to only touch the Switch twice a day. Thus this program is most suitable for mass production of hatched shinies - especially multiple Switches where you want to minimize the amount of interaction required for each Switch.

The idea of EggSuperCombined is to automate away the mass-release that's needed between EggCombined runs. So between runs after removing the shinies, you no longer need to run and wait for MassRelease to finish through before starting the next batch of eggs. Thus, the "interrupt" time between runs is now on the order of a few minutes instead of an hour to release 20+ boxes.

So you hatch eggs overnight. When you wake up in the morning, spend only a few minutes to remove shinies and fix boxes. Then start EggSuperCombined before heading to work. Likewise, if the fetch/hatch rates are in sync, you only ever need to use EggSuperCombined in the steady state.

### Instructions:

These instructions are identical to that of EggCombined.

- Casual mode is off.
- Your text speed must be set to fast.
- The “Send to Boxes” option must be set to “Automatic”.
- Nicknaming needs to be off.
- The parents are already deposited at the Route 5 daycare center.
- Your party must have exactly 1 Pokémon.
- The one Pokémon in your party MUST be a flame body (or similar) Pokémon.
- The menu icons are in their default locations:
    - The Town Map must be in the 2nd row on the far left.
    - The Pokémon option must be in the 1st row, 2nd from the left.
- The cursor in the menu is over the "Pokémon" option.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start the program in the Change Grip/Order Menu.

If you are playing in Japanese or Italian, you need to set a flag in Settings.c or this program will not work.

### Box Setup:

This is similar to, but not identical to that of EggCombined.

- Place entire boxes filled with Pokémon to be released consecutively.
- The next N boxes can contain anything. They will be skipped.
- Place entire boxes filled with eggs consecutively*.
- You must be on the first box of Pokémon to be released.
- All the eggs that you are hatching MUST hatch at the same time and at the specified # of steps. Eggs are not allowed to hatch early.

*It is okay to have non-eggs mixed in with eggs. But all boxes that you intend to hatch must be completely filled.

## Required Parameters:

- `BOXES_TO_RELEASE`: You MUST set this parameter correctly or you may release Pokémon you didn't intend to release!
- `BOXES_TO_SKIP`: You MUST set this parameter correctly or you may not hatch the correct boxes.
- `BOXES_TO_HATCH`: You MUST set this to the # of boxes you wish to hatch. Otherwise, you either won't hatch all the eggs you want, or the program goes crazy if you run out of eggs.
- `STEPS_TO_HATCH`: You MUST set this to the correct step-count for the Pokémon you are hatching. If this is set too small or too large, the program will fail and may unintentionally start a trade. (see Precautions)

All recommendations, precautions, and usage tips for EggCombined apply to this program as well.

## Additional Recommendations:

- See Maximizing Switch Stability. The MassRelease portion of this program is sensitive to jitter.
- Don't leave any Pokémon you care about in the game. It goes without saying that mass-release is inherently dangerous to run unattended.
- Use a dedicated game with 32 empty boxes. If you're at the point where you're considering this program for optimized egg hatching, you might as well just speedrun a new game and shiny charm it with a living dex that you probably already have sitting in Pokémon Home.

## Example Box Layout:

The box setup is pretty confusing. So here is an example.

```
Box  1: empty
Box  2: empty
Box  3: 30 breedjects
Box  4: 30 breedjects
Box  5: 30 breedjects
Box  6: 30 breedjects
Box  7: your regular Pokémon
Box  8: 30 eggs
Box  9: 30 eggs
Box 10: 30 eggs
Box 11: 25 eggs
Box 12: empty
```

```
Box 13: empty
……..
Box 32: empty
```

Configure EggSuperCombined with these settings:

```
BOXES_TO_RELEASE  = 3
BOXES_TO_SKIP     = 2
BOXES_TO_HATCH    = 30
```

Starting with Box 3 as the current box, the program will:

1. Release 3 boxes: 3, 4, 5
2. Skip 2 boxes: 6, 7
3. Run EggCombined for 30 boxes: 8-32 (wrap-around) 1-5

## Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### # of Boxes to Release:

```
const uint8_t    BOXES_TO_RELEASE     =    2;
```

This is the number of boxes to release.

### # of Boxes to Skip:

```
const bool       BOXES_TO_SKIP        =    1;
```

This is the number of boxes to release.

### Boxes of Eggs to Hatch:

```
const uint8_t    BOXES_TO_HATCH       =    31;
```

Hatch this many boxes of eggs.

### Egg Step-Count:

```
const uint16_t   STEPS_TO_HATCH       =    5120;
```

The number of steps needed to hatch the eggs. Look up the value on Serebii.

### Fetches per Batch:

```
const uint8_t MAX_FETCHES_PER_BATCH     =    6;
```

For each batch of eggs, fetch new eggs up to this many times.

This parameter works by dividing the incubation time by the # of fetch attempts and doing a fetch at this interval. Therefore increasing this number will decrease the time between fetches which also decreases the chance of successfully fetching an egg. The optimal value is difficult to determine and is dependent on many factors.

For fast hatching eggs like Magikarp, you may need to decrease this number to as low as 1 or 2. This implies that it's impossible for fetching to keep pace with hatching as is the case with manual hatching anyway.

The program will automatically reduce this number if it determines that it is too large to function correctly.

## Advanced Settings:

These are advanced settings. You shouldn't need to touch these unless something isn't working and you're trying to debug it yourself.

### Safety Time:

```
const uint16_t SAFETY_TIME       =   8 * TICKS_PER_SECOND;
```

Additional time added to the spinning. If any shinies are hatched, they will eat into this safety buffer along with any other unexpected slowdowns. Hatching a shiny takes 2 seconds longer than a non-shiny.

