Complete FireRed Upgrade



Contents

[Set Up 5](#_Toc20218697)

[Necessary Modifications 5](#_Toc20218698)

[Configurable Options 9](#_Toc20218699)

[Var Options 9](#_Toc20218700)

[Flag Options 11](#_Toc20218701)

[Start Menu Features 14](#_Toc20218702)

[Pedometer Flags 15](#_Toc20218703)

[Battle Tower Options 15](#_Toc20218704)

[Character Customization Vars 17](#_Toc20218705)

[Healing Place Hack 18](#_Toc20218706)

[TM / HM Options 18](#_Toc20218707)

[Times of Day 19](#_Toc20218708)

[Other Number Definitions 19](#_Toc20218709)

[Badge Obedience 21](#_Toc20218710)

[OW Palette Ids 21](#_Toc20218711)

[Pre-Battle Mugshot Options 22](#_Toc20218712)

[Safari Zone Options 22](#_Toc20218713)

[Randomizer Options 22](#_Toc20218714)

[Memory Locations 23](#_Toc20218715)

[Misc Features 23](#_Toc20218716)

[Misc Battle Effect Options 29](#_Toc20218717)

[Ability Options 29](#_Toc20218718)

[Damage Calculation Options 29](#_Toc20218719)

[Capturing Pokémon Options 29](#_Toc20218720)

[Exp. Gain Options 30](#_Toc20218721)

[Other Battle Options 30](#_Toc20218722)

[DexNav Options 31](#_Toc20218723)

[Engine Setup 33](#_Toc20218724)

[Mega Evolution / Primal Reversion / Ultra Burst 33](#_Toc20218725)

[Z-Moves 41](#_Toc20218726)

[Trainer Sliding Messages 42](#_Toc20218727)

[Multi Battles 44](#_Toc20218728)

[Wild Double Battles 46](#_Toc20218729)

[Trainer Backsprites 47](#_Toc20218730)

[Battle Terrain 49](#_Toc20218731)

[Battle Music 50](#_Toc20218732)

[Poke Balls 51](#_Toc20218733)

[Trainers With EVs 52](#_Toc20218734)

[Battle Frontier 54](#_Toc20218735)

[Upgraded TM/HM Expansion 55](#_Toc20218736)

[Reusable TMs 55](#_Toc20218737)

[Pickup 56](#_Toc20218738)

[Select from PC Hack 56](#_Toc20218739)

[Time of Day Based Wild Encounters 57](#_Toc20218740)

[Swarms 60](#_Toc20218741)

[Roaming Pokemon 60](#_Toc20218742)

[Day & Night System 61](#_Toc20218743)

[Pre-Battle Mugshots 64](#_Toc20218744)

[New Field Moves 65](#_Toc20218745)

[Other Features Included 67](#_Toc20218746)

[Save Expansion 67](#_Toc20218747)

[Updated & New Item Effects 67](#_Toc20218748)

[Trainer Face Fix 67](#_Toc20218749)

[DexNav 67](#_Toc20218750)

[Dynamic Overworld Palettes 67](#_Toc20218751)

[Ability Pop-Ups 67](#_Toc20218752)

[Hidden Abilities 68](#_Toc20218753)

[Expanded Text Names 68](#_Toc20218754)

[Pokédex Screen Stats 68](#_Toc20218755)

[Turbo Boost 68](#_Toc20218756)

[Various Customizable Updates 68](#_Toc20218757)

[Triple Layer Tiles10 68](#_Toc20218758)

[Expanded Coins 69](#_Toc20218759)

[Multichoice Windows 69](#_Toc20218760)

[Script Specials 71](#_Toc20218761)

[Creating New Battle Mechanics 109](#_Toc20218762)

[Moves 109](#_Toc20218763)

[Abilities 110](#_Toc20218764)

[Poke Balls 111](#_Toc20218765)

[Items 112](#_Toc20218766)

[Code Files 113](#_Toc20218767)

[Engine Scripts 121](#_Toc20218768)

[Clean.py 121](#_Toc20218769)

[String.py 121](#_Toc20218770)

[Credits 123](#_Toc20218771)

**Colour Coding Legend**

|  |  |
| --- | --- |
| **Encoding** | **Represents** |
| **Bold** | Indicates file paths or things to take note of. |
| *Italicized* | Used for things such as proper names or scripting command names. |
| Green | Used to indicate a definition that should be changed in **src/config.h**. In example scripts it is used for comments and strings. |
| Dull Orange | Used to indicate another constant defined somewhere (**not** in the config file). In scripts it is used for the #define and #org directives as well. |
| *Italicized Purple* | Used to indicate a table of some sort that can be modified (usually found in **src/Tables**, but not always). |
| Deep Orange | Used when referring to function names. |
| Gold | Used for certain definitions like #ifdef. |
| Blue Underlined | These words are hyperlinks. Clicking on them will take you somewhere (may or may not be in the document). |
| Red | Used for titles. In scripts they represent references to other scripts. |
| Faded Blue | Used for showing the inputs for script specials. |

# Set Up

## Necessary Modifications

**The following modifications must be made before setting up the engine!**

**Pokémon Defines**:

1. Open the files **include/constants/species.h** and **include/constants/pokedex.h**.
2. Modify the Pokémon indices found in this file to match the ones in your hack.
3. If you have not added in any new Pokémon to Fire Red, you can leave the unused species indices as their default values.
4. If you have added new Pokémon, ideally, you should not delete any Pokémon names and just assign all species you aren’t using a unique number starting after your last Pokémon slot. However, if this is too tedious for you, you’ll need to do the following two things.
5. First, modify the following tables by removing entries you don’t wish to use:

|  |  |
| --- | --- |
| **Table** | **File** |
| *sSpecialZMoveTable* | **src/set\_z\_effect.c** |
| *sSmartWildAITable* | **src/Battle\_AI/AI\_master.c** |
| *gWildSpeciesBasedBattleBGM* | **src/Tables/Music\_Tables.c** |
| All Tables | **src/Tables/Pokemon\_Tables.c** |

1. Secondly, the following Pokémon are necessary and must be changed if you are not using them (preferably all to unique numbers - I can’t guarantee they’ll all work as the same number):

|  |  |
| --- | --- |
| **Species** | **Reason** |
| SPECIES\_NONE 0x0 | It’s 0. **Don’t change or remove.** |
| SPECIES\_BULBASAUR 0x1 | Referred to by vanilla FR roaming code. |
| SPECIES\_CHARMANDER 0x4 | Referred to by vanilla FR roaming code. |
| SPECIES\_NIDORAN\_M 0x20 | Special breeding. |
| SPECIES\_FARFETCHD 0x53 | Signature item [Stick](https://bulbapedia.bulbagarden.net/wiki/Stick). |
| SPECIES\_CUBONE 0x68 | Signature item [Thick Club](https://bulbapedia.bulbagarden.net/wiki/Thick_Club). |
| SPECIES\_MAROWAK 0x69 | Signature item [Thick Club](https://bulbapedia.bulbagarden.net/wiki/Thick_Club). |
| SPECIES\_MAROWAK\_A 0x40F | Signature item [Thick Club](https://bulbapedia.bulbagarden.net/wiki/Thick_Club). |
| SPECIES\_CHANSEY 0x71 | Signature item [Lucky Punch](https://bulbapedia.bulbagarden.net/wiki/Lucky_Punch). [Luck Incense](https://bulbapedia.bulbagarden.net/wiki/Luck_Incense) breeding. |
| SPECIES\_MR\_MIME 0x7A | [Odd Incense](https://bulbapedia.bulbagarden.net/wiki/Odd_Incense) breeding. |
| SPECIES\_DITTO 0x84 | Signature items [Quick Powder](https://bulbapedia.bulbagarden.net/wiki/Quick_Powder) & [Metal Powder](https://bulbapedia.bulbagarden.net/wiki/Metal_Powder). |
| SPECIES\_SNORLAX 0x8F | [Full Incense](https://bulbapedia.bulbagarden.net/wiki/Full_Incense) breeding. |
| SPECIES\_PICHU 0xAC | [Volt Tackle](https://bulbapedia.bulbagarden.net/wiki/Volt_Tackle_(move)) breeding. |
| SPECIES\_MARILL 0xB7 | [Sea Incense](https://bulbapedia.bulbagarden.net/wiki/Sea_Incense) breeding. |
| SPECIES\_SUDOWOODO 0xB9 | [Rock Incense](https://bulbapedia.bulbagarden.net/wiki/Rock_Incense) breeding. |
| SPECIES\_WOBBUFFET 0xCA | [Lax Incense](https://bulbapedia.bulbagarden.net/wiki/Lax_Incense) breeding. |
| SPECIES\_UNOWN 0xC9 | Has many forms. Change those too. |
| SPECIES\_MANTINE 0xE2 | [Wave Incense](https://bulbapedia.bulbagarden.net/wiki/Wave_Incense) breeding. |
| SPECIES\_RAIKOU 0xF3 | Referred to by vanilla FR roaming code. |
| SPECIES\_ENTEI 0xF4 | Referred to by vanilla FR roaming code. |
| SPECIES\_SUICUNE 0xF5 | Referred to by vanilla FR roaming code. |
| SPECIES\_AZURILL 0x15E | [Sea Incense](https://bulbapedia.bulbagarden.net/wiki/Sea_Incense) breeding. |
| SPECIES\_WYNAUT 0x168 | [Lax Incense](https://bulbapedia.bulbagarden.net/wiki/Lax_Incense) breeding. |
| SPECIES\_ROSELIA 0x16B | [Rose Incense](https://bulbapedia.bulbagarden.net/wiki/Rose_Incense) breeding. |
| SPECIES\_CLAMPERL 0x175 | Signature items [Deep Sea Tooth](https://bulbapedia.bulbagarden.net/wiki/Deep_Sea_Tooth) & [Deep Sea Scale](https://bulbapedia.bulbagarden.net/wiki/Deep_Sea_Scale). |
| SPECIES\_CASTFORM 0x181 | Signature ability: [Forecast](https://bulbapedia.bulbagarden.net/wiki/Forecast_(Ability)). |
| SPECIES\_VOLBEAT 0x182 | Special breeding. |
| SPECIES\_RAYQUAZA 0x196 | Banned with [Dragon Ascent](https://bulbapedia.bulbagarden.net/wiki/Dragon_Ascent_(move)) in a Battle Tower [format](https://www.smogon.com/dex/sm/formats/uber/). |
| SPECIES\_LATIAS 0x197 | Signature item [Soul Dew](https://bulbapedia.bulbagarden.net/wiki/Soul_Dew). |
| SPECIES\_LATIOS 0x198 | Signature item [Soul Dew](https://bulbapedia.bulbagarden.net/wiki/Soul_Dew). |
| SPECIES\_CHIMECHO 0x19B | [Pure Incense](https://bulbapedia.bulbagarden.net/wiki/Pure_Incense) breeding. |
| SPECIES\_EGG 0x19C | It’s an egg. **Don’t change or remove.** |
| SPECIES\_BUDEW 0x1CB | [Rose Incense](https://bulbapedia.bulbagarden.net/wiki/Rose_Incense) breeding. |
| SPECIES\_BURMY 0x1D1 | Changes form after battle. |
| SPECIES\_BURMY\_SANDY 0x2C3 | Changes form after battle. |
| SPECIES\_BURMY\_TRASH 0x2C4 | Changes form after battle. |
| SPECIES\_CHERRIM 0x1DA | Signature ability: [Flower Gift](https://bulbapedia.bulbagarden.net/wiki/Flower_Gift_(Ability)). |
| SPECIES\_CHERRIM\_SUN 0x2EF | Signature ability: [Flower Gift](https://bulbapedia.bulbagarden.net/wiki/Flower_Gift_(Ability)). |
| SPECIES\_CHINGLING 0x1E6 | [Pure Incense](https://bulbapedia.bulbagarden.net/wiki/Pure_Incense) breeding. |
| SPECIES\_BONSLY 0x1EB | [Rock Incense](https://bulbapedia.bulbagarden.net/wiki/Rock_Incense) breeding. |
| SPECIES\_MIME\_JR 0x1EC | [Odd Incense](https://bulbapedia.bulbagarden.net/wiki/Odd_Incense) breeding. |
| SPECIES\_HAPPINY 0x1ED | [Luck Incense](https://bulbapedia.bulbagarden.net/wiki/Luck_Incense) breeding. |
| SPECIES\_MUNCHLAX 0x1F3 | [Full Incense](https://bulbapedia.bulbagarden.net/wiki/Full_Incense) breeding. |
| SPECIES\_MANTYKE 0x1FF | [Wave Incense](https://bulbapedia.bulbagarden.net/wiki/Wave_Incense) breeding. |
| SPECIES\_ROTOM 0x214 | Special breeding. |
| SPECIES\_DIALGA 0x218 | Signature item [Adamant Orb](https://bulbapedia.bulbagarden.net/wiki/Adamant_Orb). |
| SPECIES\_PALKIA 0x219 | Signature item [Lustrous Orb](https://bulbapedia.bulbagarden.net/wiki/Lustrous_Orb). |
| SPECIES\_GIRATINA 0x21C | Signature item [Griseous Orb](https://bulbapedia.bulbagarden.net/wiki/Griseous_Orb). |
| SPECIES\_GIRATINA\_ORIGIN 0x2CE | Signature item [Griseous Orb](https://bulbapedia.bulbagarden.net/wiki/Griseous_Orb). |
| SPECIES\_PHIONE 0x21E | Special breeding from [Manaphy](https://bulbapedia.bulbagarden.net/wiki/Manaphy_(Pok%C3%A9mon)). |
| SPECIES\_DARKRAI 0x220 | Only species that can use [Dark Void](https://bulbapedia.bulbagarden.net/wiki/Dark_Void_(move)). |
| SPECIES\_SHAYMIN 0x221 | Changes [form](https://bulbapedia.bulbagarden.net/wiki/Shaymin_(Pok%C3%A9mon)) when frozen. |
| SPECIES\_SHAYMIN\_SKY 0x2CF | Changes [form](https://bulbapedia.bulbagarden.net/wiki/Shaymin_(Pok%C3%A9mon)) when frozen. |
| SPECIES\_DARMANITAN 0x260 | Signature ability: [Zen Mode](https://bulbapedia.bulbagarden.net/wiki/Zen_Mode_(Ability)). |
| SPECIES\_DARMANITANZEN 0x2E1 | Signature ability: [Zen Mode](https://bulbapedia.bulbagarden.net/wiki/Zen_Mode_(Ability)). |
| SPECIES\_KELDEO 0x2BC | Changes form with [Secret Sword](https://bulbapedia.bulbagarden.net/wiki/Secret_Sword_(move)). |
| SPECIES\_KELDEO\_RESOLUTE 0x2F5 | Changes form if doesn’t know [Secret Sword](https://bulbapedia.bulbagarden.net/wiki/Secret_Sword_(move)). |
| SPECIES\_MELOETTA 0x2BD | Changes form with [Relic Song](https://bulbapedia.bulbagarden.net/wiki/Relic_Song_(move)). |
| SPECIES\_MELOETTA\_PIROUETTE 0x2EA | Changes form with [Relic Song](https://bulbapedia.bulbagarden.net/wiki/Relic_Song_(move)). |
| SPECIES\_GRENINJA 0x2FE | Signature ability: [Battle Bond](https://bulbapedia.bulbagarden.net/wiki/Battle_Bond_(Ability)). |
| SPECIES\_ASHGRENINJA 0x347 | Signature ability: [Battle Bond](https://bulbapedia.bulbagarden.net/wiki/Battle_Bond_(Ability)). |
| SPECIES\_FURFROU 0x310 | Special breeding. |
| SPECIES\_AEGISLASH 0x315 | Signature ability: [Stance Change](https://bulbapedia.bulbagarden.net/wiki/Stance_Change_(Ability)). |
| SPECIES\_AEGISLASH\_BLADE 0x341 | Signature ability: [Stance Change](https://bulbapedia.bulbagarden.net/wiki/Stance_Change_(Ability)). |
| SPECIES\_ZYGARDE 0x33A | Signature ability: [Power Construct](https://bulbapedia.bulbagarden.net/wiki/Power_Construct_(Ability)). |
| SPECIES\_ZYGARDE\_10 0x345 | Signature ability: [Power Construct](https://bulbapedia.bulbagarden.net/wiki/Power_Construct_(Ability)). |
| SPECIES\_ZYGARDE\_COMPLETE 0x346 | Signature ability: [Power Construct](https://bulbapedia.bulbagarden.net/wiki/Power_Construct_(Ability)). |
| SPECIES\_HOOPA 0x33C | Special message why trying to use [Hyperspace Fury](https://bulbapedia.bulbagarden.net/wiki/Hyperspace_Fury_(move)). |
| SPECIES\_HOOPA\_UNBOUND 0x33D | Only Pokémon that can use [Hyperspace Fury](https://bulbapedia.bulbagarden.net/wiki/Hyperspace_Fury_(move)). |
| SPECIES\_GROUDON\_PRIMAL 0x38D | Is referenced as a *Red Primal*. |
| SPECIES\_KYOGRE\_PRIMAL 0x38E | Is referenced as a *Blue Primal*. |
| SPECIES\_LYCANROC 0x3C2 | Special battle animation. |
| SPECIES\_LYCANROC\_N 0x416 | Special battle animation. |
| SPECIES\_LYCANROC\_DUSK 0x43A | Special battle animation. |
| SPECIES\_WISHIWASHI 0x3C3 | Signature ability: [Schooling](https://bulbapedia.bulbagarden.net/wiki/Schooling_(Ability)). |
| SPECIES\_WISHIWASHI\_S 0x417 | Signature ability: [Schooling](https://bulbapedia.bulbagarden.net/wiki/Schooling_(Ability)). |
| SPECIES\_MINIOR\_SHIELD 0x3DF | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_RED 0x429 | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_BLUE 0x42A | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_ORANGE 0x42B | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_YELLOW 0x42C | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_INDIGO 0x42D | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_GREEN 0x42E | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MINIOR\_VIOLET 0x42F | Signature ability: [Shields Down](https://bulbapedia.bulbagarden.net/wiki/Shields_Down_(Ability)). |
| SPECIES\_MIMIKYU 0x3E3 | Signature ability: [Disguise](https://bulbapedia.bulbagarden.net/wiki/Disguise_(Ability)). |
| SPECIES\_MIMIKYU\_BUSTED 0x430 | Signature ability: [Disguise](https://bulbapedia.bulbagarden.net/wiki/Disguise_(Ability)). |

Also be sure to change the species indices in **asm\_defines** as well (just copy and paste the data from before but search and replace all #define with .equ and all " 0x" with "," (without quotation marks).