If you see that the program is going into Y-COMM or there is less than 5 seconds of extra spinning after the last egg in the batch, please report this as a bug. As a temporary work-around, you can increase this number.

### Early Hatch Safety:

```
const uint16_t EARLY_HATCH_SAFETY   =   5 * TICKS_PER_SECOND;
```

Eggs will not hatch early by more than this period.

### Hatch Delay:

```
const uint16_t HATCH_DELAY       =   88 * TICKS_PER_SECOND;
```

Total animation time for hatching 5 eggs when there are no shinies.

## FastCodeEntry

This is one of the most evil programs. FastCodeEntry can enter any 8-digit raid code in about ~0.5 seconds. This includes complicated codes that require multiple keypad traversals.

So in all cases, this is fast enough to muscle your way into people's coded raids at superhuman speeds.

By using this program, you agree that you are an asshole. Furthermore, you will be required to embrace your asshole status by flaunting it in front of all the people you've blocked out of raids.

This is the same code entry routine that's used by the auto-hosts. But as a standalone program, it is much more dangerous and morally reprehensible.

Due to the nature of this program and its intended use-case, this program does not wait to start nor does it flash the LEDs. It runs immediately after connecting to the Switch.

### Instructions:

- The code entry pad must be up.
- The cursor must be over the "1" digit.
- No digits have been entered yet.
- Plug in the Arduino/Teensy and it will enter the code.

Once the program finishes entering the code, it will enter the lobby and disconnect itself from the Switch.

The idea here is that once you see the stamp you want, you manually mash A into it until the code entry pad pops up. Then you plug in or turn on the Arduino and it will hammer the code out for you.

To use this program effectively, it is strongly recommended to get a USB cable with a power button to easily turn on and off the Arduino/Teensy.

## Options:

### Raid Code:

```
const char* RAID_CODE = "9107 3091";
```

Pretty self-explanatory.

The program will skip non-digit characters. So the space or hyphen separator is optional.

## GodEggItemDupe

Duplicate items using the God Egg/MissingNo.

Keep in mind that the God Egg is considered a form of hacking since there is no known way to obtain it without ***someone*** having a hacked Switch.

### Instructions:

- Casual mode is off.
- Your text speed must be set to fast.
- The “Send to Boxes” option must be set to “Manual”.
- You must have the God Egg and a Ditto deposited at the Route 5 daycare center.
- You must have defeated the trainer past the east end of the bridge and collected the item next to her.
- Your party is full and none are eggs.
- The cursor in the menu is over the Town Map.
- You must be on Route 5.
- You must be on your bike.
- You must be in the overworld. (not in the menu)
- There is nothing in front of you that can be interacted with.
- Start this program at the Change Grip/Order menu.

If you are playing in Japanese or Italian, you need to set a flag in Settings.c or this program will not work.

The program will clone each Pokémon in your party, take its item, and release it. Then it repeats the process forever.

### Box Setup:

- The entire 1st column of the current box must be empty.
- Your party has 6 throw-away Pokémon* - each holding an item you wish to duplicate.
- All 6 Pokémon* in your party are releasable. (not an egg, not a fused mon)

*If you don't want to clone items on all 6 Pokémon in your party, you can set "`PARTY_ROUND_ROBIN`" to cycle through fewer Pokémon/items.

### Strongly Recommended:

- Have the Oval Charm.
- Have a fully upgraded bike.

### Safety Recommendations:

- Don't leave any Pokémon you care about in the game. This program releases Pokémon and has been observed to fail in a way that saves the game.
- If you do need to leave Pokémon you care about, keep them out of the 1st column of any box.

## Options:

In addition to the main options below, there are more options in Settings.c that can be configured if you encounter problems.

### Party Round Robin:

```
const uint8_t PARTY_ROUND_ROBIN      =   6;
```

Round-robin through this many party Pokémon. By attaching different items to different Pokémon in your party, you can duplicate different items in the same run. Thus it can clone up to 6 different items in each run.

Examples:

- If set to 1, it will only clone the item held by the 1st party member.
- If set to 2, it will alternate cloning items held by the 1st and 2nd party members.
- If set to N, it will clone the items held by the 1st N members in your party in a round-robin fashion.

This option doesn't increase the speed of the item duplication. It merely gives you more variety. If you run the program unattended overnight, you may get around 300 of the same item. By setting this parameter, you can instead get 50 of 6 different items.

### Detach Item:

```
const bool DETACH_BEFORE_RELEASE     =   false;
```

When you release a Pokémon, the item automatically detaches and goes to your inventory. But certain items (like Rusted Sword/Shield) will prevent you from releasing the Pokémon.

If you are duplicating such items, you must set this to true.

## Advanced Settings:

These are advanced settings. You shouldn’t need to touch these unless something isn’t working and you’re trying to debug it yourself.

### Additional Travel Delay:

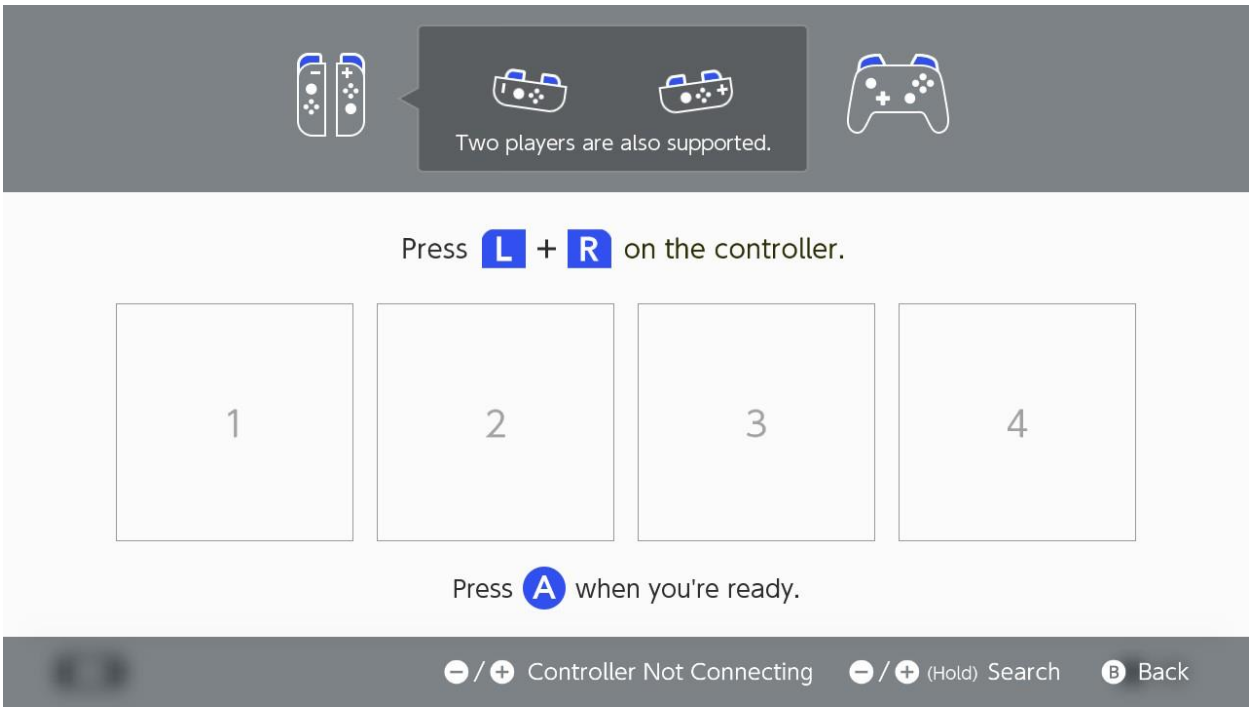
```
const uint16_t EXTRA_TRAVEL_DELAY    =   800;
```

Extra duration to ride the bike before flying back to fetch another egg. You shouldn’t need to change this option.

# Appendix

## Change Grip/Order Menu

Nearly all programs start in the grip menu. This is what it looks like:




Pokémon Sword/Shield only allows one external controller to be connected at a time. If you enter the game with multiple controllers, all but the first one will be disconnected. Thus the Arduino must be the first controller.

When you enter grip menu, all external controllers will be disconnected. That way when the Arduino is connected, it will become controller #1.

## Raid Code

This option is used by all the auto-hosts:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1280 0000";
```

These two values let you set the raid code.

### No code (FFA):

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "";
```

### Fixed Code:

```
const uint8_t RANDOM_DIGITS     =   0;
const char* RAID_CODE           =   "1234 5678";
```

This will always use the code “12345678” for every raid.

Notes:

- The program will skip non-digit characters. So the space or hyphen separator is optional.
- If the code is less than 8 digits, it will be padded out to 8 digits with zeros.

### Random Code:

```
const uint8_t RANDOM_DIGITS     =   4;
const char* RAID_CODE           =   "1234 5678";
```

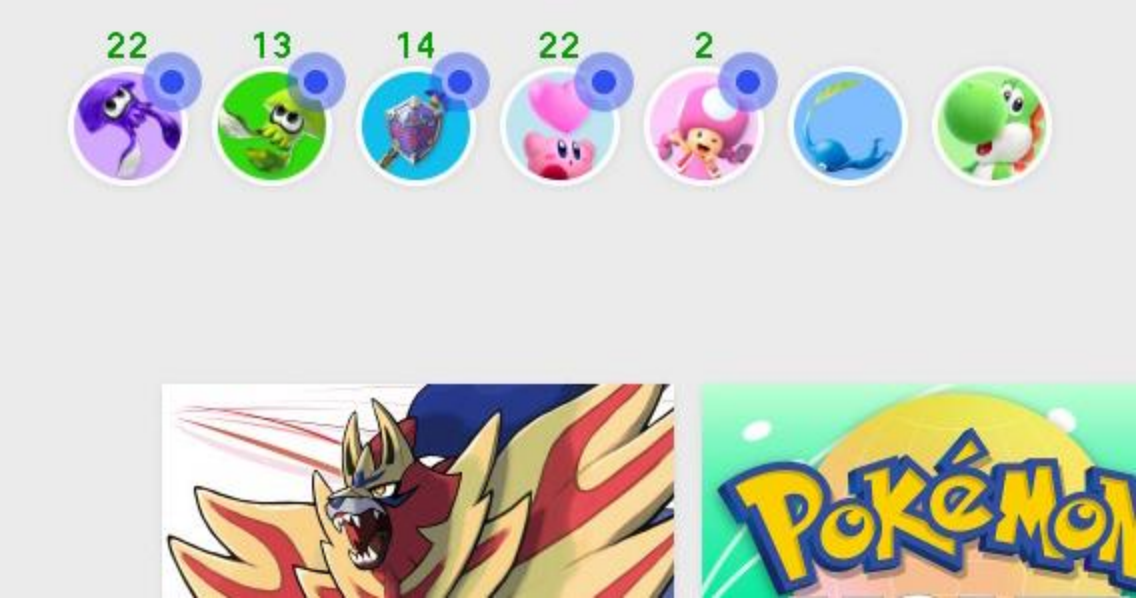
In this example, it will randomize the first 4 digits of the raid code. The remaining digits will be copies of the 4$^{th}$ digit. For example: “12344444”, “78966666”, etc... The purpose of repeating the last digit is to make it easier for raiders to enter the code.