**Item Defines**:

Part 1

1. Open the file **include/constants/items.h**.
2. Modify the item indices found in this file to match the ones in your hack.
3. **DO NOT DELETE ANY ITEM NAMES**. If there is an item you are not using, then set its index to **0xFEFE**.

Part 2

1. Open the file **include/constants/hold\_effects.h**.
2. Scroll down to where it says “//NEW ITEM EFFECTS”. This is the list of new hold item effects. When relevant, the item quality is included.
3. Set up your items in G3T using the example for the *Fire Gem* as a base:



As you can see, the item effect is set to 94 and the item quality is set to 10 (the move type for *Fire*).

**Configuration Options**

See below.

## Configurable Options

There are many configurable options in the file **src/config.h**. These options are meant to provide the user with as much versatility as possible. Below is a more detailed description of each option than show in the configuration file:

### Var Options

|  |  |
| --- | --- |
| *Flag Definition* | *Description* |
| TERRAIN\_VAR | Setting this var to one of the following values before initiating a battle will load the battlefield with the corresponding [terrain](https://bulbapedia.bulbagarden.net/wiki/Terrain):  1: Electric Terrain  2. Grassy Terrain  3. Misty Terrain  4. Psychic Terrain |
| TOTEM\_VAR | This represents are series of vars using for initiating battles with [Totem Pokémon](https://bulbapedia.bulbagarden.net/wiki/Totem_Pok%C3%A9mon). There are four vars in total, each representing a specific Pokémon slot on the field. Adding the following values to the var will indicate which slot that var is for:  0: *Player Pokémon in Singles, Left Player Pokémon in Doubles*  1: *Enemy Pokémon in Singles, Right Enemy Pokémon in Doubles*  2: *Right Player Pokémon in Doubles*  3: *Left Enemy Pokémon in Doubles*  The vars must be set to the addition of two values. Choose one from each of the following sets:  **Stats**:  1: *Attack*  2: *Defense*  3: *Speed*  4: *Special Attack*  5: *Special Defense*  6: *Accuracy*  7: *Evasion*  **Amount**:  0x10: *Increase Stat by 1*  0x20: *Increase Stat by 2*  0x30: *Increase Stat by 3*  0x40: *Increase Stat by 4*  0x50: *Increase Stat by 5*  0x60: *Increase Stat by 6*  0x90: *Decrease Stat by 1*  0xA0: *Decrease Stat by 2*  0xB0: *Decrease Stat by 3*  0xC0: *Decrease Stat by 4*  0xD0: *Decrease Stat by 5*  0xE0: *Decrease Stat by 6*  So, for instance, in a single battle, having the enemy Pokémon start the battle with its *Attack* raised by *2*, you would set the var TOTEM\_VAR + 1 to the value of 0x21 (0x1 + 0x20).  Additionally, this feature can also be used in trainer battles. If used here, any Pokémon sent out into the set position will have their stats raised. So, using the example from above, all the opposing trainer’s Pokémon would start with their A*ttack* raised by *2*. |
| NPC\_FOLLOWING\_VAR | If the *Follow Me* feature is used, this var must be set to the *NPC Id* (*Person Id* in *AdvanceMap*) of the following NPC. If no NPC is following the player, this var should be set to 0. |
| BACKSPRITE\_SWITCH\_VAR | Setting this var to a value other than 0 will change the default back sprite loaded for the player in battle.  See **src/Tables/ Backsprite\_Tables.c** for a list of available backsprites. |
| BATTLE\_BG\_VAR | If CUSTOM\_BATTLE\_BACKGROUNDS is defined, setting this var to a value other than 0 will cause the regular battle background loaded to be replaced by a custom one.  See **/include/battle.h** for a list of options.  Search for BATTLE\_TERRAIN\_GRASS in the file to see them. |
| SWARM\_INDEX\_VAR | A var that is automatically set by the engine. It contains the index in the swarming table of the species that is currently [swarming](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_outbreak). Swarms are set to change on a daily basis. If TIME\_ENABLED is commented out, then the code will need to be modified to find an alternative method to enable swarms. It can be found in **src/wild\_encounter.c**.  The swarming table can be edited by searching for *gSwarmTable* in **src/Tables/Wild\_Encounter\_Tables.c**. |
| SWARM\_DAILY\_EVENT\_VAR | A pair of two vars (this one and the one immediately following it) that are automatically set by the engine. They are used to determine if a swarm has already been chosen for the given date. If TIME\_ENABLED is commented out, then these vars will be set once and never again. |
| DEFAULT\_WALKING\_SCRIPT\_VAR | A var the contains the index of the script in walking script table that is set to run on every step. Set this var to 0 if you do not want any custom scripts to be run each step. To add a walking script, search for *gDefaultWalkingScripts* in **src/overworld.c** and either add *(const u8\*)* pointers to scripts preloaded in the rom, or define your own script in one of the assembly files. |
| DEXNAV\_VAR | A var that holds the species to search for in the Overworld via the *DexNav* feature. Press *Select* in the DexNav GUI to save. |
| STATUS\_INDUCER\_VAR | A var that if set, causes wild and Trainer Pokémon to be generated with the set status condition. It is split into an upper and lower byte. If the upper byte is set to 0, then this status condition will be given to Pokémon until this var is cleared. Any value in the upper byte other than 0 acts as a timer and is subtracted by 1 for each battle completed. For example, setting this var to *0x0640* will paralyze all enemy Pokémon in battles for 6 battles. See [here](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_data_structure_in_Generation_III#Status_condition) for a list of status bytes. |
| SECOND\_OPPONENT\_VAR | A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xB*) to represent the trainer id of the second trainer in battles against two opponents. If set manually in conjunction with TWO\_OPPONENT\_FLAG, a battle against two opponents will be started the next time a trainer battle is initiated. |
| PARTNER\_VAR | A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xC*) to represent the trainer id of the partner trainer in multi battles. If set manually in conjunction with TAG\_BATTLE\_FLAG, a battle with a partner against a single trainer will be started the next time a trainer battle is initiated. If setting manually, take care to set PARTNER\_BACKSPRITE\_VAR as well. |
| PARTNER\_BACKSPRITE\_VAR | A var that can be set by the engine (with *trainerbattle 0xA* or *trainerbattle 0xC*) to represent the backsprite id of the partner trainer in multi battles. If setting manually, take care to also set PARTNER\_VAR and TAG\_BATTLE\_FLAG. |

### Flag Options

**NOTE:** Many of the following flags are cleared at the end of battle. To remove this, open the file **src/end\_battle.c** and remove the flag from *gEndBattleFlagClearTable*.

|  |  |
| --- | --- |
| *Flag Definition* | *Description* |
| INVERSE\_FLAG | Setting this flag will enable [Inverse Battles](https://bulbapedia.bulbagarden.net/wiki/Inverse_Battle).  This flag is automatically cleared at the end of each battle. |
| SKY\_BATTLE\_FLAG | Setting this flag will indicate to the engine that a [Sky Battle](https://bulbapedia.bulbagarden.net/wiki/Sky_Battle) is in progress.  This flag is automatically cleared at the end of each battle. |
| NO\_CATCHING\_FLAG | Setting this flag will cause enemy Pokémon to always dodge balls thrown at them.  This flag is automatically cleared at the end of each battle. |
| NO\_RUNNING\_FLAG | Setting this flag prevents the player from running away during wild battles.  This flag is automatically cleared at the end of each battle. |
| NO\_CATCHING\_AND\_RUNNING\_FLAG | This flag acts as a combination of the above two flags.  This flag is automatically cleared at the end of each battle. |
| CATCH\_TRAINERS\_POKÉMON\_FLAG | Setting this flag allows the player to capture Pokémon belonging to the opposing trainer. Capturing a Pokémon in this way will automatically end the battle. Comment out this line if you do not want to use this feature.  This flag is automatically cleared at the end of each battle. |
| EXP\_SHARE\_FLAG | If OLD\_EXP\_SHARE is commented out, then setting this flag activates the Gen 6+ Exp. Share. |
| DOUBLE\_BATTLE\_FLAG | Setting this flag will cause battles against trainers to be [Double Battles](https://bulbapedia.bulbagarden.net/wiki/Double_Battle), if possible. Comment out this line if you don’t want to use this flag. |
| TAG\_BATTLE\_FLAG | This flag is set by the engine when the scripting command *trainerbattle 0xA* or *trainerbattle 0xC* is used in a script to activate a tag battle. If setting this flag manually, take care to also set PARTNER\_VAR and PARTNER\_BACKSPRITE\_VAR. This flag is automatically cleared at the end of each battle. |
| TWO\_OPPONENT\_FLAG | This flag is set by the engine when the scripting command *trainerbattle 0xA* or *trainerbattle 0xB* is used in a script to activate a battle against two opponents. If setting this flag manually, take care to also set SECOND\_OPPONENT\_VAR. This flag is automatically cleared at the end of each battle. |
| ACTIVATE\_TUTORIAL\_FLAG | If TUTORIAL\_BATTLES is defined, Setting this flag activates Professor Oak’s tutorial during the next trainer battle. This flag is automatically cleared at the end of each battle. Comment out this line if you don’t want to use this flag. |
| WILD\_CUSTOM\_MOVES\_FLAG | Setting the flag before a wild battle starts will create the wild Pokémon with the moves given in the input vars. This works with both regular wild battles and scripted wild battles (if scripted, set the input vars before using the *wildbattle* scripting command). Setting any value to 0xFFFF will cause the default move to be loaded in that slot. Setting any value to 0x0 will load a blank move in that slot. Note that there are additional inputs for wild double battles. The input is as follows:  Var 0x8000: **Move 1** - Pokémon **1**  Var 0x8001: Move 2 - Pokémon **1**  Var 0x8002: Move 3 - Pokémon **1**  Var 0x8003: Move 4 - Pokémon **1**  Var 0x8004: **Move 1** - Pokémon **2** (Wild Double)  Var 0x8005: Move 2 - Pokémon **2** (Wild Double)  Var 0x8006: Move 3 - Pokémon **2** (Wild Double)  Var 0x8007: Move 4 - Pokémon **2** (Wild Double) |
| SMART\_WILD\_FLAG | Setting this flag allows wild Pokémon to use the basic AI checks used in trainer battles. This flag is automatically cleared at the end of each battle. |
| SCALE\_WILD\_POKEMON\_LEVELS\_FLAG | Setting this flag will cause all **random** wild Pokémon encounters (this does not include scripted encounters) to have Pokémon with levels that match the lowest level in your party. Comment out this line if you don’t want to use this feature. |
| SCALE\_TRAINER\_LEVELS\_FLAG | Setting this flag causes all Trainer Pokémon to have levels that match the highest level in your party. |
| HIDDEN\_ABILITY\_FLAG | Setting this flag before a Wild battle causes Wild Pokémon to be generated with their [hidden abilities](https://bulbapedia.bulbagarden.net/wiki/Ability#Hidden_Abilities). It is cleared at the end of each battle. It also lets the *givepokemon* scripting command give Pokémon with their hidden abilities. |
| DOUBLE\_WILD\_BATTLE\_FLAG | Setting this flag causes all wild battles to be against two wild Pokémon in a Double battle format (if the player has at least two viable Pokémon on their team). It is cleared at the end of each battle. |
| NO\_RANDOM\_WILD\_ENCOUNTERS\_FLAG | Setting this flag will stop Pokémon from appearing while walking through grass or caves, or while surfing on water. Pokémon can still appear if the player chooses to fish, smash rocks, or use *Sweet Scent*. |
| FLAG\_REMOVE\_EVO\_ITEM | A flag set by the engine to help with certain item-based evolutions. |
| SHINY\_CREATION\_FLAG | The next Pokémon generated by the engine will be shiny. This includes wild Pokémon, gift Pokémon, or the first Pokémon in the next trainer battle. It is automatically cleared at the end of each battle. |
| AUTO\_RUN\_FLAG | Setting this flag enables auto-run. For convenience, this flag is toggled in the overworld by pressing the *L-Button* (as long as the button setting is not set to *L=*A). By default, this flag is defined in the config to be the same as the flag that allowed running in vanilla FR. |
| RUNNING\_ENABLED\_FLAG | If this line is uncommented, this flag can be used to control when the player can run. If this flag is not set, the player will be forced to walk. |
| DISABLE\_BAG\_FLAG | Setting this flag prevents the bag from being utilized in-battle. |
| MOVE\_RELEARNER\_IGNORE\_LEVEL\_FLAG | Setting this flag allows the [Move Reminder](https://bulbapedia.bulbagarden.net/wiki/Move_Reminder) to show a full list of moves Pokémon can learn through level-up (not restricted by the Pokémon’s level). |
| EGG\_MOVE\_RELEARNER\_FLAG | Setting this flag allows the *Move Reminder* to teach Egg Moves instead. |

### Start Menu Features

Any of the following flags can be commented out to remove them from the engine. If any of the flags are defined and not set, their respective start menu item will be hidden by default.

|  |  |
| --- | --- |
| *Definition* | *Description* |
| FLAG\_SYS\_BAG\_HIDE | This flag allows the hacker to toggle the *bag* in the start menu, for events where the player isn’t allowed to use items, or lost their bag for various reasons. If this is commented out, *BAG* will always be present on the start menu. (Set to Hide) |
| FLAG\_SYS\_PLAYER\_HIDE | This allows the hacker to toggle on/off the *Trainer Card* from the start menu. Commenting this out will cause *PLAYER* to always be present. (Set to Hide) |
| FLAG\_SYS\_SAVE\_ HIDE | This allows the hacker to toggle the *save game* feature from the start menu. Commenting this out causes *SAVE* to be permanent on the start menu. (Set to Hide) |
| FLAG\_SYS\_DEXNAV | This allows the hacker to toggle *TOOLS* from the start menu. If this flag is defined and not set, *POKéDEX* will show up on the start menu. When the flag is set, *TOOLS* will replace *POKéDEX,* which yields a separate menu including both *POKéDEX* and *DEXNAV*. If this is commented out, The [DexNav](#_Dex_Nav) feature will be inaccessible. |
| FLAG\_POKETOOLS\_MENU | This flag causes *TOOLS* to open a separate start menu as opposed to the multichoice list generated by default. This flag is purely for aesthetic purposes, although an advanced hacker could use this to create two separate start menus. If this is commented out, the default *TOOLS* multichoice will load (see above). |

### Pedometer Flags

Setting any of these flags will initiate a pedometer of the corresponding size. The pedometer value can be read using *special 0x8A*.

|  |  |
| --- | --- |
| *Definition* | *Description* |
| FLAG\_LONG\_PEDOMETER | 4 byte pedometer (max value 0xFFFFFFFF or 4 294 967 295) |
| FLAG\_MED\_PEDOMETER | 2 byte pedometer (max value 0xFFFF or 65 535) |
| FLAG\_SMALL\_PEDOMETER\_1 | 1 byte pedometer (max value 0xFF or 255) |
| FLAG\_SMALL\_PEDOMETER\_2 | 1 byte pedometer (max value 0xFF or 255) |