When using a random raid code, the contents of “`RAID_CODE`” are used to generate a starting seed for the RNG. If you don’t know what that means, just ignore it and leave it as is.

## User Slot Number

The user profile # is used by multiple programs. For the auto-hosts, it needs to be set correctly for friend requests to be accepted on the correct account. For MultiGameFossil, it needs to be set correctly to enter the correct game.

On the Switch Home menu, all the user profiles are at the upper-left corner of the screen.




The left-most user is slot 1. The next one over is slot 2, etc... The Switch allows up to 8 users.

## Maximizing Switch Stability

Some programs that are timing sensitive will recommend or require that Switch be put into a stabilized state where it is less likely to jitter and drop button presses.

This is most important to the day skippers, but it is also applicable to fault-intolerant programs like MassRelease and EggSuperCombined.

To maximize program stability:

1. Disconnect all controllers and Joy-Cons which are known to drift. (Don't forget the attached Joy-Cons!)
2. Disconnect from the internet. Unplug the Ethernet cable if you're wired up.
3. Turn on airplane mode. To run docked, undock the Switch, turn on airplane mode, then redock. The Switch will remain in airplane mode after redocking.
4. Restart the Switch if it's been playing for a while.

Empirical evidence seems to suggest that completely isolating the Switch from outside interference (by shutting off all network traffic) improves stability and reduces the frequency of errors.

## Day Skipper Errors and Auto-Corrections

This package has 3 high-speed date skippers – one for each of the 3 different date formats.

- **DaySkipperJPN:** YYYY/MM/DD – hh:mm
- **DaySkipperEU:** DD/MM/YYYY – hh:mm
- **DaySkipperUS:** MM/DD/YYYY – hh:mm – AM/PM

Like other fast day skipper implementations, these skippers are not 100% reliable. They can make errors. In stable environments, these skippers can run flawlessly for many days straight. But in unstable environments they can make multiple errors in a day or have bursts of many errors in just an hour.

The JPN and EU day skippers are the fastest and most reliable. The US skipper is slower and has been observed to be significantly less stable possibly due to its longer scrolling sequence.

Day skipper errors fall into two categories:

1. **Missed Skip:** A single skip is missed, but the program continues to normally.
2. **Trapping:** The program gets stuck toggling the time-sync.

Missed skips are harmless besides missing the skip. The program continues normally as if nothing happened. But when it finishes, it will finish short of the target number of skips. These are the most common errors.

However, trapping errors are the more severe since they prevent any further skipping from being done. These happen when an error causes the program to break out of its normal skipping routine and get stuck toggling the time-sync. Thus they are not self-correcting and you can go away for a long time and come back having completed very few skips.

The skippers in this package are different from other day skippers. First, they are more resilient to errors. Minor errors that cause other day skippers to trap will not cause these skippers to trap. Secondly, if a trapping error still manages to happen, the skippers here will self-recover with a periodic routine that puts the program back into the normal skipping state. Needless to say, all the skips that were supposed to happen while being trapped will have been dropped.

While errors will still cause the skipper to fall short of the target, they will not stop the program from skipping. Rest assured that if you leave one of these day skippers running for hours or days, it will still complete the vast majority of skips even if the environment is unstable.

**Notes:**

- All Skippers: If the program is stuck toggling the time-sync, it has encountered a trapping error, but has not recovered from it yet. Wait a bit and it will recover when the auto-recovery routine runs.
- JPN Skipper: If the hour has changed, it means the program has missed at least one skip.
- JPN Skipper: If the month and/or time zone have changed for the JPN skipper, it means that the program has encountered and recovered from a trapping error.
- JPN Skipper: If the month has been changed to the current month (due to a trapping error) and is not a 31-day month, the amount of efficiency you lose is negligible. (0.1% for 30-day month, 0.3% for February) So it's fine to just let it continue unless you have an OCD to fix it or something.

## Global Settings

This section covers all the common global settings in **Settings.c**. There are many more advanced settings not documented here which you should not need to touch. They exist mainly for debugging purposes.

### Start Game Requires Internet

```
const bool START_GAME_REQUIRES_INTERNET    =    false;
```

Set this to true if starting the game requires checking the internet. Otherwise, programs that require soft-resetting may not work properly.

If the game is not a physical cartridge and the Switch is not the primary Switch, starting the game will require checking the internet to see if it can be played. If this is the case, set this to true.

Setting this option to true will slow down soft-resetting by about 3 seconds.

### Tolerate System Update Menu

```
const bool TOLERATE_SYSTEM_UPDATE_MENU     =    false;
```

Some programs have the ability to tolerate the system update menu at the cost of speed/performance. Setting this to true enables this.

Most programs that start from the grip menu and enter the game only once will always try to dodge the system update window. This setting only applies to programs that repeatedly go in-and-out of the game. For example: DenRoller and AutoHostRolling.

The date-spam farmers are performance-critical and do not use this setting. They do not tolerate the update window and will instead enter a safe loop within the settings that does nothing.

### Automatic Deposit

```
const bool AUTO_DEPOSIT          =    true;
```

When set to true, all programs will assume that “Send to Boxes” is set to “Automatic”. This slightly improves the speed of the relevant programs.

### Japanese Egg Fetching

```
const bool JPN_EGG_FETCHING      =    false;
```

For some languages (namely Japanese and Italian), fetching an egg from the daycare lady has an extra line of text which will normally break the egg programs. If this applies to you, set this to true to make the program bypass that extra line of text.

### Uncatchable Pokémon Prompt Avoidance Method

```
const bool DODGE_UNCATCHABLE_PROMPT_FAST    =   false;
```

Some den Pokémon are uncatchable (example, Mewtwo and Zeraora). When these Pokémon show up in den, there is an extra prompt that is needs to be cleared before you can enter the lobby.

This package provides two methods to clear the extra prompt.

- **Fast Method:** Attempt to avoid the prompt using the fast method of quickly pressing A again.
- **Slow Method:** Attempt to avoid the prompt using a slower method that's more reliable.

This setting allows you to choose which method to use.

Intuitively, this seems more complicated than it needs to be. Most other programs will avoid the prompt using the 2nd method – quickly pressing A again. However, this method is not reliable.

Some Switches are really fast and some are really slow.

- If the 2nd A press is too early, it may not be able to clear the uncatchable prompt on slow Switches.
- If the 2nd A press is too late, it may enter Switch Pokémon if the Pokémon is catchable on fast Switches.

Empirical evidence has shown that there is no window of time where it is safe to press A that would work on all Switches, or even the majority of Switches.

Therefore, the recommendation is:

1. If you won't want to deal with this, leave the setting set to false.
2. If you want to make den rolling and lobby entry fast, you try setting this option to true.

If you want to get fancy, you can experiment with "`UNCATCHABLE_PROMPT_DELAY`" as well.

## Version History

**2020-09-30:**

- Fixed some errors in the manual.

**2020-09-27:**

- Increased the default start game delay by 2 seconds to better accommodate slower Switches.
- Revamped the handling of uncatchable Pokémon. Uncatchable prompt avoidance can now be configured on a per-program basis. For AutoHostMultiGame, it can be configured on a per-game basis.

**2020-09-26:**

- All programs that receive Pokémon must now have “Send to Boxes” set to “Automatic”.
- Added a global setting that speeds up certain programs when “Send to Boxes” is set to “Automatic”. This new option is enabled by default.
- EggCombined, EggSuperCombined, and MultiGameFossil are now slightly faster due to the above.
- Added EggFetcher2 which uses the triangulation method to avoid fly-backs.

**2020-09-21:**

- Slightly improved reliability of date-skipping for den rolling.
- Added a note for the day skippers about airplane mode and non-primary Switch digital games.
- Added some more setup instructions for Arduino and Teensy.
- Added a hardware recommendations section.
- Added a start program callback to complement the end program callback.

**2020-09-11:**

- Minor timing changes to improve stability for game-switching programs. (i.e. multi-host and fossils)
- Fixed a stability issue with the code entry. Some codes will be a tiny bit slower.

**2020-09-09:**

- Reduced memory footprint for all programs.
- The !BuildAll scripts will now work even if you run them from the wrong working directory.

**2020-09-07:**

- The !BuildAll scripts will now tell you if WinAVR isn’t installed.
- AutoHostRolling now closes the game before it rolls back the date for the next run.
- AutoHostFriendSearch now disconnects one second earlier to compensate for the extra time needed to navigate all the way into the user profile.

**2020-09-04:**

- Added DaySkipperJPN-7.8k. A less fault-tolerant JPN skipper that runs at 7.8k skips/hour.
- Increased the auto-correction period for the JPN skipper from 500 to 1000.

- Day skippers are furthered hardened against trapping errors.
- Slightly adjusted timings for MultiGameFossil.
- Numerous updates to this manual.

**2020-08-30:**

- Fixed the category rotation option for ClothingBuyer.

**2020-08-29:**

- Fixed a stupid bug for all date navigation programs.

**2020-08-28:**

- Added ClothingBuyer. It's not efficient, but it will eventually buy out an entire store.
- When a program finishes, it will run a user-provided callback. Here you can insert code to turn on LEDs or other output pins.
- Failure case mitigation for date-spamming.

**2020-08-27:**

- Fixed another issue with the rollover prevention.
- Fixed a stability issue in date-spamming.
- Reduced binary sizes.

**2020-08-26:**

- Fixed a bug in the new code entry.
- Fixed the rollover prevention to not mess up the date.
- Fixed the date-spam farmers so that they correctly roll the year for EU date format.

**2020-08-25:**

- Added rollover prevention to AutoHostAirplane, AutoHostFriendSearch, AutoHostRolling, and AutoHostMultiGame. This will prevent the den from rolling over if left running for a very long time.
- When starting a program, the LEDs will flash momentarily to indicate that the program has actually started running. If the LEDs don't flash (stay on or off), it means the device did not connect properly.
- GodEggItemDupe has a new option to forcibly detach the item before releasing the Pokémon. This is needed to allow duplication of special items like Rusted Sword/Shield.

**2020-08-22:**

- Code entry (including FastCodeEntry) is slightly faster for codes that require multiple long traversals.
- When starting a program, the LEDs will turn on momentarily to indicate that the program has actually started running.
- Slightly tweaked date-spamming to improve reliability.

**2020-08-09:**

- Added TradeBot.

- Increased raid exit time by 1 second for all auto-hosts.

**2020-07-31:**

- MultiGameFossil can now hop between game versions. (Sword + Shield)
- AutoHostMultiGame will now accept FRs for the next den instead of the current one. Raiders no longer need to wait for the autohost to complete a full cycle before they see any stamps. This can be configured.
- Add recommendation to stand behind the den/tree for WattFarmer and BerryFarmer.
- Improved efficiency of GodEggItemDupe.
- Minor reliability tweaks.

**2020-07-19b:**

- AutoHostMultiGame can now hop between game versions. (Sword + Shield)
- Tweaks to make the rolling auto-hosts less likely to kill the den.

**2020-07-19:**

- Bug fixes and reliability improvements.

**2020-07-15:**

- Added AutoHostMultiGame.
- Numerous updates to this manual.