### Battle Tower Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| BATTLE\_TOWER\_FLAG | Setting this flag indicates to the engine that the Player is in a [Battle](https://bulbapedia.bulbagarden.net/wiki/Battle_Tower) Facility. This means that:   * Trainer Pokémon will be generated within the restrictions of the tier set in the var defined in BATTLE\_TOWER\_TIER. The amount of Pokémon generated will match the number set in the var defined in BATTLE\_TOWER\_POKE\_LEVEL. * Trainer Pokémon will have Pokémon with the level contained in the var defined in BATTLE\_TOWER\_POKE\_LEVEL. * The battle format will be loaded from the var defined in BATTLE\_TOWER\_BATTLE\_TYPE. * The music in-battle will be played based on what is contained in the var defined in BATTLE\_TOWER\_SONG\_OVERRIDE. * The bag will be inaccessible in battle. |
| BATTLE\_TOWER\_POKE\_NUM | Setting this var to a value between 1 & 6 dictates the quantity of Pokémon the player and trainers can use in the Battle Tower.  Setting it to 0 will default in 1.  Setting it to a number greater than 6 will default in 6. |
| BATTLE\_TOWER\_POKE\_LEVEL | Setting this var to a value between 1 and what is defined in MAX\_LEVEL will set all Pokémon in the Battle Tower to that level.  Setting it to 0 will default in 1.  Setting to a number greater than MAX\_LEVEL will default in MAX\_LEVEL. |
| BATTLE\_TOWER\_BATTLE\_TYPE | Setting this var to one of the below values set the battle format in the Battle Tower:  0: Single Battle  1: Double Battle  2: Multi Battle  3: Link Multi Battle  4: Random Single Battle  5: Random Double Battle  6: Random Multi Battle  Any other value defaults in Single Battle. |
| BATTLE\_TOWER\_TIER | Setting this var to one of the below values indicates to the engine which ruleset should be following for battles in the Battle Tower:  0: Regular Battle Tower Rules  1: No Restrictions  2: [Smogon OU](https://www.smogon.com/dex/sm/formats/ou/)  3: [Smogon Uber](https://www.smogon.com/dex/sm/formats/uber/)  4: [Smogon Little Cup](https://www.smogon.com/dex/sm/formats/lc/)  5: Skeli’s Middle Cup / GS Cup  6: Smogon Monotype  7: Smogon Camomons  8: Uber Camomons  9: Little Cup Camomons  10: Middle Cup Camomons / GS Cup Camomons  11: Smogon Scalemons  12: Smogon 350 Cup  13: Smogon Averagemons  14: Smogon Benjamin Butterfree  15: Battle Mine Format 1 (OU, Camomons, Benjamin Butterfree)  16: Battle Mine Format 2 (Scalemons, 350 Cup, Averagemons)  17: Battle Mine Format 3 (Little Cup, Little Cup Camomons) |
| BATTLE\_TOWER\_TRAINER1\_NAME | This var is automatically set by the engine to hold the index of the random name for the first Battle Tower trainer. It is set to 0xFFFF after every battle. Do not set it manually. |
| BATTLE\_TOWER\_TRAINER2\_NAME | This var is automatically set by the engine to hold the index of the random name for the second Battle Tower trainer in Multi Battles. It is set to 0xFFFF after every battle. Do not set it manually. |
| BATTLE\_TOWER\_SONG\_OVERRIDE | Setting this var to a song Id will cause that song to be played in *Battle Tower* battles and *Link Battles*. |
| TOWER\_TRAINER\_ID\_VAR | This var is set by *special 0x52* to indicate which trainer class and details is being spawned as the first opponent. The frontier trainer details should be added to the *gTowerTrainers* table which can be found in **src/Tables/Frontier\_Trainers.c**. |
| TOWER\_TRAINER\_ID\_2\_VAR | This var is set by *special 0x52* to indicate which trainer class and details is being spawned as the second opponent in multi battles. The frontier trainer details should be added to the *gTowerTrainers* table which can be found in **src/Tables/Frontier\_Trainers.c**. |
| TOWER\_TRAINER\_ID\_PARTNER\_VAR | This var is set by *special 0x52* to indicate which trainer class and details is being spawned as the player’s partner in multi battles if the partner is chosen to be randomized. The frontier trainer details should be added to the *gTowerTrainers* table which can be found in **src/Tables/Frontier\_Trainers.c**. |

### Character Customization Vars

|  |  |
| --- | --- |
| *Definition* | *Description* |
| VAR\_PLAYER\_WALKRUN | Set this var to change the player’s walking/running overworld sprite frames. The upper byte is used as the table Id, For example, setting to 0x0200 will load the walking/running frames from table 2, sprite 0. |
| VAR\_PLAYER\_BIKING | Switch player biking frames (same rules as above). |
| VAR\_PLAYER\_SURFING | Switch player surfing frames (same rules as above). |
| VAR\_PLAYER\_VS\_SEEKER | Switch player VS Seeker frames (same rules as above). |
| VAR\_PLAYER\_FISHING | Switch player Fishing frames (same rules as above). |
| VAR\_PLAYER\_VS\_SEEKER\_ON\_BIKE | Switch player Biking/*VS Seeker* frames (same rules as above). |
| VAR\_TRAINERCARD\_MALE | Set this var to the trainer sprite id of the male player front sprite that appears on the trainer card. |
| VAR\_TRAINERCARD\_FEMALE | Set this var to the trainer sprite id of the female player front sprite that appears on the trainer card. |
| VAR\_RUNTIME\_CHANGEABLE | If a person event has a given overworld table id 0xFF, it can be changed at runtime by changing these variables to a sprite number. There are 15 variables used in total.  For example, setting VAR\_RUNTIME\_CHANGEABLE+2 to 16, will cause all NPCs with ids 0xFF02 to appear with the little boy overworld sprite (in vanilla FR).    #define VAR\_RUNTIME\_CHANGEABLE 0x4080  #org 0x800000  setvar VAR\_RUNTIME\_CHANGEABLE+2 16 'Var 0x4082 |

### Healing Place Hack

The following vars relate to JPAN’s healing place hack. If SET\_HEALING\_PLACE\_HACK is not defined, ignore these vars.

|  |  |
| --- | --- |
| VAR\_HEALINGMAP | Set this var to the map and bank for the player to respawn to after whiting out. For example, if it is set to 0x0104, the player will respawn in their room (vanilla FR) (map bank 4, map 1). Configure in **asm\_defines.s**. |
| VAR\_HEALING\_XPOS | Set this var to the x-position the player will respawn at on the map in VAR\_HEALINGMAP. |
| VAR\_HEALING\_YPOS | Set this var to the y-position the player will respawn at on the map in VAR\_HEALINGMAP. |

### TM / HM Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| EXPANDED\_TMS | Allows TMs and HMs past the original 58 to be used up to 128 total TMs + HMs. |
| EXPANDED\_MOVE\_TUTORS | Allows Move Tutors past the original 16 up to 128. This also removes the restriction that only allows the powerful elemental moves to be learned by the Kantonian starters. |
| NUM\_TMS | Set this to the total number of TMs in the game. |
| NUM\_HMS | Set this to the total number of HMs in the game. |
| TMS\_BEFORE\_HMS | Uncomment this line if you’d like HMs to appear at the end of the TM list in the bag. |
| NUM\_MOVE\_TUTORS | The number of move tutors. 64 is a good number. |
| LAST\_TOTAL\_TUTOR\_NUM | Should be equal to (NUM\_MOVE\_TUTORS - 1) + 8. Must be set to an actual integer or the compilation will not work. This allows special additional move tutors to be added without having to change the size of the compatibility table. These are:  1. Draco Meteor  2. Secret Sword  3. Relic Song  4. Volt Tackle  5. Dragon Ascent  6. Thousand Arrows  7. Thousand Waves  8. Core Enforcer  For more information regarding the move tutor defines, see **include/constants/tutors.h**. |
| TMS\_BEFORE\_HMS | Uncomment this line if you want the HMs to appear after the TMs in your bag. |
| DELETABLE\_HMS | Uncomment this line if you’d like HMs to be deleted without the use of the *Move Deleter* like normal moves. |
| REUSABLE\_TMS | Allows TMs to be reused infinitely without being removed from the bag. Also prevents TMs from being sold, held, or bought more than once. If using this feature, don’t forget to assign all TMs a *Mystery 1* (in G3T) value of 1! |

### Times of Day

|  |  |
| --- | --- |
| *Definition* | *Description* |
| TIME\_MORNING\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that morning starts. This is also the day start time used for many events that only have a daytime/nighttime variant. |
| TIME\_DAY\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that day starts. |
| TIME\_EVENING\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that evening starts. |
| TIME\_NIGHT\_START | If TIME\_ENABLED is defined, set this to the hour (in 24 hr system) that night starts. |

### Other Number Definitions

|  |  |
| --- | --- |
| *Definition* | *Description* |
| KANTO\_DEX\_COUNT | Number of Pokémon in the regional Pokedex. |
| NATIONAL\_DEX\_COUNT | Number of Pokémon in the national Pokedex. |
| MAX\_LEVEL | The highest possible level for a Pokémon. If you change this value, make sure to also modify the equivalent value found in “[special\_inserts.asm](https://github.com/Skeli789/Complete-Fire-Red-Upgrade/blob/master/special_inserts.asm)”. |
| NUM\_TRAINER\_CLASSES | The number of trainer classes. Vanilla FR has 107. |
| EVOS\_PER\_MON | If you’ve changed the number of evolutions per Pokémon, update this number. Vanilla FR has 5. |
| EV\_CAP | The most EVs a Pokémon can accrue for a given stat. |
| POWER\_ITEM\_EV\_YIELD | The amount of additional EVs gifted in a certain stat by Power Items such as the [Power Bracer](https://bulbapedia.bulbagarden.net/wiki/Power_Bracer). |
| DUSK\_BALL\_MULTIPLIER | The catch rate (\*10) for [Dusk Balls](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9_Ball#Dusk_Ball). So *30* is 3.0. |
| STANDARD\_IV | The number of IVs for each stat that standard Trainer’s Pokémon are generated with. |
| SWARM\_CHANCE | The chance in percent that a [swarm](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_outbreak) Pokémon will appear on a route if there is currently a swarm in progress on that route. |
| WILD\_DOUBLE\_RANDOM\_CHANCE | The chance that a wild double will be initiated if the player is walking in grass with a background byte with its 4th bit set. For instance, grass with a background byte of 0x5 will have a chance of starting a wild double battle, and grass with a background byte of 0x25 will have a chance of starting a wild double battle and be covered by the player (water is similarly 0x6 and 0x26). |
| CREATE\_WITH\_X\_PERFECT\_IVS | Set this to the number of 31 IVs Pokémon defined in the *gSetPerfectXIvList* should be generated with. For example, if Mewtwo is in the table, and this is defined to 3, any Mewtwo generated (wild or Trainer) will always have at least 3 IVs set to 31. This does not include roaming Pokémon. |
| CREATE\_ROAMER\_WITH\_X\_PERFECT\_IVS | Set this to the number of 31 IVs roaming Pokémon should be generated with. |
| EGG\_HATCH\_LEVEL | Set this to level Pokémon *Eggs* should hatch at. Before Gen 4 it was *5*, from Gen 4 onwards it has been *1*. |
| AI\_TRY\_TO\_KILL\_RATE | In battles against a trainer with AI flags of *1* (only check if the move shouldn’t be used), the AI will try to use a move to knock out the opponents XX percent of the time, where XX is the number defined here. Setting this to 0 means basic AI will always use random moves in-battle, assuming that move has no reason not to be used. |
| MB\_OMNIDIRECTIONAL\_JUMP | Setting a tile’s behaviour byte to this number makes the tile act as a 4 way ledge. Meaning, the player will jump over it when approaching it from all 4 directions. For a related option, see CAN\_ONLY\_USE\_OMNIDRECTIONAL\_JUMP\_  ON\_HEIGHT\_2. |
| MB\_ROCK\_CLIMB\_WALL | Setting a tile’s behaviour byte to this number makes it climbable with the move [Rock Climb](https://bulbapedia.bulbagarden.net/wiki/Rock_Climb_(move)). See [New Field Moves](#_New_Field_Moves) for more instructions. |
| MAP\_PLAYER\_HOME | The map bank and map number of the player's home. This is used to help load the correct message when the Player *white’s out*. The upper byte represents the map bank, while the lower byte represents the map number. So:    Means *map bank 4* and *map number 0*, aka, the Player’s home in vanilla FR. |
| MAX\_COINS\_DIGITS | Number of digits of game corner coins the player can hold (eg. default is 4 digits = up to 9999 coins). |

### Badge Obedience

|  |  |
| --- | --- |
| *Definition* | *Description* |
| BASE\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have no badges. |
| BADGE\_1\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 1 badge. |
| BADGE\_2\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 2 badges. |
| BADGE\_3\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 3 badges. |
| BADGE\_4\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 4 badges. |
| BADGE\_5\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 5 badges. |
| BADGE\_6\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 6 badges. |
| BADGE\_7\_OBEDIENCE\_LEVEL | The highest level that a traded Pokémon will obey the player at if they have 7 badges. |

### OW Palette Ids

These represent the OW palette Ids of the ninja overworld disguises. If you have changed the default OW palettes, you may want to change these values as well.

|  |  |
| --- | --- |
| *Definition* | *Description* |
| TREE\_DISGUISE\_PAL\_ID | The OW palette Id the ninja tree disguise uses (movement type *0x39* in AdvanceMap). |
| ROCK\_DISGUISE\_PAL\_ID | The OW palette Id the ninja mountain disguise uses. (movement type *0x3A* in AdvanceMap). |
| WEIRD\_DISGUISE\_PAL\_ID | The OW palette Id the ninja weird disguise uses (I have no idea what the movement type is). |

### Pre-Battle Mugshot Options

Different vars for the [Pre-Battle Mugshots](#_Pre-Battle_Mugshots).

|  |  |
| --- | --- |
| *Definition* | *Description* |
| FR\_PRE\_BATTLE\_MUGSHOT\_STYLE | Enable pre-battle mugshots. |
| VAR\_PRE\_BATTLE\_MUGSHOT\_STYLE | Change mugshot tilemap style. |
| VAR\_PRE\_BATTLE\_MUGSHOT\_SPRITE | Can be used to change the player’s mugshot image to a Vs. Symbol. |
| VAR\_MUGSHOT\_PLAYER\_PAL | Change player’s tilemap palette for Two Bars. |
| FLAG\_LOAD\_MUGSHOT\_SPRITE\_FROM\_TABLE | Load Opponent mugshot from table. |

### Safari Zone Options

The following options allow you to set specific safari ball and safari step counter initial and maximum values.

|  |  |
| --- | --- |
| Definition | Description |
| EXPAND\_SAFARI\_BALLS | If defined, the player can hold up to 0xFFFF (65535) Safari Balls! |
| SAFARI\_ZONE\_STEPS\_START | The number of Safari steps the player starts with. |
| SAFARI\_ZONE\_MAX\_STEPS | Maximum number of Safari steps the player can have remaining (see *special 0x89* to add/remove steps). |
| SAFARI\_ZONE\_BALL\_START | The number of Safari balls the player will start the safari zone with. |
| MAX\_SAFARI\_BALLS | The maximum number of Safari balls the player can have (See *special 0x87* to add/remove safari balls up to this amount). |

### Randomizer Options

|  |  |
| --- | --- |
| Definition | Description |
| POKEMON\_RANDOMIZER\_FLAG | Setting this flag causes any Pokemon created to have its species randomized. The Pokemon created can fall anywhere bewteen SPECIES\_BULBASUAR and NUM\_SPECIES. Pokemon found in the \*gRandomizerBanList\* table found in \*\*src/Tables/Pokemon\_Tables.c\*\* will never be created. |
| POKEMON\_LEARNSET\_RANDOMIZER\_FLAG | Setting this flag randomizes the level-up moveset of each Pokemon. |
| ABILITY\_RANDOMIZER\_FLAG | Setting this flag randomizes the possible abilities for each Pokemon. The abilities found in the \*gRandomizerAbilityBanList\* table found in \*\*src/Tables/Pokemon\_Tables.c\*\* will never be chosen. |

### Memory Locations

|  |  |
| --- | --- |
| *Definition* | *Description* |
| EXISTING\_FOSSIL\_IMAGE\_TABLE\_ADDRESS | If FOSSIL\_IMAGE HACK is defined, and you already have a fossil image table inserted somewhere in your hack that you wish to use, uncomment this line and replace the given pointer with your pointer. See [Special 0x18B](#_Other_Specials) for table details. |
| EXISTING\_OW\_TABLE\_ADDRESS | If you have used JPAN’s hacked engine to add new overworld tables in, and you would like to use the table already inserted in your hack to keep track of these tables, uncomment this line and replace the given pointer with your pointer. |

NOTE: Putting a // before any of the below features will remove them from the engine.