**2020-07-14:**

- Added StowOnSideFarmer to farm the bargains dealer.
- Added 7k skippers for US and EU date formats. These run at 7.1k and 7.5k skips/hour respectively.
- More reliability tweaks to the day skippers.
- All day skippers now default to 10 skips instead of 200,000. This makes it harder to overskip by using the wrong program.
- Adjusted egg timings to improve reliability.
- Code entry will skip invalid digits instead of converting them to zero.
- Removed “!ReadMe.txt” since it has been superseded by this manual.
- Removed documentation for all programs since they are now in this manual.
- Minor grammar and wording changes to this manual.
- Build scripts have been shuffled around.

**2020-07-11:**

- Added this manual.

**2020-07-10:**

- The 7k skipper should be more resistant to trapping errors.

**2020-07-09:**

- Fixed an issue in the 7k skipper auto-correction.

**2020-07-08:**

- Merged all 3 MCU packages into one.
- Exposed some more den timings.

**2020-07-04:**

- Fixed an issue in MassRelease where it wouldn't turn on the LEDs when done.

**2020-07-03:**

- Minor timing adjustments to den rolling and auto-hosts.
- Adjustments to game startup for MultiGameFossil.
- Fixed the way MultiGameFossil ends.
- EggSuperCombined now recommends turning on airplane mode to improve mass release stability.

**2020-06-30b:**

- Fixed an issue where date-spam programs may not completely roll back the date.

**2020-06-30:**

- WattFarmer and BerryFarmer now have an option to periodically save the game. This is disabled by default since crashes are extremely rare.

**2020-06-29b:**

- When the 7k skipper finishes, it will go in-and-out of the date change menu every 15 seconds to prevent the time from naturally advancing past midnight thereby resulting in an extra (unintended) day skip.

**2020-06-29:**

- The BuildAll script will now tell you which programs failed to build.
- The BuildAll script will now delete old .hex files so you don't accidentally use old ones.
- Minor timing adjustments to auto-hosts.
- Month adjusting corrections have been removed from 7k the skipper. Thus it's no longer necessary to specify the real life month. The overhead for a month-adjusting correction is greater than skipping on a 30-day or even a 28-day month following a trapping error that syncs the clock.

**2020-06-27:**

- Update LUFA library to latest. (790ac4d)

**2020-06-26:**

- Fixed a possible compilation error in the LUFA library for Mac.

- The 7k skipper will now blink the LEDs to indicate roughly how many skips are left. The faster it blinks, the fewer skips are left.

**2020-06-25:**

- The BuildAll script now tells you when it's safe to close it.
- Yet even more timing changes to improve egg program stability post-DLC update.
- Potential fix for GodEggItemDupe failing in French.
- Potential fix for den rolling getting stuck in box.
- Potential fix for den rolling not rolling back the date.
- Experimental feature to alternate between Sword+Shield games for AutoHostRolling.

**2020-06-22:**

- More den and auto-host timing adjustments to increase stability.

**2020-06-21b:**

- Added FastCodeEntry. All the power to be an asshole.

**2020-06-21:**

- Added new (slower, but more reliable) method for dodging uncatchables when den rolling. This method is disabled by default.
- Updated instructions for the 7k Skipper to specify that the Isle of Armor Dojo is not a place that can be used to skip frames. You must be in an actual Pokémon center.

**2020-06-19:**

- Even more timing changes to improve egg program stability post-DLC update.

**2020-06-18:**

- Timing changes to improve the reliability of egg programs post-DLC update.
- Timing changes to hopefully improve reliability of den rolling.
- Auto-hosts now support partially random raid codes. This will make it much easier to enter random raid codes for streamers.

**2020-06-17b:**

- Auto-hosting with no code now works for uncatchable Pokémon.
- Den rolling should now work for uncatchable Pokémon.
- Faster code entry because 8-digit codes suck.

**2020-06-17:**

- 8-digit codes for DLC update.

**2020-06-15:**

- Added AutoHostFriendSearch.
- The artificial delays for auto-hosts has now been fixed.
- Updated documentation for the 7k skipper. Empirical evidence suggests it's a lot more stable than the documentation makes it sound like.

**2020-06-13:**

- Fixed an issue where AutoHostSleep and AutoHostAirplane will start the raid even if nobody joined.

**2020-06-12:**

- When a program finishes, it will turn on the LEDs.
- Further relaxed timings for mass release to improve stability.
- Increased the default auto-correction interval for 7k skipper from 250 -> 500.

**2020-06-09:**

- New implementations for all remaining Brainuuu programs:
    - TurboA
    - MassRelease
    - LotoFarmer
    - BerryFarmer
- Build scripts are now parallelized.

**2020-06-08:**

- Most program instructions have been rewritten to make them more clear.
- AutoHostSleep and AutoHostAirplane now have a (dangerous) option to start a raid before the 3:00 timer runs out.
- Possible fix for egg programs messing up box operations.

**2020-06-06b:**

- Possible fix for the egg programs playing a different game.

**2020-06-06:**

- FriendDelete now has an option to block users instead of just deleting them.
- Soft-reset timings have been adjusted and are now configurable.
- Den rolling timings have been adjusted to hopefully improve reliability.
- Potential fix for auto-hosts killing dens.
- Build scripts now save build errors to logs.
- Build warnings will now error.
- Minor timing changes to all date-spamming programs.

**2020-06-04:**

- Exposed FriendDelete timings for user configuration.
- Potential fix for AutoHostRolling destroying dens.
- Potential fix for DenRoller and AutoHostRolling failing the first date skip.

**2020-06-01:**

- Add debugging option to toggle timesync after egg program ends.