### Misc Features

|  |  |
| --- | --- |
| *Definition* | *Description* |
| INVERSE\_BATTLES | Enables the possibility of having an [Inverse Battle](https://bulbapedia.bulbagarden.net/wiki/Inverse_Battle) if the INVERSE\_FLAG is set. |
| TIME\_ENABLED | Various features are updated to account for the time of day. Certain palettes are also dynamically faded depending on the time of day. Removing this feature will always result in *Daytime*. |
| DNS\_IN\_BATTLE | If TIME\_ENABLED is defined, then certain background palettes will be faded dynamically in battle depending on the time of day. These values can be set in **include/new/dns\_data.h**. |
| OVERWRITE\_RIVAL | Loads the buffered rival’s name for trainer classes 0x51, 0x59, and 0x5A. |
| TRAINER\_CLASS\_POKE\_BALLS | Creates trainer Pokémon with custom Poke Balls based on trainer class determined by what is set in **src/Tables/** **Class\_Based\_Poke\_Ball\_Table.c**. |
| TRAINERS\_WITH\_EVS | Trainer Pokémon with a custom moveset, custom item, an AI value greater than 1, and an IV value (labeled EVs in most trainer editors) greater than 0 will have custom stats loaded from the spread number indicated by the IV value in  **src/Tables/trainers\_with\_evs\_table.h**.  For example, setting the IV value to 1 will loaded the spread labeled “*1*” in *gTrainersWithEvsSpreads*.  Modifying the required conditions to activate this feature can be done by searching for the line *#ifdef TRAINERS\_WITH\_EVS* in **src/build\_Pokémon.c**. |
| CONTINUE\_LOST\_BATTLES | If TUTORIAL\_BATTLES is defined, then if Var 0x8000 is set to 0xFEFE before a battle begins, *trainerbattle 0x9* can be used to continue a lost battle. The result of the battle will then be stored in Var LAST\_RESULT (LastResult).  If TUTORIAL\_BATTLES is not defined, then *trainerbattle 0x9* will always allow a lost battle to be continued. |
| REALLY\_SMART\_AI | The AI knows everything about the Pokémon on the player’s side (such as which moves it has, its ability, item, etc.) without having seen it first. This does not allow the AI to know what move the player will use before they use it. |
| DISPLAY\_REAL\_MOVE\_TYPE\_ON\_MENU | When choosing a move or viewing a Pokémon’s moves on the summary screen, the move type the move will become when used will be displayed (such as Hidden Power, Weather Ball in Weather, etc.). |
| DISPLAY\_REAL\_ACCURACY\_ON\_MENU | When pressing the *L-button* while choosing a move, the "true" move accuracy will be displayed. For example, the move *Psychic* used by a Pokémon with *Compound Eyes* will have its accuracy appear as 130. |
| DISPLAY\_REAL\_POWER\_ON\_MENU | When pressing the *L-button* while choosing a move, the "true" move power will be displayed. For example, moves *like Fury Cutter* and *Return* will show their correct power. |
| DISPLAY\_EFFECTIVENESS\_ON\_MENU | Move types will be highlighted in the move menu based on their type effectiveness against opponents. The highlighting scheme is as follows:   * Green: Super Effective * Yellow: Not Very Effective * Red: No Effect * **Bold: STAB** (applies over above except for “No Effect”) |
| CUSTOM\_BATTLE\_BACKGROUNDS | Setting BATTLE\_BG\_VAR will allow custom backgrounds to be loaded in battles. |
| OVERWRITE\_BG\_FOR\_LEADER\_CHAMPION | Special Battle Background palettes will be loaded in for battles against Gym Leaders and the Champion, using the regular indoor background graphics as a base. |
| BRIDGE\_FIX | The water battle background will only be loaded in battle if the player's surfing sprite is shown. This means that if the player is walking on water, the battle background loaded will be incorrect. |
| MEGA\_EVOLUTION\_FEATURE | [Mega Evolutions](https://bulbapedia.bulbagarden.net/wiki/Mega_Evolution) can be used. |
| TUTORIAL\_BATTLES | Professor Oak’s tutorial will be activated for *trainerbattle 0x9*. |
| TANOBY\_RUINS\_ENABLED | Causes [Unown](https://bulbapedia.bulbagarden.net/wiki/Unown_(Pok%C3%A9mon)) to be spawned in maps using the [Tanoby Ruins](https://bulbapedia.bulbagarden.net/wiki/Tanoby_Chambers) map names according to the current chamber. Error prevention has been added to also allow random Unown to be generated outside of the Tanoby Ruins maps. |
| ALTERING\_CAVE\_ENABLED | If the current map is the [Altering Cave](https://bulbapedia.bulbagarden.net/wiki/Altering_Cave) and Var 0x4024 is set, Wild Pokémon will spawn based on the contents of the var. |
| SWEET\_SCENT\_ONLY\_IN\_CLEAR\_WEATHER | In certain generations, [Sweet Scent](https://bulbapedia.bulbagarden.net/wiki/Sweet_Scent_(move)) only spawns wild Pokémon in the Overworld if the weather is clear. |
| OBEDIENCE\_BY\_BADGE\_AMOUNT | Pokémon obedience is determined by the number of badges the Player has rather than by which badges the player has. The other badge defines in this case act as “number of badges acquired” instead of “acquired badge X”. |
| SAVE\_BLOCK\_EXPANSION | Expands the amount of memory that is saved when the player saves the game. This feature breaks compatibility with the FR *Mystery Gift* and *Trainer Tower* features. Uncommenting this line also requires removal of all related hooks. Search for *Save Expansion Hooks* in **hooks**. WARNING: Removing this also removes added vars, PC box space, roaming Pokémon, and many other features. It is highly recommended to keep it. |
| SELECT\_FROM\_PC | If uncommented, allows the player to select and manipulate data of Pokémon from the PC storage boxes. See [PC Selection](#_Select_from_PC) for more details. |
| SET\_HEALING\_PLACE\_HACK | If uncommented, the whiteout hack from JPAN’s FR engine is implemented, allowing *VAR\_HEALINGMAP*, *VAR\_HEALING\_XPOS*, and *VAR\_HEALING\_YPOS* to be utilized to overwrite the default respawn point. |
| FOSSIL\_IMAGE\_HACK | Grants the ability to load custom images from a table using [Special 0x18B](#_Other_Specials). If *EXISTING\_FOSSIL\_IMAGE\_TABLE\_ADDRESS*  is commented out, then the table of images can be found by searching for *gFossilImageTable* in **src/script\_specials.c**. Otherwise the table is loaded from  *EXISTING\_FOSSIL\_IMAGE\_TABLE\_ADDRESS*. |
| EVO\_HOLD\_ITEM\_REMOVAL | Evolving a Pokémon by having it hold an item upon level up or trading removes the item after evolution (like normal). Commenting this out means Pokémon will retain their items after evolution. |
| EXPAND\_MOVESETS | Adds level up moves for each Pokémon which can be found in **src/Tables/Learnsets.c**. This file also includes learnests for expanded Pokémon which are commented out by defeault.  Comment this line if you would rather use the learnsets created in the *Dynamic Pokémon Expansion*. Commenting this line out without properly having expanded the level up moves in some way will cause Pokémon to learn garbage moves. |
| FATHER\_PASSES\_TMS | During breeding, any TMs the father knows will be passed down to the baby if it can learn that TM. This feature was removed from main series Pokémon games in Gen 6. |
| INHERIT\_MASTER\_CHERISH\_BALL | If defined, an offspring can be hatched into a parent’s *Master* or *Cherish Ball* (unlike in the actual games). |
| GIVEPOKEMON\_BALL\_HACK | The last byte of the *givepokemon* scripting command allows you to pass in a Poké Ball type to assign to the Pokémon. Ball types can be found in **include/new/catching.h**. |
| FRLG\_ROAMING | When a [roaming](https://bulbapedia.bulbagarden.net/wiki/Roaming_Pok%C3%A9mon) Pokémon is created, it will either be a *Entei*, *Raikou*, or *Suicune*, depending on the player's starter choice. No other roamer can be created other than these 3. |
| CAN\_RUN\_IN\_BUILDINGS | Allows the player to run indoors (you’re a jerk if you don’t let them). |
| NO\_POISON\_IN\_OW | Pokémon will not take damage from [*Poison* in the overworld](https://bulbapedia.bulbagarden.net/wiki/Poison_(status_condition)#Outside_of_battle) as the player walks. |
| POISON\_1\_HP\_SURVIVAL | Instead of allowing Pokémon to faint from Poison in the overworld, Pokémon will survive the poison with 1 HP like in Gen 4. If NO\_POISON\_IN\_OW is defined then this line is useless. |
| BW\_REPEL\_SYSTEM | When the player’s repel wears off, they will be prompted with a textbox to use another one if they have in their bag. |
| AUTO\_NAMING\_SCREEN\_SWAP | After the player types the first character as an uppercase character in any naming screen, the text with automatically flip to lowercase letters. |
| MULTIPLE\_PREMIER\_BALLS\_AT\_ONCE | When the Player buys more than 10 Poké Balls at once, they will receive Premier balls equal to the number of Poké Balls they bought divided by 10 (rounded down), as opposed to just a single [Premier Ball](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9_Ball#Premier_Ball) regardless of how many Poké Balls were purchased. |
| NON\_TRAINER\_SPOTTING | This feature allows you to assign scripts to regular NPCs and set their trainer byte, to cause them to walk up to the player as if they were a trainer. Additionally, by setting the NPC’s *Unknown* halfword in AdvanceMap (right under *Person Id*) to a flag, the game will check (and set) that flag before the NPC walks up to the player to prevent the script from running in an infinite loop (placing a flag here is not needed if you’ll make the NPC disappear after their first encounter). |
| BIKE\_ON\_ANY\_NON\_INSIDE\_MAP | Normally, the ability to bike on the map is determined by a byte in the map header. This feature makes it so the bike can be used on all outdoor maps. Note that biking in areas such as *Route Gates* will no longer work. |
| GEN\_4\_PLAYER\_RUNNING\_FIX | Increases the lag between frames as the player OW runs, to simulate a more accurate Gen 4 running effect. |
| EXPAND\_MOVE\_REMINDER\_DESCRIPTION | Move descriptions will take up 5 lines in the *Move Reminder* screen as opposed to 4. It is recommended to use this feature as many attack descriptions were expanded to take up 5 lines. |
| ITEM\_PICTURE\_ACQUIRE | An item is shown in the lower right corner of the textbox when the player finds or obtains an item used one of the standard scripting functions (*callstd*). Key Items, TMs, & HMs appear enlarged in the centre of the screen instead. |
| EXPANDED\_TEXT\_BUFFERS | Expands the number of text buffers available for string scripting. The following text buffers are now available.   |  |  |  | | --- | --- | --- | | XSE Buffer # | String Arg | Size | | 0x5 | FD 07 (\v\h07) | 32 bytes | | 0x6 | FD 08 (\v\h08) | 32 bytes | | 0x7 | FD 09 (\v\h09) | 32 bytes | | 0x8 | FD 0A (\v\h0A) | 32 bytes | | 0x9 | FD 0B (\v\h0B) | 32 bytes | | 0xA | FD 0C (\v\h0C) | 100 bytes | | 0xB | FD 0D (\v\h0D) | 100 bytes | |
| FOOTSTEP\_NOISES | When the player or any NPC walks through grass or sand, sounds will be played. The sounds that play can be altered by modifying the values in the functions PlayGrassFootstepNoise and  PlaySandFootstepNoise both found in  **src/overworld.c**.[[1]](#footnote-1) |
| CAN\_ONLY\_USE\_OMNIDRECTIONAL\_  JUMP\_ON\_HEIGHT\_2 | If a tile’s behaviour byte in AdvanceMap is set to MB\_OMNIDIRECTIONAL\_JUMP, then the player will only jump over the block when their height is 2 (*0xC* in AdvanceMap, also labeled *Height 2 (0x3)*). This means that the player will not jump over the tile if they’re surfing, for example. |
| HOOPA\_CHANGE\_IN\_PC | Hoopa-Unbound will revert to its confined form when placed in or withdrawn from a PC Box. |
| SHAYMIN\_CHANGE\_IN\_PC | Shaymin-Sky will revert to its confined form when placed in or withdrawn from a PC Box. This feature was removed in Gen 7. |
| HIGH\_PITCH\_MEGA\_PRIMAL\_CRY | Whenever the cry of a Mega Evolved Pokémon is played, it will played at a higher pitch. This is useful if you're using the base form cries for the Mega Evolutions as well. |
| SCROLLING\_MULTICHOICE | Enables [scrolling multichoice menus](#_Scrolling_Multichoice_Windows) by using special 0x158. |
| REPLACE\_SOME\_VANILLA\_SPECIALS | Replaces the following FireRed specials:  Special 0x7C – BufferNickname  Special 0x7D – CheckTradedPokemon  Special 0x9E – NicknamePokemon  Special 0x156 – StartGhostBattle  Special 0x18B – ShowFossilImage  with newer versions. This will likely break compatability with vanilla FR and require modifications to any scripts that use them. |

### Misc Battle Effect Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_BURN\_DAMAGE | [Burn](https://bulbapedia.bulbagarden.net/wiki/Burn_(status_condition)) damage does 1/8 of max health instead of 1/16. |
| OLD\_PARALYSIS\_SPD\_DROP | [Paralysis](https://bulbapedia.bulbagarden.net/wiki/Paralysis_(status_condition)) lower Speed down to 1/4 instead of ½. |
| OLD\_CONFUSION\_CHANCE | [Confusion](https://bulbapedia.bulbagarden.net/wiki/Status_condition#Confusion) stops attacks 50% of the time instead of 33%. |
| INFINITE\_WEATHER | [Weather abilities](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_with_effects_on_weather_conditions) make weather last for infinite turns. |
| INFINITE\_TERRAIN | [Terrain abilities](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_with_effects_on_terrain) make terrain last for infinite turns. |
| NO\_SHEER\_COLD\_NERF | Remove all Gen 7 [Sheer Cold](https://bulbapedia.bulbagarden.net/wiki/Sheer_Cold_(move)) nerfs. |
| OLD\_MOVE\_SPLIT | The [Physical/Special](https://bulbapedia.bulbagarden.net/wiki/Damage_category#Physical.2FSpecial_split) split is based on move types.  Status moves are still set with the split byte, however. |

### Ability Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_GALE\_WINGS | [Gale Wings](https://bulbapedia.bulbagarden.net/wiki/Gale_Wings_(Ability)) activates regardless of the user's HP. |
| OLD\_PRANKSTER | [Prankster](https://bulbapedia.bulbagarden.net/wiki/Prankster_(Ability)) won’t fail against Dark-Types. |

### Damage Calculation Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_CRIT\_DAMAGE | [Critical hits](https://bulbapedia.bulbagarden.net/wiki/Critical_hit) to do 2x damage; 3x with [Sniper](https://bulbapedia.bulbagarden.net/wiki/Sniper_(Ability)). |
| CRIT\_CHANCE\_GEN\_6 | Uses the Gen 6 crit chance. |
| CRIT\_CHANCE\_GEN\_2\_TO\_5 | Uses the Gen 2-5 crit chance. |
| BADGE\_BOOSTS | Having badges gives the Player’s Pokémon [stat boosts](https://bulbapedia.bulbagarden.net/wiki/Badge#Stat_boost). |
| OLD\_ATE\_BOOST | “[Ate](https://bulbapedia.bulbagarden.net/wiki/Category:Abilities_that_can_modify_move_types)” abilities give a 1.3x boost instead of 1.2x. |
| OLD\_GEM\_BOOST | [Gems](https://bulbapedia.bulbagarden.net/wiki/Gem) give a 1.5x boost instead of 1.2x. |
| OLD\_EXPLOSION\_BOOST | [Exploding](https://bulbapedia.bulbagarden.net/wiki/Explosion_(move)) moves halve the target’s Defense. |
| OLD\_HIDDEN\_POWER\_BP | [Hidden Power](https://bulbapedia.bulbagarden.net/wiki/Hidden_Power_(move)) has its Base Power calculated from the attacker’s IVs. |
| PORTAL\_POWER | Enables Hoopa-Unbound’s signature ability in Pokémon Unbound, *Portal Power*. This reduces the power of non-contact moves by 25%. |
| OLD\_SOUL\_DEW\_EFFECT | [Soul Dew](https://bulbapedia.bulbagarden.net/wiki/Soul_Dew) doubles Latios & Latias' Sp. Atk & Sp. Def. |
| OLD\_PARENTAL\_BOND\_DAMAGE | The second hit of [Parental Bond](https://bulbapedia.bulbagarden.net/wiki/Parental_Bond_(Ability)#Generation_VI) does 50% of the original damage instead of 25%. |

### Capturing Pokémon Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| NO\_HARDER\_WILD\_DOUBLES | In Gen 5, Pokémon encountered in wild double battles were [harder to catch](https://bulbapedia.bulbagarden.net/wiki/Catch_rate#Capture_method_.28Generation_V.29) (based on how many species are owned). This feature implements that catch rate decrement. |
| CRITICAL\_CAPTURE | Allows for [Critical Capture](https://bulbapedia.bulbagarden.net/wiki/Catch_rate#Critical_capture) to occur. The odds at which this occurs can be found in the function:  *static bool8 CriticalCapture(u32 odds)* found in **src/catching.c**. |

### Exp. Gain Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| OLD\_EXP\_SHARE | The [Exp. Share](https://bulbapedia.bulbagarden.net/wiki/Exp._Share) acts like it did before Gen 6. |
| TRAINER\_EXP\_BOOST | Gives an Exp boost for defeating a trainer’s Pokémon. (Pre Gen 7) |
| OLD\_EXP\_SPLIT | Exp. is split amongst all participating Pokémon. (Pre Gen 6) |
| FLAT\_EXP\_FORMULA | Use a Flat Exp. calculation formula. (Gens 2- 4, 6) |
| GEN\_7\_BASE\_EXP\_YIELD | Base Exp is retrieved from the table *gBaseExpBySpecies* found in the file **src/Tables/Experience\_Tables.c**, instead of being loaded from the Pokémon’s base stats. This is done to account for larger Exp. values that started in Gen 5. The table is pre-set to match [Gen 7 Exp. values](https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_effort_value_yield). |
| CAPTURE\_EXPERIENCE | When a Pokémon is caught, experience will be rewarded as if the caught Pokémon fainted. |

### Other Battle Options

|  |  |
| --- | --- |
| *Definition* | *Description* |
| NO\_GHOST\_BATTLES | Disables the [Ghost battle](https://bulbapedia.bulbagarden.net/wiki/Ghost_(literal)) feature from Pokémon Tower in Lavender town. |
| GEN4\_PLUS\_SELECTION\_SCREEN | This does not give the [Gen 4+ selection screen](https://www.pokecommunity.com/showthread.php?t=414022), it only adds in-battle features that supports it. |
| OBEDIENCE\_CHECK\_FOR\_PLAYER\_ORIGINAL\_POKÉMON | Opens up the possibility that the Player's Pokémon can [disobey](https://bulbapedia.bulbagarden.net/wiki/Obedience) them, as opposed to just traded Pokémon. |
| WILD\_ALWAYS\_SMART | All wild Pokémon use AI features meant for trainers. |
| HAIL\_IN\_BATTLE | Enables the [Hail](https://bulbapedia.bulbagarden.net/wiki/Hail_(weather_condition)) weather effect in battle when the OW weather is set to WEATHER\_STEADY\_SNOW (0x7). |
| FOG\_IN\_BATTLE | Enables the [Fog](https://bulbapedia.bulbagarden.net/wiki/Fog) weather effect in battle. Do not enable this feature without first enabling one of the fog features below! |
| FOG\_IN\_BATTLE\_1 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_1 (0x6). |
| FOG\_IN\_BATTLE\_2 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_2 (0x9). |
| FOG\_IN\_BATTLE\_3 | Enables the Fog weather effect when the OW weather is set to WEATHER\_FOG\_3 (0xA). |
| HIDE\_HEALTHBOXES\_DURING\_ANIMS | Hides the healthboxes (battle bars, etc.) during move animations, and some special animations (like Mega Evolution). This has been done since Gen 4. |
| DONT\_HIDE\_HEALTHBOXES\_ATTACKER\_STATUS\_MOVES | If HIDE\_HEALTHBOXES\_DURING\_ANIMS is defined, when the attacker is using a move that only targets itself, the healthboxes will not be hidden. |
| ENCOUNTER\_MUSIC\_BY\_CLASS | The music played when a trainer spots the player in the overworld is determined by the trainer class, rather than the music Id set in the trainer data. The song options are listed in **src/Tables/Music\_Tables.c** and can be modified by changing the values in  *gClassBasedTrainerEncounterBGM*. Any class not defined in the array will be automatically set to BGM\_EYE\_BOY. |
| OKAY\_WITH\_AI\_SUICIDE | The AI is allowed to use self-destructing moves. |

### DexNav Options

The following options relate to the [DexNav](#_Dex_Nav) feature and can be found in **include/new/dexnav\_config.h**.

|  |  |
| --- | --- |
| *Definition* | *Description* |
| DEXNAV\_TIMEOUT | The number of seconds before the *DexNav* search times out for the player. The maximum is 1092 seconds. |
| SNEAKING\_PROXIMITY | The amount of tiles (both horizontally and vertically) the player is allowed to be before having to sneak to avoid the Pokémon from running away. *Sneaking* is defined by not running or biking. |
| SEARCHLEVEL0\_MOVECHANCE  SEARCHLEVEL5\_MOVECHANCE  SEARCHLEVEL10\_MOVECHANCE  SEARCHLEVEL25\_MOVECHANCE  SEARCHLEVEL50\_MOVECHANCE  SEARCHLEVEL100\_MOVECHANCE | The chance in percent that a Pokémon with any of these search levels will be encountered with an [Egg Move](https://bulbapedia.bulbagarden.net/wiki/Egg_Move) in the first slot. |
| SEARCHLEVEL0\_ABILITYCHANCE  SEARCHLEVEL5\_ABILITYCHANCE  SEARCHLEVEL10\_ABILITYCHANCE  SEARCHLEVEL25\_ABILITYCHANCE  SEARCHLEVEL50\_ABILITYCHANCE  SEARCHLEVEL100\_ABILITYCHANCE | The chance in percent that a Pokémon with any of these search levels will be encountered with its [Hidden Ability](https://bulbapedia.bulbagarden.net/wiki/Ability#Hidden_Abilities) (if it can have one). |
| SEARCHLEVEL0\_ITEM  SEARCHLEVEL5\_ITEM  SEARCHLEVEL10\_ITEM  SEARCHLEVEL25\_ITEM  SEARCHLEVEL50\_ITEM  SEARCHLEVEL100\_ITEM | The chance in percent that a Pokémon with any of these search levels will be encountered with a held item (if it can have one). |
| SEARCHLEVEL0\_ONESTAR  SEARCHLEVEL5\_ONESTAR  SEARCHLEVEL10\_ONESTAR  SEARCHLEVEL25\_ONESTAR  SEARCHLEVEL50\_ONESTAR  SEARCHLEVEL100\_ONESTAR  SEARCHLEVEL0\_TWOSTAR  SEARCHLEVEL5\_TWOSTAR  SEARCHLEVEL10\_TWOSTAR  SEARCHLEVEL25\_TWOSTAR  SEARCHLEVEL50\_TWOSTAR  SEARCHLEVEL100\_TWOSTAR  SEARCHLEVEL0\_THREESTAR  SEARCHLEVEL5\_THREESTAR  SEARCHLEVEL10\_THREESTAR  SEARCHLEVEL25\_THREESTAR  SEARCHLEVEL50\_THREESTAR  SEARCHLEVEL100\_THREESTAR | The chance in percent that a Pokémon with any of these search levels will be encountered with X number of perfect IVs, where X is equal to the number of stars. |

## Engine Setup

### Mega Evolution / Primal Reversion / Ultra Burst

Before setting up Mega Evolution, two things must be done. First, make sure EVOS\_PER\_MON in the config file is set to the correct number (it should be the same as the number + 1 at 0x43116 in your rom). Second, if you’re not using the *Dynamic Pokémon Expansion*, your Pokémon Editor of choice will need to be modified:

**G3T**:

In your Gen3Tools folder, open up **Customisation/Pokémon Editor.ini**, and add the line **FE=Mega Evolution** to the bottom of the file.



**D&D**:

Has Mega Evolution pre-installed, however it cannot set up Wish-based Mega Evolution correctly.

**G3HS**:

1. Open up the file **PokeRoms.ini**, find your rom code.
2. Modify “evolutionmethods” such that the 254th evolution method is set to *Mega Evolution*:



1. Modify “evomethodsproperties” such that the 254th method is set to *Item*.

[Mega Evolution](https://bulbapedia.bulbagarden.net/wiki/Mega_Evolution)

Mega Evolution set up is similar to the how the [previous](https://www.pokecommunity.com/showthread.php?t=348182) Mega Evolution system by Touched was set up. If a Pokémon is able to Mega Evolve, Mega Evolution can be trigged by pressing start on the move menu once the mega trigger appears.

**NOT Using Dynamic Pokémon Expansion**:

1. Create a Mega Stone in G3T with the following format:



1. Create a Mega Ring key item. Its format is the same as any other key item.
2. Update the definition for the Mega Ring item in **include/constants/items.h**.
3. Open the file **src/mega.c** and search for *KeystoneTable*. Add your mega ring item to this table:



\*\*NOTE: The item does not need to be called *Mega Ring*. Any item added to this table can act as a mega ring.

1. If DEBUG\_MEGA is defined, then Mega Evolution can be used from the start of the game without the requirement of having the Mega Ring in the bag. Otherwise, the player must have a Mega Ring item in the bag to use Mega Evolution (Mega Evolution can always be used in Link Battles or the Battle Tower). If you want a trainer to be able to use Mega Evolution, add a mega ring item to their battle items:



Don’t forget to give the trainer’s Pokémon the relevant Mega Stone to hold!

1. Find the Pokémon in G3T that you wish to bestow the ability of Mega Evolution upon. Set up the evolution data with following template:



In the above image:

* “Nothing required” is set to the mega stone item index created earlier.
* “Evolve to” is set to a new species representing the mega form (you’ll need to expand Pokémon or replace an existing one). Make sure this mega form has the same gender rate and Exp rate as the non-mega form or things will get messed up.

*Wish Mega Evolution* (for Rayquaza) should be set up using the following template:

In the above image:

* “Nothing required” should be set to the move id for *Dragon Ascent* (0x22F / 559). G3T has issues with setting numbers past 0xFF, so you’ll need to hex edit or use a different Pokémon editor to set the proper value for *Dragon Ascent*.
* “Evolve to” is set to a new species representing Mega Rayquaza (you’ll need to expand Pokémon or replace an existing one). Make sure Mega Rayquaza has the same gender rate and Exp rate as Rayquaza or things will get messed up.
* The first of “Unknown Bytes” should be set to 2.

1. Set the Mega form’s evolution data like the following template:



Notice that the item is left blank. Also notice that *Mega Rayquaza* still has its first unknown byte set to 2. Do not forget this!

1. Now just give the player a mega ring item and a Pokémon with the appropriate mega stone and they will be able to Mega Evolve!

**Using Dynamic Pokémon Expansion**:

Add evolution methods in **src/Evolution\_Table.c** with the following format:

Regular Mega Evolution:





Wish Mega Evolution:





Notes about Mega Evolution:

* Only a single Mega Evolution may be used by each side during any given battle.
* In multi battles, both trainers can Mega Evolve a single Pokémon, effectively allowing for two Mega Evolutions per side during any given battle.
* Mega Evolution is not prevented if any Pokémon on the side Ultra Bursted or underwent Primal Reversion.
* If Rayquaza uses a Z-Move, it’ll be unable to Mega Evolve.

[Primal Reversion](https://bulbapedia.bulbagarden.net/wiki/Primal_Reversion)

Primal Reversion does not rely on a mega ring to activate and will thus always activate if Kyogre or Groudon hold the appropriate item. Much less setup is required for Primal Reversion.

**NOT Using Dynamic Pokémon Expansion**:

1. Find the Red Orb and Blue Orb in G3T’s item editor and modify them to match the following:



The item effect should be set as 93 and the item quality should be set as 0 for the Red orb and 1 for the Blue Orb. These numbers determine whether the Primal Reversion is *Red* (0) or *Blue* (1).

1. Modify Kyogre and Groudon’s evolution data in G3T to match the following:



In the above images:

* “Nothing required” is set to the *Blue Orb* item index for Kyogre and to the *Red Orb* item index for Groudon. Note again that G3T does not represent these item indices correctly so you may need to use another editor.
* “Evolve to” is set to a new species representing the primal form (you’ll need to expand Pokémon or replace an existing one). Make sure this primal form has the same gender rate and Exp rate as the non-primal form or things will get messed up.
* The first of the “Unknown Bytes” is set to 1 to represent Primal Reversion.

1. Set up the primal forms’ evolution data in G3T to match the following:



Notice that the items are left blank. Also notice that both Primal Pokémon still have their first unknown bytes set to 1. Do not forget this!

Note about Primal Reversion:

* The alpha and omega symbols on the health bar are generated based on which species is in its Primal form. By default, the alpha symbol is set to appear if the species is Primal Kyogre, and the omega symbol is set to appear if the species is Primal Groudon. To change this requirement, modify the following functions in **src/mega.c**:



These can easily be modified by adding species to compare to. So if I wanted the omega symbol to appear for Primal Dialga as well, I would make the following modification:



Don’t forget to also define PKMN\_PRIMAL\_DIALGA in **include/constants/species.h**!

**Using Dynamic Pokémon Expansion**:

Add evolution methods in **src/Evolution\_Table.c** with the following format:





[Ultra Burst](https://bulbapedia.bulbagarden.net/wiki/Ultra_Burst)

Primal Reversion does not rely on a mega ring to activate and will thus always activate if Necrozma holds the appropriate item. Much less setup is required for Ultra Burst.

**NOT Using Dynamic Pokémon Expansion:**

1. Create an *Ultranecrozium Z* item in G3T with the following format:



1. Modify **both** Necrozma forms’ evolution data in G3T to match the following:



In the above images:

* “Nothing required” is set to the *Ultranecrozium Z* item index. Note again that G3T does not represent these item indices correctly so you may need to use another editor.
* “Evolve to” is set to a new species representing *Ultra Necrozma* (you’ll need to expand Pokémon or replace an existing one). Make sure all *Necrozma* forms have the same gender rate and Exp rate or things will get messed up.
* The first of the “Unknown Bytes” is set to 3 to represent Ultra Burst.

1. Set up the *Ultra Necrozma’s* evolution data in G3T to match the following:



In the above image:

* The items are left blank.
* The first unknown byte is set to 3. Do not forget this!

Note about Ultra Burst:

* Contrary to what is shown in the above image, *Ultra Necrozma* does **not** need reversion data from both *Necrozma* fusion forms (it needs for at least one of them). *Ultra Necrozma* will always revert to the form it *Ultra Bursted* from at the end of the battle, regardless of which species is written in its evolution data. If the *Ultra Necrozma* was encountered in the wild, it will revert to the first species in its evolution list by default.
* *Ultra Burst* is trigged the same way as *Mega Evolution* on the move menu.
* As *Ultra Burst* is not considered *Mega Evolution*, *Ultra Necrozma* can still use Z-Moves if it knows the appropriate base move (*Photon Geyser* by default).

**Using Dynamic Pokémon Expansion**:

Add evolution methods in **src/Evolution\_Table.c** with the following format:



### Z-Moves

[Z-Moves](https://bulbapedia.bulbagarden.net/wiki/Z-Move) work akin to how they work in real Pokémon games.

If a Pokémon holds a *Z-Crystal* corresponding to a specific type, any move of that type can be turned into a Z-Move by pressing the *Start*-button on the move menu, and then the *A*-button to confirm the selection. If a move cannot be turned into a Z-Move, the *Start­*-button will do nothing.

If a certain Pokémon holds its signature *Z-Crystal*, then its signature move can be turned into its signature *Z-Move* (also with the *Start*-button). A list of these signature Z-Moves can be found under *gSpecialZMoveTable* in **src/set\_z\_effect.c**.

The only setup required for Z-Moves involves the creation of Z-Crystals. Each Z-Crystal should be created in G3T with the following format:



The item effect should be set as 130 and the item quality should be set to the move type the Z-Crystal works for (in the above image it is set to 2 [TYPE\_FLYING]). The *Mystery 1* byte can also be set to 1 to remove the item quantity for Z-Crystals (acts as if the player only has the single, unique Z-Crystal).

Special Z-Crystals should be set up similarly, the key difference being that item quality should be set to 255.



Once the Z-Crystals are created, have a Pokémon hold one, give it the appropriate move, and then watch the magic happen!

### Trainer Sliding Messages[[2]](#footnote-2)

In generations after Gen 3, Trainers could interrupt the battle with a message. This engine supports three kinds of those messages:

1. After the opponent’s first Pokémon faints.
2. After the opponent’s last Pokémon is sent in.
3. When the opponent’s last Pokémon is low on health.

Trainers can have any combination (or none at all) of the above messages. To set these up, do the following:

1. Navigate to **src/trainer\_sliding.c** and search for *sTrainerSlides.* This is the table used to define the sliding messages.
2. Add an entry with the following format:



Where 0x59 is the trainer Id (Youngster Ben in this case), *sText\_BenFirstMonDown* is the message displayed when the opponent’s first Pokémon has fainted, *sText\_BenLastSwitchIn* is the message displayed when the opponent switches in their last Pokémon, and *sText\_BenLastLowHP* is the message displayed when the opponent’s last Pokémon is on low health. If you would not like the trainer to say anything at one of these stages, simply replace the entry with *NULL*:



In this case, the trainer will say something after the first Pokémon is defeated and when their last Pokémon is low on HP, but not after they send in their last Pokémon.

1. At the top of the file (under the #include’s), add declarations for the strings you’ve defined. For example, for the first message struct declared above, the top of the file should look like this:



Each string title is preceded by "extern u8 " and ends with "[];".

1. Open the file **strings/trainer\_sliding\_strings.string** and add entries for the strings you’ve defined. For example, for the first message struct declared above:



The format for the strings follows similarly to *XSE*, the key difference being that each line does **not** start with "= ". An equal’s sign at the beginning of the line will be treated as such and be seen in the game. Also note that each line ends with "\p" in order to wait for the player’s key press.

Follows these steps and continuously add new entries to the table to add flavour to battles!



### Multi Battles

There are 4 different types of [Multi Battles](https://bulbapedia.bulbagarden.net/wiki/Multi_Battle) supported by this engine:

1. Player Vs. Two Trainer Opponents.
   * Can be set up in a script or by being spotted by two different trainers.
2. Player & Partner Vs. Two Trainer Opponents.
3. Player & Partner Vs. One Trainer Opponent.
4. Player & Partner Vs. Two Wild Pokémon.

There are two different ways of setting these up:

**XSE Friendly Method**

This method allows you to set up multi battles through scripted events.