**2020-05-29:**

- Increased box pickup/drop time to increase stability of egg programs.
- Speculative fix for MultiGameFossil not stopping when it's supposed to.
- Updated warnings and recommendations for all auto-hosts.

**2020-05-27:**

- Fixed a build issue with EventBeamFinder caused by a naming conflict.
- Adjusted Home->Game timings to improve DenRoller and AutoHostRolling stability.

**2020-05-26:**

- More adjustments to EggSuperCombined mass release timings.
- The default exit time for auto-hosts has been pushed earlier.
- MultiGameFossil now tolerates the system update window.

**2020-05-23:**

- Adjusted mass release timings to increase stability of EggSuperCombined.
- Added empirical fetch/hatch rates data for EggCombined.
- The default for START_GAME_REQUIRES_INTERNET is now false. I have yet to see anyone play a downloaded copy of the game on a non-primary Switch.

**2020-05-22:**

- Tweaked den rolling timings to hopefully increase stability.
- Exposed more den rolling timings to configuration.

**2020-05-21:**

- Tightened some timings in the auto-hosts.
- Slightly relaxed timings for MultiGameFossil to improve reliability.
- Relaxed den rolling entry timings and add an option to configure it.

**2020-05-19:**

- Relaxed mass release timings in EggSuperCombined to improve reliability.
- More build script refactorings.

**2020-05-18:**

- New method of building. Double-click on the appropriate .cmd file instead of editing the makefile!
- Re-added the atmega16u2 MCU because I'm stupid.
- Fixed some settings options that broke during the refactor.

**2020-05-17:**

- Major refactoring of the build process.
- Further tweaked date spamming for WattFarmer.
- Increased box-change delay to improve stability of egg programs.

**2020-05-16:**

- The 4k skipper has been removed. It will be distributed in a separate package.
- Fixed a minor issue where JPN egg hatching tolerance was always enabled.
- Added a "Required Parameters" section to the instructions of each program to clarify what parameters need to be set properly to use the program.
- Slightly tweaked timings for date spamming.
- Moved a ton of box navigation timings into Settings.h so they're more easily configured.

**2020-05-11:**

- BeamReset now goes in-and-out of the game several times to flash the beam before it resets.
- EventBeamFinder: Drop wishing pieces until you find an event den.
- Added an option to make the egg programs work in Japanese.

**2020-05-10:**

- Added MultiGameFossil (beta). Now you can revive fossils for more than 4.5 hours at a time by utilizing multiple saves!
- BeamReset has been updated to require slow text. Apparently some Switches save extremely quickly - enough to break the program under fast text speeds.

**2020-05-09:**

- Added BeamReset to make purple beams less painful.
- Fixed some timings that could cause den rolling and rolling-auto-host to fail to reset the game.
- Lot of minor timing changes.

**2020-05-08:**

- Added support for random raid codes to the auto-hosts.
- Relaxed the box entry timings for all the egg programs.
- Rolling auto-host start time no longer includes the 8 second lobby open wait. So now it starts closer to 2:00 now.

**2020-05-06:**

- Recalibrated "TICKS_PER_SECOND" and adjusted it from 123 -> 125.

- Fixed an issue in the egg hatchers where it can incorrectly back out all the way to the overworld on faster Switches.
- Fixed some timings in the egg hatchers that broke after migrating the framework.
- Fixed game entry timings when starting game requires internet.

**2020-05-05:**

- Moved "Settings.h" into the main folder so that it's more accessible.
- Cleaned up "Settings.h" so that's clearer and better documented.
- Added an option that will allow programs that soft-reset to work when starting the game requires checking the internet because the game isn't a physical cartridge and the Switch isn't the primary Switch.
- The 7k skipper's is slightly faster again.
- The 7k skipper's default auto-correction interval has been reduced from 500 to 250.
- Adjusted some timings with the mass release to make it more reliable.

**2020-05-03: (beta)**

- Slightly faster date spamming for watt farmer and den rolling.
- Slightly modified timings for box operations in the egg programs.
- The 7k skipper is about 1.6% faster.

**2020-04-29:**

- Fixed EggHatcher which was broken by a refactoring.
- Fixed the start-timer for AutoHostRolling to 2:00. Accidentally left it on 2:15 after hosting yesterday.
- Added this ReadMe file.

**2020-04-27:**

- Lots of minor timing adjustments for all programs.
- Another fix that hopefully fixes the no-code raids.
- Added EggSuperCombined. Combines mass-release with EggCombined.
- Added forbidden program GodEggItemDupe. Mass duplicate items with the God Egg.

**2020-04-23:**

- Relaxed the den entry timings for auto-hosting to hopefully fix the no-code problem.
- Add warnings about messing with TV displays to all programs.
- Updated documentation for EggCombined.

**2020-04-21:**

- Fixed some timings with the egg programs to make them more reliable.
- Fixed some timings with the WattFarmer to make them more reliable.
- The WattFarmer's starting sequence is more optimized.
- Renamed the two JPN day skippers to "DaySkipperJPN_4k" and "DaySkipperJPN_7k".
- Slightly faster auto-correction logic for both day skippers.

- The 7k skipper now asks for the current (real-life) month so that auto-corrections will land you back on a month with 31 days.
- Added a global option to make the programs more tolerant of the system update window. This is disabled by default for den-rolling and auto-hosts since it affects performance.

**2020-04-20:**

- Added WattFarmer which is about 50% faster than Brainuuu's.
- Den rolling is now better optimized.

**2020-04-19: (beta)**

- Massive refactor of all programs to switch to a new controller framework.
- The 4700 skipper has not been migrated since doing so would make it identical to the 7k skipper. For now, the public only knows (rumors) about the 4700 skipper. But they have no idea about the 7k skipper.