* To set up a battle against two opponents:
  + Set the flag TWO\_OPPONENT\_FLAG.
  + Set the var SECOND\_OPPONENT\_VAR to the trainer id of the second trainer.
  + Use the *loadpointer* scripting command in conjunction with special **0xXX** to load the second trainer’s defeat text.
  + Then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle.
* To set up a battle with a partner:
  + Set the flag TAG\_BATTLE\_FLAG.
  + Set the var PARTNER\_VAR to the trainer id of the partner.
  + Set the var PARTNER\_BACKSPRITE\_VAR to the [backsprite Id](#_Trainer_Backsprites) of the partner.
  + Then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle.

**This will initiate a battle with a partner against a single opponent.**

* To battle with a partner and two opponents, set all flags and vars listed in the previous two steps, and then use the scripting command *trainerbattle 0x3* (or *0x9*) to start the battle. A sample script to do this looks as follows:

#define TWO\_OPPONENT\_FLAG 0x909

#define TAG\_BATTLE\_FLAG 0x908

#define SECOND\_OPPONENT\_VAR 0x5010

#define PARTNER\_VAR 0x5011

#define PARTNER\_BACKSPRITE\_VAR 0x5012

#define SPECIAL\_LOAD\_SECOND\_DEFEAT\_TEXT 0xAC

#org @start

setflag TWO\_OPPONENT\_FLAG 'Setup battle against two opponents

setflag TAG\_BATTLE\_FLAG 'Setup battle with partner

setvar SECOND\_OPPONENT\_VAR 0x59 'Youngster Ben

setvar PARTNER\_VAR 0x5B 'Team up with Youngster Josh

setvar PARTNER\_BACKSPRITE\_VAR 0x2 'Brendan’s Backsprite

loadpointer 0x0 @SecondTrainerDefeatText

special SPECIAL\_LOAD\_SECOND\_DEFEAT\_TEXT

trainerbattle 0x3 0x5A 0x0 @FirstTrainerDefeatText 'Youngster Calvin

end

**Non-XSE Friendly Method**

This method for setting up multi battles is not possible to code in XSE, but it is significantly easier to code and allows more versatility with random trainer scripts. It is recommended to use the Thumb assembler in conjunction the xse defines provided in this engine to compile these custom scripts (which can then be called after inserting the hex data).

* To set up a battle against two opponents, use the following scripting command:

trainerbattle 0xB **FOE\_1\_ID** *FOE\_2\_ID* **FOE\_1\_NPC\_ID** *FOE\_2\_NPC\_ID* 0x0 **INTRO\_TEXT\_A** *INTRO\_TEXT\_B* **DEFEAT\_TEXT\_A** *DEFEAT\_TEXT\_B* **CANNOT\_BATTLE\_TEXT**

Where:

* + FOE\_1\_ID: The trainer Id of the first opponent.
  + FOE\_2\_ID: The trainer Id of the second opponent.
  + FOE\_1\_NPC\_ID: The local Id (person Id in AdvanceMap) of the first opponent.
  + FOE\_2\_NPC\_ID: The local Id (person Id in AdvanceMap) of the second opponent.
  + INTRO\_TEXT\_A: The intro battle text said by the first opponent.
  + INTRO\_TEXT\_B: The intro battle text said by the second opponent.
  + DEFEAT\_TEXT\_A: The defeat text said by the first opponent.
  + DEFEAT\_TEXT\_B: The defeat text said by the second opponent.
  + CANNOT\_BATTLE\_TEXT: The text said by either opponent when the player doesn’t have enough viable Pokémon to fight with.

This trainerbattle 0xB command is special such that you can assign it to random NPCs to effectively make better random double battles than with a *Twins* class, for example. When using this on two random NPCs, make sure they stand next to each other! Otherwise it’ll look off when they walk towards the player together.

* + To set up a battle with a partner, use the following scripting command:

trainerbattle 0xC **FOE\_ID** *PARTNER\_ID* **PARTNER\_BACKSPRITE\_ID** 0x0 *DEFEAT\_TEXT*

Where:

* FOE\_ID: The trainer Id of the opponent.
* PARTNER\_ID: The trainer Id of the player’s partner.
* PARTNER\_BACKSPRITE\_ID: The backsprite Id of the player’s partner.
* DEFEAT\_TEXT: The text said when the opponent loses the battle.
* To battle with a partner and two opponents, use the following scripting command:

trainerbattle 0xA **FOE\_1\_ID** *FOE\_2\_ID* **PARTNER\_ID** *PARTNER\_BACKSPRITE\_ID* 0x0 **DEFEAT\_TEXT\_A** *DEFEAT\_TEXT\_B*

Where each of the title definitions is the same as listed *trainerbattle 0xB* and *trainerbattle 0xC*. Note that both *trainerbattle 0xB* and *trainerbattle 0xC* cannot be used on random NPCs. They must be used from within and event script.

And with that, you can set up amazing multi battles!

### Wild Double Battles

Encountering two wild Pokémon at once was introduced in Gen 4 with it occurring when the player was teamed up with another player. Then, in Gen 5, it became possible to encounter two Pokémon at once in special grass. This engine supports both of those features.

**Wild Double Battles With Partner**

If there is ever a situation where you’d like all wild battles in a given area to be with a partner, add the following as an *On entering map/on menu close [5]* level script in AdvanceMap:

#define DOUBLE\_WILD\_BATTLE\_FLAG 0x9F9

#org @start

setflag DOUBLE\_WILD\_BATTLE\_FLAG

setflag TAG\_BATTLE\_FLAG 'Setup battle with partner (same as above)

setvar PARTNER\_BACKSPRITE\_VAR 0x2 'Brendan’s Backsprite (same as above)

end

This will cause all battles against trainers on the map to be fought with a partner, and all wild battles fought on the map to be against two Pokémon. Conditions can also be added to the script (such as a checkflag to only execute the script if some flag is set). Don’t forget to clear these flags once the player no longer needs a partner!

**Wild Double Grass**

Special grass tiles that can initiate wild battles can be created as well. Grass with a background byte of 0x5 will have a chance of starting a wild double battle, and grass with a background byte of 0x25 will have a chance of starting a wild double battle and be covered by the player. Wild double water tiles are similarly done using 0x6 and 0x26. The chance a wild double battle will be started when walking in this grass is determined by the value set (in percent) in WILD\_DOUBLE\_RANDOM\_CHANCE.

**Scripted Wild Double Battles**

Here is a sample script:

wildbattle 0xFFFF 0x0 0x0 'Double indicator

wildbattle PKMN\_CLEFAIRY 20 ITEM\_NONE

wildbattle PKMN\_PIKACHU 20 ITEM\_NONE

special 0x138 'Starts the battle

waitstate

Sample double wild grass in AdvanceMap:



I’m not entirely sure if this script will compile properly in XSE, but if you use it in PKSV by replacing *wildbattle* with *battle* and *waitstate* with *waitspecial* the battle will begin properly.

### Trainer Backsprites

Adding a backsprite into the game has never been easier.

1. Go to **graphics/Backsprites** and add a minimum of **4** backsprites to a single file for your new trainer. Make sure they’re each a size of 64 x 64 and indexed to 16 colours! Sample *Brendan* and *May* backsprites come bundled with the engine.



1. Open **include/constants/trainers.h** and define a new constant for your new backsprite. I will call the tutorial one TRAINER\_BACK\_PIC\_BRENDAN\_NEW:



1. Open **src/Tables/Backsprite\_Tables.c** to allow the game to read your new backsprites.
2. Search for *gTrainerBackPicPaletteTable* and right above it add two lines with format:

**extern const u8 [FILE\_NAME\_NO\_EXTENSION]Pal[];**

**#define gTrainerPalette\_[NAME] gTrainerBackPic\_[FILE\_NAME\_NO\_EXTENSION]Pal**

So our entry for the new Brendan would look like:



1. Now add this new palette underneath the Old Man’s entry in the table *gTrainerBackPicPaletteTable*:



1. Add a line for the animation of the backsprite. The pointer should be *0x8239F44* for 5 frames, and *0x8239F54* for four frames. So since our backsprite has four frames:



1. Search for *gTrainerBackAnimsPtrTable* and add your new entry after the old man.



1. Add an entry in the gTrainerBackPicCoords table after the old man:



1. Next, add a declaration and a table for the image frames like so:





Format for each name is [FILE\_NAME\_NO\_EXTENSION]Tiles.

1. Finally, search for gSpriteTemplateTable\_TrainerBackSprites and add an entry for the new backsprite after the old man:



The entry in *.images* should be the same as the table added in step 9.

And you’re done!

### Battle Terrain

If you have inserted any new battle backgrounds using the tutorial [here](https://www.pokecommunity.com/showthread.php?t=302401), then certain modifications will need to be made to make the engine compatible with those backgrounds.

1. Open up the file **include/battle.h**. Search for BATTLE\_TERRAIN\_CHAMPION in thefile, and add a new definition there. So, for instance, if your new background was a snow field:



1. Open up the file **src/Tables/Terrain\_Table.c**. Search for BATTLE\_TERRAIN\_CHAMPION in thefile, and add a new entry to the table. So continuing on with the snow example:



Notice the entry name is the same as the definition from earlier **+ 4**.

The table has entries for:

* camouflageType: The type the move [Camouflage](https://bulbapedia.bulbagarden.net/wiki/Camouflage_(move)) transforms into.
* secretPowerEffect: The secondary effect of the move [Secret Power](https://bulbapedia.bulbagarden.net/wiki/Secret_Power_(move)).
* secretPowerAnim: The animation of the move *Secret Power*.
* naturePowerMove: The attack the move [Nature Power](https://bulbapedia.bulbagarden.net/wiki/Nature_Power_(move)) becomes.
* burmyForm: The form [Burmy](https://bulbapedia.bulbagarden.net/wiki/Burmy_(Pok%C3%A9mon)) transforms into after this battle. If you would not like *Burmy* to change form after this battle, leave it as SPECIES\_NONE (as shown above).

### Battle Music

There are three different music tables that can be set up. Each table can be found in **src/Tables/Music\_Tables.c**. Any custom song definitions used in these tables should be added to **include/constants/songs.h**. Trainer classes can be found and added into the file **include/constants/trainer\_classes.h**.

**Class Based Encounter Music**

If ENCOUNTER\_MUSIC\_BY\_CLASS is defined, then the table, *gClassBasedTrainerEncounterBGM*, can be modified to determine which music plays in the background when each trainer class is encountered in the overworld. For example, this table causes all *Youngsters* to have the same encounter music, without having to the set the byte for each of them in their trainer data. If you wanted to change which encounter music Youngsters have, all you have to do is make the following change:

→ 

Now, all *Youngsters* will play the girl encounter when they spot the player.

If you do not want to use this feature, the switch statement in *SetUpTrainerEncounterMusic* found in **src/overworld.c** will need to be modified to add new encounter song Ids in.

**Class Based Battle Music**

The table, *gClassBasedBattleBGM*, can be modified to determine the song that plays during trainer battles against certain classes. For example, if I wanted to make all *Team Rocket* battles play the *Gym Leader* theme, I would add the following onto the end of the table:



In a *Multi Battle*, if either trainer has custom battle music, their theme will play. If both trainers have custom battle music, then the theme for the trainer on the right (first opponent) will play.

**Wild Species Based Battle Music**

The table, *gClassBasedTrainerEncounterBGM*, can be modified to determine the song that plays during wild battles against certain species of Pokémon. For example, if I wanted to make all *Rattata* battles play the *Deoxys* theme, I would add the following onto the end of the table:



**Victory Music**

In the file **src/end\_battle.c** is a function called *HandleEndTurn\_BattleWon*. In this function is a switch statement (look for *VICTORY\_MUSIC\_SELECTION*) which controls the songs that play when defeating certain trainer classes. See the example from Pokémon Unbound (include in the file) for how to append new classes to this switch statement.

### Poke Balls

Several new Poké Balls have been added to the engine, in addition to the Trainer Class Based Poké Ball hack.

**Adding Support for Added Balls**

Although catching data has been added in for the new balls, item data has not. This means that if you want to give the player a certain ball, you’ll need to add in item data for it. Adding a new Poké Ball follows the following format in G3T:



Regarding the *Type* field, this related to the *Ball Type* - 1 of the given ball. So looking in **include/new/catching.h**, we see that the *Dusk Ball’s*type is 13, so subtracting 1 is 12.

**Class Based Poke Balls**

Loosely based on the hack created by [Sagiri](https://github.com/Sagiri/cbpb), if TRAINER\_CLASS\_POKE\_BALLS is defined, this implements the feature from Gen 7 where certain Trainer classes always send out Pokémon in a specific type of Poké Ball.

To modify the trainer class balls, open **src/Tables/Class\_Based\_Poke\_Ball\_Table.c**. All the trainer classes have been preloaded into the table, but if you would like to change a trainer class name to your own custom name, do so in **include/constants/trainer\_classes.h** and then update the table accordingly. The ball type defines that can be used can be found in **include/new/catching.h**. So, for example, if you wanted to change the *Youngster*’s ball to a *Great Ball*, you’d make the following change:

→ 

And then the next time the player battles a *Youngster*, all their Pokémon would be sent out in *Great Balls*!

### Trainers With EVs

Loosely based on [DoesntKnowHowToPlay’s hack](https://www.pokecommunity.com/showthread.php?t=307117), if TRAINERS\_WITH\_EVS is defined, this feature allows trainer Pokémon to have certain EV spreads. Preloaded spreads can be found in **src/Tables/trainers\_with\_evs\_table.h**. New spreads should also be added in here as well.

Note that since this is a **.h** file, if you want your changes to compile, you will need to make a change to the file **src/build\_pokemon.c** so the compiler recompiles the Trainer’s With EVs file. The change can be as simple as adding a whitespace character and then removing it. **Don’t use the undo button to remove the change!** The change needs to be removed manually by using the backspace character so the editor saves the file with an updated timestamp.

**Assigning a Spread**

In order for a Pokémon to use a spread, the following must be done:

* The Pokémon must have a custom moveset.
* The Pokémon must have a custom item (this item can still be set to 0).
* The Pokémon must have an IV (labeled "EVs" in most Trainer editors) value greater than 1 (the 0th spread is left empty for this reason).

For example, looking in the file, you can see that spread 5 is predefined as a "Bulky Physical Attacker". If I wanted to assign this to a Pokémon, the layout would look like this (in [Hopeless Trainer Editor](https://www.pokecommunity.com/showthread.php?t=352109)):



As can be seen in the image, both *Custom Movesets* and *Custom Hold Items* have been selected, and the *EVs* value has been set to 5. Note that the hold item has been set to "*????????"*, meaning that *Emboar* is not actually holding an item (this is still fine).

**Creating a New Spread**

If you want to create a new spread, add it to the end of the table and start counting at 31. The table can at most hold up to the 255th spread. Spreads have the following members:

* .nature: The nature of the Pokémon. Natures can be found in **include/pokemon.h**.
* .ivs: The IVs for the Pokémon. All stats are set to this value (meaning custom *Hidden Powers* are not possible).
* .hpEv: The number of *HP* EVs the Pokémon has.
* .atkEv: The number of *Attack* EVs the Pokémon has.
* .defEv: The number of *Defense* EVs the Pokémon has.
* .spAtkEv: The number of *Special Attack* EVs the Pokémon has.
* .spDefEv: The number of *Special Defense* EVs the Pokémon has.
* .spdEv: The number of *Speed* EVs the Pokémon has.
* .ball: If set to TRAINER\_EV\_CLASS\_BALL and TRAINER\_CLASS\_POKE\_BALLS is defined, then the ball loaded will be the one for the trainer class. Otherwise look in **include/new/catching.h** for a list of available Poké Ball types.
* .ability: The ability number of the Pokémon. Can be set to:
  + Ability\_1: The Pokémon will have its first ability.
  + Ability\_2: The Pokémon will have its second ability.
  + Ability\_Hidden: The Pokémon will have its [hidden ability](https://bulbapedia.bulbagarden.net/wiki/Ability#Hidden_Abilities).
  + Ability\_Random\_1\_2: The Pokémon will have one of its primary abilities.
  + Ability\_RandomAll: The Pokémon will have one of its possible abilities.

Example:



In the above example, the nature for the spread is set to *Docile*, each stat has an IV of 31, the *HP* stat has 128 EVs, the *Attack* stat has 252 EVs, and the *Speed* stat has 128 EVs. All EV stat not referenced in this spread will be set to a default value of 0. Any Pokémon using this spread will also be released in a *Cherish Ball* and have any one of it possible abilities (including its hidden ability).

### Battle Frontier

//TO WRITE

### Upgraded TM/HM Expansion

This engine includes a vastly improved TM/HM system over those created in the past. This version allows the hacker to use ANY item id for their TMs. None of this item ID sorting or item gap nonsense! It also fixes the graphical problems with high TM numbers in the TM case.

To set this feature up, you must set the *Mystery 2* byte of your item to the TM Id, starting at 01 for TM01. The HMs must start at a TM Id **after** the max number of TMs you plan on including. For example, if you want 120 TMs and 8 HMs, HM01 would be given a TM Id of 121. Otherwise, the items are set up as normal (see image ->). The item sprite and palette don’t matter, as the disk is loaded outside of the item data, and the TM animation must be removed for the expansion to work correctly.

This engine does not mess with any of the original tables to allow a hack in progress to keep all its data. Therefore, once all your item data is set up, you must set up the *TM Move Table* and *TM Compatibility Table*. The *TM Move Table* has a pointer at 0x125A8C and is simply a list of move IDs corresponding with the TM ID. The *TM Compatibility Table* is more complicated. Its pointer lies at 0x43C68 and is 16 bytes per species. Taken from [this tutorial](https://www.pokecommunity.com/showthread.php?t=401063):

The new TM compatibility is broken into 4 words (4 bytes each), where the first word corresponds to TMs 1-32, then 33-64, and so on. Each bit starting from the far right corresponds to a TM.  
  
For example, if you want to learn TMs 3, 7, 20, 54, 87, and 102:  
1-32: 00000000000010000000000001000100  
33-64: 00000000001000000000000000000000  
65-96: 00000000010000000000000000000000  
97-128: 00000000000000000000000000100000  
  
becomes, in hex: [00 08 00 44] [00 20 00 00] [00 40 00 00] [00 00 00 20]  
  
reversing each part and combining:  
44 00 08 00 00 00 20 00 00 00 40 00 20 00 00 00

### Reusable TMs

If you’ve defined *REUSABLE\_TMS* in *src/config.h*, you will then also need to set the *Mystery 1* byte to 1 for each of your TMs. This will tell the game to treat the item like an HM (except extra code has been written to sort TMs/HMs by the TM index. For more info see [TM/HM Expansion](#_Upgraded_TM/HM_Expansion).), preventing you from giving it to a Pokemon to hold, selling it, etc. That’s it!

### Pickup[[3]](#footnote-3)

The items found by the ability [Pickup](https://bulbapedia.bulbagarden.net/wiki/Pickup_(Ability)) can be modified in **src/Tables/Pickup\_Items.c**. Modify *sPickupCommonItems* and *sPickupRareItems* to change the items that appear. **DO NOT** add any new items to the tables; only change the pre-existing items. The default values were chosen based on the following table (common is highlighted in blue, rare is highlighted in red):



### Select from PC Hack

If SELECT\_FROM\_PC is defined, the player can select Pokémon directly from the PC to modify certain data. To initiate, set *pcSelect\_StateTracker* (defined in **asm\_defines.s**) to 1 before using Special 0x3C. The hack will store the box number and slot to *Var 0x8000* and *0x8001*, respectively. All data retrieval/manipulation specials will be able to access the selected PC Pokémon by setting *Var 0x8003* to 1. For example, nicknaming a Pokémon in the PC:

#define pcSelect\_StateTracker 0x203B7AC

#org @start

writebytetooffset 0x1 pcSelect\_StateTracker

special 0x3C 'Select boxed mon, box stored to Var8000, slot to Var8001

waitstate

writebytetooffset 0x0 pcSelect\_StateTracker

compare LASTRESULT 0x7F

if 0x1 goto @NothingSelected 'User cancelled out of the PC menu

bufferpokemon 0x0 0x8002

setvar 0x8003 0x1 'Data source is in the PC Box

special 0x7d

compare LASTRESULT 0x1

if 0x1 goto @traded

special 0x9e

waitstate

end

### Time of Day Based Wild Encounters

In order to use this feature, TIME\_ENABLED must be defined. Once it is open the file **src/Tables/Wild\_Encounter\_Tables.c**. Find where it says //Modify this section. This is the data that will be modified.

**Example: Modifying the Night Data**

For this example, HOOTHOOT will be added onto ROUTE 1 in place of PIDGEY.

First, the wild data needs to be created. If you open up AdvanceMap to the wild data for ROUTE 1, you should see this:



Now that you know what the data looks like, convert it into a C structure (see image below) and paste it at the top file, directly under the line “#ifndef UNBOUND //Modify this section”, or under previously created wild data structures. No matter what, it must be placed above the line

“const struct WildPokémonHeader gWildMonMorningHeaders[] =”:



^ This line is very important and must be added in as well. Make sure the label correctly matches the name for your newly created wild data structure. The *21* is the encounter rate.

Now that the wild data has been copied from AdvanceMap, it’s time to make modifications.

Change all the “PKMN\_PIDGEY” to “PKMN\_HOOTHOOT”:



The data for Route 1 at night is now complete. As this is data for night time, we need to add it to our night table, *gWildMonNightHeaders*. Make sure you leave the pre-existing entry at the bottom of the table:



The only thing left to do now is define “*MAP\_ROUTE\_1*”. Go back to AdvanceMap and find the map bank and map number for ROUTE 1 (the map bank is 3, and the map number is 19). At the top of the file, add a line formatted like #define MAP\_NAME ((MAP\_BANK << 8) | MAP\_NUM)



**Make sure the map name matches what’s in the brackets for “.mapGroup” and “.mapNum” (ie. ROUTE\_1 became MAP\_ROUTE\_1).**

Now wild night data has successfully been added for Route 1. Morning and Evening data follow the same pattern. Any route that doesn’t have morning or night data defined will load the standard day data set in AdvanceMap.

For water, fishing, or Rock Smash data, follow the same steps, but look [here](https://github.com/pret/pokeemerald/blob/c74bfbe11017e4f937dc6466a93a6d3dc72cdac7/src/data/wild_encounters.h) to see how to structure those kinds of wild datasets.

If you followed everything correctly, here is what the file should look like now:  




### Swarms

Also known as “[mass outbreaks](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_outbreak)”, swarms are when a certain species of Pokémon that doesn’t normally appear in the wild, makes an appearance for a single day. To get the most out of this feature, the real-time clock is utilized to run a function daily that updates the swarms. To add a swarm, in the file **src/Tables/Wild\_Encounter\_Tables.c**, search for the first occurrence of *gSwarmTable*. Each element in this table is a struct with the following members:

* mapName: The map name (or section) id where the swarming Pokémon will be located.
* species: The species to swarm in locations with the given mapName.

For example, to make *Sentret* swarm on *Route 1*, add the following to the table (the levels are loaded from *Route 1*’s wild data):



The species names can be found in **include/constants/species.h**. The map names are not predefined anywhere (use AdvanceMap 1.92’s world map editor as a map name reference), so before this entry is added, the following line needs to be added above the table (or in an external, included file):



To buffer swarm text so an NPC can tell the player what and where a Pokémon is swarming, see *special 0x58*.

### Roaming Pokemon

[Roaming Pokémon](https://bulbapedia.bulbagarden.net/wiki/Roaming_Pok%C3%A9mon) have been staples in the Pokémon games since Gen 2. This engine allows you have up to 10 roaming Pokémon at once, where, if one is encountered, can be trapped, battled, and captured by the player. To set up Pokémon that roam around, use *special 0x129* (see the example script for *special 0x59*).

In order to control where Pokémon can roam, open up **src/roamer.c** and look for sRoamerLocations. The roaming movement is divided into several sets of maps with each set containing at most seven different maps (each row in the table is a set). The different maps are and should be defined in **include/constants/maps.h**. Every time the player changes maps, the roamer moves as well. Most likely they will move within a given set, but they may change sets entirely. It’s recommended to only place closely related maps within each set. So, for example, if you had two regions, you’d want one set containing only maps from region A, and another set containing only maps from region B. As many sets as you’d like can be added to the table, however the table must be terminated with:



### Day & Night System

A brand new DNS has been included in the engine. It features dynamic palette fading throughout the day, as well as options to allow windows to light up at night. Both of these options can be customized by editing **include/new/dns\_data.h**. Note that since this is a **.h** file, if you want your changes to compile, you will need to make a change to the file **src/DNS.c** so the compiler recompiles the DNS file. The change can be as simple as adding a whitespace character and then removing it. **Don’t use the undo button to remove the change!** The change needs to be removed manually by using the backspace character so the editor saves the file with an updated timestamp.

**Changing Which Palettes Are Faded**

Open up the DNS.h file and search for OW\_BG\_PAL\_0. Here is a listing of all palettes that can be faded. OW\_BG\_PAL\_0 through OW\_BG\_PAL\_15 represent the palettes of the background in the overworld. By default, OW BG palettes 0 - 11 are set to be faded, but 12 can be faded as well by making the following change:



Similarly, whether any other palette is faded can be changed by changing its value from TRUE to FALSE or vice versa. It is **NOT** recommended to fade palettes 13-15 as these are the colours used for the menus and text boxes.

Other than the OW backgrounds, the OW\_SPRITE\_PAL represent the sprites in the overworld, the BBG\_PAL defines represent the background in battle, and the B\_SPRITE\_PAL defines represent the sprites in battle.

**Changing the Colours Faded Throughout the Day**

In the same file is a table representing the actual fading colours (*gDNSNightFadingByTime*). Currently the table only has entries from 12 AM - 7:59 AM and 5 PM - 11:59 PM. The rest of the day no fading changes can be seen. If you would like to add fading for more time during the day, simply look for the line *Day has no fade* and start adding new entries there. For example, adding an entry for 8 AM - 8:59 AM:



The colour can change every ten minutes, so each of those colour indices represent a colour at that ten minute period.

**Light Up Windows**

There are currently two ways to handle light up windows. The first is more tedious. It involves leaving OW\_BG\_PAL\_12 as FALSE (see *Changing Which Palettes Are Faded*) and make all tiles you’d like to light up use that palette. Then, use an on-entry script in AdvanceMap to do setmaptiles and place your light up windows only when the time is night. This can be an extreme annoyance and a time consumer, which is why the second, new method was developed.

This new method uses a table to fade certain palette colours if it nighttime.

1. To start off, find the offset of the tileset containing the palette you wish to fade. For this example, I’ll be fading the windows of the player’s door in *Palette Town*. Looking in AdvanceMap, the player’s door uses tiles from *Tileset 1*, which has an offset of 0x82D4AAC.



1. Now open the tile viewer and determine which colours are the ones you wish to fade. For my example, I’ve determined that the player’s door uses palette 8, and the colours used are in indices 8, 9, and 10:



1. Now determine the colours you wish to change them to. It’s okay to play around with these colours in AdvanceMap. Hitting *Apply* won’t save anything permanently unless you exit out of the block editor and choose to save. For my example, I could going to change the window to a yellowy colour:



These colours can be represented by the RGB values of (31, 31, 20), (31, 31, 11), and (31, 31, 10).

1. Putting it all together now:



In the image *.tilesetPointer* is set to the offset of the tileset found in step 1, *.paletteNumToFade* is set to *8* representing the 8th palette. *.paletteIndicesToFade* has entries for index 8, 9, and 10, each with the corresponding colour they should be faded. After all that, is the line TILESET\_PAL\_FADE\_END. **DO NOT FORGET THIS LINE!**

Some more examples have been left in the file to help guide you further if you need it.



### Pre-Battle Mugshots

The battle transition from the Elite Four has been modified for Fire Red. [Originally created by Jambo51](https://www.pokecommunity.com/showthread.php?t=240183), the updated feature allows toggling of different transition styles with vars, as well as use custom images in place of the trainer’s front pic. To include this feature, FR\_PRE\_BATTLE\_MUGSHOT\_STYLE must be defined in **src/config.h**. Then, you must define var values for VAR\_PRE\_BATTLE\_MUGSHOT\_STYLE and VAR\_PRE\_BATTLE\_MUGSHOT\_SPRITE. The following table details the types of configurations you can use with these vars.

|  |  |
| --- | --- |
| **VAR\_PRE\_BATTLE\_MUGSHOT\_STYLE** |  |
| 0 | One Big Bar. Full Trainer Sprites |
| 1 | One Small Bar. Half Trainer Sprites |
| 2 | Two Small Bars. Half Trainer Sprites |

|  |  |
| --- | --- |
| **VAR\_PRE\_BATTLE\_MUGSHOT\_SPRITE** |  |
| 0 | Player’s Front Picture |
| 1 | “VS” Symbol instead of Player |

To load a custom image instead of the opponent’s trainer front sprite, set the flag FLAG\_LOAD\_MUGSHOT\_SPRITE\_FROM\_TABLE. The table *sPreBattleMugshotSprites* in **src/Tables/Mugshot\_Tables.c** allows you to set the custom image to load for that given trainer’s front pic index. Two examples exist to show you how to add one in for a different trainer picture.

As per Jambo’s original routine, the unused 3rd argument in the trainerbattle script command lets you turn on the pre-battle mugshot. Changing the value to *0x0Y00* lets you select different palettes for each transition tilemap (for a total of 15 different palettes). The palettes associated with each can be modified by changing the palettes of the images found in the directories **graphics/Battle\_Mugshots/Big** and **graphics/Battle\_Mugshots/DP**.

Finally, if you are using the Two Bar mugshot style, the 3rd trainerbattle argument only changes the palette of the opponent’s tilemap. The value of VAR\_MUGSHOT\_PLAYER\_PAL can be changed to load a different palette for the player’s side as well.

**Example Script:**

#define VAR\_PRE\_BATTLE\_MUGSHOT\_STYLE 0x5038

#define VAR\_PRE\_BATTLE\_MUGSHOT\_SPRITE 0x5039

#org @RivalBattle

msgbox @BattleMe 0x6

setvar VAR\_PRE\_BATTLE\_MUGSHOT\_STYLE 0x2

setvar VAR\_PRE\_BATTLE\_MUGSHOT\_SPRITE 0x0

trainerbattle 0x3 0x146 0x100 @LossMessage

end

### New Field Moves

The engine allows you to (relatively) easily add new field moves! Rock Climb, Dive, and Defog are already added for you. Note that this feature may require some coding on your behalf.

If you want to only adjust the added field moves, you can adjust the badge requirement in the array **gFieldMoveBadgeRequirements** inside *src/party\_menu.c*

1. Open up *src/party\_menu.c* and find the section “Field Moves”
2. Add your field move to the **enum FieldMovesIDs** section (always include **FIELD\_MOVE\_COUNT** at the end!). For example, let’s add ***FIELD\_MOVE\_SUNNY\_DAY****.*
3. Add your field move to the list of party menu options, **gPartyMenuCursorOptions** as so:

[MENU\_FIELD\_MOVES+FIELD\_MOVE\_SUNNY\_DAY] = {gMoveNames[MOVE\_SUNNY\_DAY], CursorCb\_FieldMove},

1. Create the appropriate callback for your field move inside **gFieldMoveCursorCallbacks:**

[FIELD\_MOVE\_SUNNY\_DAY] = {SetUpFieldMove\_SunnyDay, 0x0D},

Where *SetUpFieldMove\_SunnyDay* is a function you will have to create (see step 8)

1. Add the field move description to **gFieldMoveDescriptions:**

[FIELD\_MOVE\_SUNNY\_DAY] = gText\_FieldMoveDesc\_SunnyDay,

Where *gText\_FieldMoveDesc\_SunnyDay* can be defined in ./*strings/party\_menu.string* and declared at the top of the “Field Moves” section of *src/party\_menu.c* like:

*extern const u8 gText\_FieldMoveDesc\_SunnyDay[];*

1. Link the field move and move ID inside **gFieldMoves** (keep FIELD\_MOVE\_COUNT at the end!)

[FIELD\_MOVE\_SUNNY\_DAY = MOVE\_SUNNY\_DAY,

1. Add the badge requirement of your new field move to **gFieldMoveBadgeRequirements:**

[FIELD\_MOVE\_SUNNY\_DAY] = 6 // requires badge 6 to use, for example

1. Now we must include the logic for seeing if we can use our field move! This is dependent on what the new field move is. Ostensibly, Sunny Day can only be used outside, which can be checked the same way as fly, for our intents and purposes. We might also want to make sure it isn’t already sunny via ***GetCurrentWeather*.**So, our function might look like (feel free to adjust):

static bool8 SetUpFieldMove\_SunnyDay(void)

{

if (GetCurrentWeather() == WEATHER\_SUNNY)

return FALSE;

else if (Overworld\_MapTypeAllowsTeleportAndFly(gMapHeader.mapType) == TRUE)

{

gFieldCallback2 = FieldCallback\_PrepareFadeInFromMenu;

gPostMenuFieldCallback = FieldCallback\_SunnyDay;

return TRUE;

}

return FALSE;

}

These functions return TRUE if the field move is usable and FALSE if not. Most also set up some callbacks to run a script upon exiting the party menu. In our case we want to run *FieldCallback\_SunnyDay*, which we can emulate from *FieldCallback\_RockClimb* as so:

static void FieldCallback\_SunnyDay(void)

{

((u32\*) gFieldEffectArguments)[0] = GetCursorSelectionMonId();

ScriptContext1\_SetupScript(SystemScript\_SunnyDay);

}

We will then have to create our script of *SystemScript\_SunnyDay*, either in XSE or in the CFRU engine via *assembly/overworld\_scripts/system\_scripts.s* (see Step 9)

1. Our script will only run if the field move is usable, so we don’t have to worry about adding a ton of logic into our script. For sunny day, all we need to do is display the HM bar animation and then set the new weather, like so

.global SystemScript\_SunnyDay

SystemScript\_SunnyDay:

lockall

doanimation 0x28

waitstate

setweather 0x1

doweather

msgbox gText\_BecameSunny MSG\_KEEPOPEN @define this in strings/overworld\_strings.string

releaseall

end

1. Everything should be working now!

# Other Features Included

### Save Expansion[[4]](#footnote-4)

The default save space has been expanded tremendously to allow for:

* 4096 new Flags (0x900 - 0x18FF)
* 200 new Vars (0x5000 - 0x51FF)
* 10 new PC Boxes (for a total of 24)
* Up to 10 Roaming Pokémon
* Up to 778 unique items in the bag
* 4 New Step Counters

### Updated & New Item Effects

Several new items effects have been added such as effects for the [Power Items](https://bulbapedia.bulbagarden.net/wiki/EV-enhancing_item) and the [Shiny Charm](https://bulbapedia.bulbagarden.net/wiki/Shiny_Charm), a complete set of battle effects, as well as some other items like the [Black Flute](https://bulbapedia.bulbagarden.net/wiki/Black_Flute) and [White Flute](https://bulbapedia.bulbagarden.net/wiki/White_Flute) (which have been updated to the standards from ORAS).