**2020-04-18:**

- Refactor of a ton of code. So not a lot of functional changes.
- Den rolling and FR-accepts are now faster.

**2020-04-16:**

- DaySkipper_JP_Fast2: A new day skipper that runs at fucking like 7000/hour.
- DenRoller and AutoHostRolling use a new method to roll the date.
    - There no longer a constraint on your starting date.
    - It rolls backwards first.

The idea here is that you'll be resetting the game on the target frame when you're done hosting that. So it's better to have your date correct at this point. In the past, if you forgot to fix the clock after rolling the den, every mon you caught would be in the future. Some of the timings for den rolling have also been tightened to make it faster.

- New Program: AutoHostSleep - Similar to AutoHostAirplane, but it uses sleep instead of airplane mode. Thus it works while docked. But techinical limitations with the method mean that the program will wait 30 seconds before it starts hosting. Similarly, the artificial delay between raids cannot be skipped for the first raid.

**2020-04-14:**

- EggHatcher and EggCombined now require that you have the cursor over "Pokémon" instead of the "Town Map".
- Adjusted some timings in EggHatcher and EggCombined to improve reliability. (Rowlett was failing with EggCombined.)
- Added SurpriseTrade: A super-dangerous program that will wonder trade entire boxes of Pokémon. :lul:

Brianuuu also dropped a new version of his Arduino programs which includes a lot of the stuff that we have here. Neither set of programs are strictly better than the other. Thus they will continue to complement each other.

**2020-04-12:**

- When connecting to the internet for auto-hosting, the cursor will be moved as far away as possible from the Link Trade and Surprise Trade buttons. This reduces the chances of a slow internet connection breaking the program in a way that initiates a trade.
- Add warnings that unattended auto-hosting is not recommended due to accidental trade risk.
- Improved documentation for all the delay parameters in the auto-hosts.

**2020-04-11:**

- Auto-Hosts: Auto FR-accepts is now delayed by a few seconds to keep it from clipping the start of the lobby.
- Auto-Hosts: The artificial delay between raids is suppressed for the first raid.
- Tightened some timings for DenRoller and AutoHostRolling.*
- EggCombined: Misc. reliability improvements.
- EggCombined: The default # fetches/batch has been increased to 6.

**2020-04-09:**

- Added EggCombined. It's harder to use than EggFetcher and EggHatcher, but can be a lot more efficient.
- Relaxed some timings in EggHatcher to make it more reliable.
- Relaxed some more timings in the auto-hosts to improve reliability.

**2020-04-06:**

- The FR-accepting auto-hosts have been merged into the regular ones as an optional feature. So it's back to two programs, AutoHostAirplane and AutoHostRolling.
- The per-raid extra delay option has been added to the rolling auto-host as well. But it only make sense to use if you're hard-locked instead of rolling.
- For the rolling auto-host, added the ability to select a 1st move. Now you can flex your hosting shiny!

**2020-04-05:**

- Relaxed the timings of FriendDelete.
- Add an option to the airplane auto-hosts to increase the time between raids. This makes farming more efficient when the raid cannot be cleared before the next raid starts.
- Add FriendDelete to the makefile comments.
- Clarified the instructions that you need to set USER_SLOT in order to pick the right account for auto FR-accepts.

**2020-04-04:**

- Tweaked the EggHatcher incubation period equation. Should fix issue with Lapras not having enough time to hatch.
- Add FriendDelete. Mass delete friends to clean up after auto-hosting with auto-FR accept.

The friend-deleter isn't super-well tested since it's a pain-in-the-ass to test.

**2020-04-03:**

- Added AutoHostAirplaneFRs. Now both of the auto-hosts have automatic FR-accepting versions.

- Minor code changes to everything.

**2020-04-02:**

- Added AutoHostRollingFRs: Rolling auto-host that will automatically accept friend requests.
- Slightly tweaked timings for EggFetcher, DenRoller, and both auto-hosts.

**2020-04-01:**

- Loosened some timings that can sometimes cause the auto-host programs to not connect to the internet.
- Loosened the timings for the box operations in the EggHatcher.
- The EggHatcher now travels further to the right for added safety.
- Speculative fix for an issue that can cause the egg programs to fail to fly back to Route 5.
- Updated comments and instructions including some precautions.

**2020-03-30:**

- Loosened some timings in DenRoller and the auto-hosts to make them more reliable. This helps fix some (recoverable) issues where it would fail to roll the correct # of times (and thus host the wrong frame) or will fail to enter a code.
- The EggFetcher has been retuned. It now properly dodges the man on the bridge and has a higher chance of successfully fetching an egg on each fly-back.
- EggHatcher timings have been updated to make it more reliable.

**2020-03-29:**

- EggHatcher now uses the spinning method. I strongly recommend that you stay offline. One of the failure cases may result in it entering YCOMM and starting a trade.

**2020-03-28b:**

- In the JPN skipper, the default auto-recovery period has been increased to 200 skips.
- Some refactoring of all the progs.
- Added EggFetcher and EggHatcher.

**2020-03-27:**

- Fixed a bug where 0s in the code for auto-host might not work.
- Added airplane auto-hosting for soft-locks.
- Tweaked timings for various things.

**2020-03-26:**

- Fixed a bug in the DaySkipper_JP_Fast where it didn't count days properly.
- Adjusted the timings for AutoHostRolling to make it more reliable.
- AutoHostRolling will now button mash A from start of raid to when it closes the game. This gives about ~10 seconds of leeway for people to be late without killing the raid.

**First Release:**

- DaySkipper_JP_Fast: A faster JPN skipper that runs up to 4700/hour.
- DenRoller: A more generic den roller that rolls N days instead of 3.
- AutoHostRolling: Roll N days, host raid, reset. Repeat.