### Trainer Face Fix

The player will face trainers before battle.

### DexNav[[5]](#footnote-5)

A simplified [DexNav](https://bulbapedia.bulbagarden.net/wiki/DexNav) system for Fire Red is included in the engine. It replaces the *POKEDEX* option in the Start Menu with *TOOLS*, which can either load a multichoice menu containing the *Pok*é*dex* and *DexNav*, or an entirely new start menu. Selecting the *DexNav* opens the graphical user interface (GUI) to allow the player to view what Pokémon they have caught/seen on the current map, and either press the *R-Button* or *A-Button* on a given Pokémon to search for it on the map. *R* will save the Pokémon information to DEXNAV\_VAR and allow the player to search for that Pokémon via *R* from the overworld.

### Dynamic Overworld Palettes[[6]](#footnote-6)

Overworld sprites are now loaded in dynamically, allowing for more freedom with creating new overworld sprite palettes.

### Ability Pop-Ups[[7]](#footnote-7)

Whenever a Pokémon’s ability activates in battle, a pop-up will appear showing the ability akin the Gen 5+ games.

### Hidden Abilities[[8]](#footnote-8)

[Hidden abilities](https://bulbapedia.bulbagarden.net/wiki/Ability#Hidden_Abilities) have been implemented using one of the empty bytes in the Pokémon data.

### Expanded Text Names

The following text names can expanded by replacing the first four characters in the name with a pointer to a longer text string:

* Trainer Class Names
* Item Names

Additionally, ability names have had their name length changed from 12 to 16.

### Pokédex Screen Stats[[9]](#footnote-9)

The boring old size comparison when viewing a Pokémon’s pokedex data has been replaced with a routine by that displays the Pokémon’s stats and abilities instead.

### Turbo Boost

A pipelined routine has been written to replace the old main loop with one that now allows the game to run at over 1000% speed using the fast-forward button.

### Various Customizable Updates

The file **bytereplacement** contains a list of optional byte changes:

* Don’t count eggs while healing at the Pokémon Centre.
* Extend number of direct sound tracks to 12.
* Eggs hatch at level 1 (there is also a config that goes along with this)
* EV cap lowered to 252.
* Fix for movement type 0xC (Hidden).
* Remove *National Dex* evolution limiters.
* Remove plot related trade restrictions.
* Fix for *Pokédex* species name issue.
* Display foreign Pokémon’s id without *National Dex*.
* PC boxes use more wallpapers.
* Max money increased to 9999999.

To remove any of these changes, simply add a **#** symbol at the beginning of the line or delete the byte change lines entirely.

### Triple Layer Tiles10

Taken from [this source](https://www.pokecommunity.com/showthread.php?t=352725), blocks can now have three layers on top of each other. See the original post for details on using this feature.

### Expanded Coins

The expanded coins options also comes with rewritten coin-related scripting commands. The original scripting commands only allow an argument of up to *0xFFFF*, but if we’ve expanded coins up to *999,999,999*, the command would need to be repeated multiple times. Now, if you use the coin commands (***checkcoins***, ***givecoins***, ***removecoins***) with argument ***0xFFFF***, the value will be loaded from *Var8000* and *Var8001*. The upper halfword will be the value in *Var8000*, and the lower halfword will load from *Var8001*.

**Example Script:**

#org @giveLotsOfCoins

lock

setvar 0x8000 0x1234

setvar 0x8001 0x5678

givecoins 0xFFFF ‘the player will receive 0x12345678 (or 305,419,896) coins

release

end

### Multichoice Windows

#### Static Multichoice Windows

Like in JPANs Fire Red Hacked Engine, you can script up custom multichoice boxes for players to select from various options. The hack uses [Special 0x24](#_Other_Specials_1) or [Special 0x25](#_Other_Specials_1) to add list items, and the ***multichoice*** scripting command to display the list. The first two arguments in this command are the x,y coordinates on the screen. The third argument is the number of list items you have (0x20 is 2 options, 0x25 is 7 options, the maximum). The final argument is either *0* if B allows the player to cancel, or *1* if the player cannot cancel with the B button. Here is a scripting example using special 0x25:

#org @StaticMultichoiceExample

setvar 0x8006 0x0 ‘First list item

loadpointer 0x0 @text1

special 0x25

setvar 0x8006 0x1 ‘2nd list item

loadpointer 0x0 @text2

special 0x25

setvar 0x8006 0x2 ‘3rd list item

loadpointer 0x0 @text3

special 0x25

preparemsg @msg ‘String to display in the message box

waitmsg

multichoice 0x0 0x0 0x21 0x1 ‘Multichoice at (0x0), 3 options, ‘B’ CANNOT cancel.

compare LAST\_RESULT 0x0

if 0x1 goto @firstOption

//etc…

#### Scrolling Multichoice Windows

Ported from [FBIs asm routine](https://www.pokecommunity.com/showpost.php?p=8903237&postcount=692) and improved slightly, this feature allows you to include multichoice boxes with more than 6 options at once, such as in the example to the right. To set these up, go to the bottom of *src/scripting.c* (or jump there by searching “Scrolling Multichoice”). You can define new multichoice lists by adding ***static const u8\* sMultichoiceSet3[] =***  {…}, where the brackets are filled with pointers to the text you want to add. You can include strings inside the engine by emulating the example strings inside *strings/scrolling\_multichoice.string*, eg.

**#org @sExampleText\_11**

**Example 11**

And then add **extern const u8 sExampleText\_11[];**underneath *Text Declarations* in *src/scripting.c.*

Next, you must add the pointer to your new multichoice list to the set of multichoice lists, ***gScrollingSets***, defined below the multichoice sets. You can add your new list in the same format as the examples given: **{sMultichoiceSet3, ARRAY\_COUNT(sMultichoiceSet3)},**

Finally, here’s how you would display your new list in a script:

#org @ScrollingMultichoiceExample

setvar 0x8000 0x2 ‘Choose the 3rd multichoice list in *gScrollingSets*

setvar 0x8001 0x5 ‘Display 5 text items at once (defines the box height)

special 0x158

waitstate

compare LAST\_RESULT 0x7F

if 0x1 goto @cancelled

compare LAST\_RESULT 0x0

if 0x1 goto @firstOption

//and so on…

### Script Specials

Several new scripting specials have been added to the engine. Many have been ported from JPAN’s hacked engine, and thus will work similar to how they worked there.

If a special is shown to have a var (such as *Var 0x8000*) as an input, set that var to the required data. If SELECT\_FROM\_PC is defined, remember to keep track of the Pokémon source from *Var 0x8003* before calling specials that manipulate Pokémon attributes.

If a special is shown to have a return value, it must be called with the **special2** scripting command.

#### Pokémon Specials

The following specials check or change Pokémon attributes. If SELECT\_FROM\_PC is defined, *Var 0x8003* will allow you to check/change data from PC boxed Pokémon if it is set to 1. Otherwise it will check/change from a party Pokémon.

Special 0x7 – EV/Contest Stat Checker

**Details**: Checks a party/boxed Pokémon’s EVs or Contest stats.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Stat to check:

|  |  |
| --- | --- |
| HP EV | 0x0 |
| Attack EV | 0x1 |
| Defense EV | 0x2 |
| Speed EV | 0x3 |
| Special Attack EV | 0x4 |
| Special Defense EV | 0x5 |
| Coolness | 0x6 |
| Beauty | 0x7 |
| Cuteness | 0x8 |
| Smartness | 0x9 |
| Toughness | 0xA |
| Luster | 0xB |

**Returns**: Stat value to given var.

**Example Script:**

setvar 0x8003 0x0 'Select from party

setvar 0x8004 0x3 '4th Pokémon in party

setvar 0x8005 0x1

special2 LAST\_RESULT 0x7

buffernumber 0x0 LAST\_RESULT 'Buffer Attack EV stat into [buffer1]

Special 0x8 – Pokémon IV Checker

**Details**: Checks a party/boxed Pokémon’s IVs

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* IV stat to check:

|  |  |
| --- | --- |
| HP IV | 0x0 |
| Attack IV | 0x1 |
| Defense IV | 0x2 |
| Speed IV | 0x3 |
| Special Attack IV | 0x4 |
| Special Defense IV | 0x5 |

**Returns**: IV stat value to given var.

**Example Script (with PC Selection Hack):**

writebytetooffset 0x1 0x0203b7ac 'Or whatever pcSelect\_StateTracker is set to

special 0x3C 'Select boxed mon, box stored to var8000, slot to var8001

waitstate

compare LAST\_RESULT 0x7F 'Player exited without selecting

if 0x1 goto @DidNotSelect

setvar 0x8003 0x1 'From boxed mon

setvar 0x8005 0x1 'Check attack IV

special2 LAST\_RESULT 0x8

buffernumber 0x0 LAST\_RESULT 'Buffer attack IV to [buffer1]

Special 0x9 – Pokémon Ribbon Checker

**Details**: Checks a Pokémon’s ribbons. The first 5 ribbons have values from 1 - 4 for Normal, Super, Hyper, and Master rank, so be sure to reference the correct bit value(s). See [this page](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_data_substructures_in_Generation_III#Ribbons_and_Obedience) for more info. These values cannot be checked from the PC as they are removed to compress box Pokémon size.

**Input**:

*Var 0x8004*: Holds the party slot number.

*Var 0x8005:* Ribbon bit to check. Here are the possible bits and known ribbon values.

|  |  |  |
| --- | --- | --- |
| **Ribbon** | **Bit(s)/Rank** | **Var 0x8005 Val** |
| Cool Ribbons | Normal (1) = bit 1 (0001)  Super (2) = bit 2 (0010)  Hyper (3) = bits 1,2 (0011)  Master (4) = bit 3 (0100) | 0  1  0 & 1  2 |
| Beauty Ribbons | Normal (1) = bit 4  Super (2) = bit 5  Hyper (3) = bit 4,5  Master (4) = bit 6 | 3  4  3 & 4  5 |
| Cute Ribbons | Normal (1) = bit 7  Super (2) = bit 8  Hyper (3) = bit 7,8  Master (4) = bit 9 | 6  7  6 & 7  8 |
| Smart Ribbons | Normal (1) = bit 10  Super (2) = bit 11  Hyper (3) = bit 10,11  Master (4) = bit 12 | 9  10  9 & 10  11 |
| Tough Ribbons | Normal (1) = bit 13  Super (2) = bit 14  Hyper (3) = bit 13,14  Master (4) = bit 15 | 12  13  12 & 13  14 |
| Champion | Bit 16 | 15 |
| Winning | Bit 17 | 16 |
| Victory | Bit 18 | 17 |
| Artist | Bit 19 | 18 |
| Effort | Bit 20 | 19 |
| Special 1 | Bit 21 | 20 |
| Special 2 | Bit 22 | 21 |
| Special 3 | Bit 23 | 22 |
| Special 4 | Bit 24 | 23 |
| Special 5 | Bit 25 | 24 |
| Special 6 | Bit 26 | 25 |
| Special 7 | Bit 27 | 26 |
| Fateful Encounter | Bits 28-30 | 27 - 30 |
| Obedience (Mew/Deoxys) | Bit 31 | 31 |

**NOTE**: the “hyper” status is a bit more challenging to check for, as the input is a bit number and these ribbon statuses require checking two bits. If this is something you are interested in implementing, you would need to remove the Normal bit upon receiving Super status, and then check for both bits with two separate special calls to determine the Hyper status.

**Returns:** 1 if the ribbon flag is set, 0 if not.

**Example Script:**

setvar 0x8004 0x2 '3rd mon in party

setvar 0x8005 15 'Check Champion ribbon

special2 LAST\_RESULT 0x9

buffernumber 0x0 LAST\_RESULT

compare LAST\_RESULT 0x1

if 0x1 goto @IsAChampion

'Else, Pokémon was not in the hall of fame

Special 0xA – Pokerus Timer Checker

**Details**: Checks the [Pokérus](https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9rus) virus timer on a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Pokérus time left to given var.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x0 'First party Pokémon

special2 LAST\_RESULT 0xA

buffernumber 0x0 LAST\_RESULT 'Pokérus timer into [buffer1]

bufferpartypokemon 0x1 0x0 'Buffer first poke name into [buffer2]

msgbox @timeLeft 0x6

#org @timeLeft

= [buffer2] is sick for [buffer1] more cycles!

Special 0xB – Poké Ball Checker

**Details**: Check the Poké Ball type of a Pokémon. The ball Ids can be found in **include/new/catching.h**.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Poké Ball type to given var.

**Example Script:**

writebytetooffset 0x1 0x0203B7AC 'Select from PC hack

Special 0x3C 'Store box/slot into vars 0x8000, 0x8001, respectively

waitstate

Compare LAST\_RESULT 0x7F

If 0x1 goto @didNotSelect 'Player cancelled without selection

setvar 0x8003 0x1 'From box

special2 LAST\_RESULT 0xB

buffernumber 0x1 LAST\_RESULT 'Buffer item number to [buffer2]

special 0x7C 'Buffer boxed mon nickname to [buffer1]

msgbox @ball 0x6

#org @ball

= [buffer1] is inside a [buffer2]! How fortunate!

Special 0xC – Check Capture Location

**Details**:

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Capture Location Id to given var.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x5 'Last party mon

special2 LAST\_RESULT 0xC

buffernumber 0x0 LAST\_RESULT 'Buffer capture location to [buffer1]

Special 0xD – Happiness Checker

**Details**: Check the number of happiness points for a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Happiness value (0-255) to given var.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x2 '3rd Pokémon

special2 LAST\_RESULT 0xD

buffernumber 0x0 LAST\_RESULT 'Buffer happiness to [buffer1]

compare LAST\_RESULT 255

if 0x1 goto @maxedHappiness

Special 0xE – Hold Item Checker

**Details**: Check hold item value of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Hold Item Id to given var.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x3 '4th Pokémon

special2 LAST\_RESULT 0xE

buffernumber 0x0 LAST\_RESULT 'Buffer to [buffer1]

Special 0xF – Add/Subtract to EVs

**Details**: Add or subtract values to Pokémon EVs (between 0 and 252).

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Stat for math (see Special 0x7 for indices).

*Var 0x8006:* Value to add. 0x01YY to subtract YY, 0x00ZZ to add ZZ.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x1 '2nd party Pokémon

setvar 0x8005 0x3 'Speed EV

setvar 0x8006 0x0150 'Subtracting 0x50, or 80 speed EVs

special 0xF

Special 0x10 – Set IVs

**Details**: Set IV values for a Pokémon. No math here, just setting to a specific value.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* IV stat to change (see Special 0x8 for indices), between 0 and 31 (0x1F).

*Var 0x8006:* IV value to set.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x0 'First Pokémon

setvar 0x8005 0x0 'HP IV

setvar 0x8006 31 'Value to set

special 0x10 'Maximize first party Pokémon’s HP IV

Special 0x11 – Set Ribbons

**Details**: Set or clear a Pokémon’s ribbon flag.

**Input**:

*Var 0x8004:* Holds the party slot number.

*Var 0x8005:* Ribbon flag to set (see Special 0x9 for indices/values).

0x00XX will set a ribbon, 0x01YY will clear a ribbon.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0x2 '3rd Pokémon

setvar 0x8005 15 'Set Champion ribbon

Special 0x11 'Set the ribbon

Special 0x12 – Set Pokérus

**Details**: Set a Pokérus timer of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Number of cycles, 0x0 to 0xF, 0x10 to “cure”.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party.

setvar 0x8004 0x0 'First mon.

setvar 0x8005 0x10 'Cure Pokémon’s Pokérus.

special 0x12

Special 0x13 – Change Happiness

**Details**: Add or subtract to a Pokémon’s happiness.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Amount to add/subtract.

0x01YY will subtract YY from happiness; 0x00XX will add XX to happiness.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x1 '2nd party Pokémon

setvar 0x8005 0x0150 'Subtracting 0x50, or 80 friendship points

special 0x13

Special 0x14 – Change Pokeball

**Details**: Set the ball type of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Ball ID to set.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x0 'First Pokémon

setvar 0x8005 0x1 'Set to Master Ball

special 0x14

Special 0x15 – Change Hold Item

**Details**: Set the hold item of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Item Id to set.

**Note** that if the Pokémon is already holding an item, this will not change the item, unless *Var 0x8005* is set to 0 to remove the item. Then you can call it again to set a new hold item.

**Returns:** 0 to LAST\_RESULT if successful item change, 1 if not.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x5 'Last mon

setvar 0x8005 0x0 'Remove a hold item first

special 0x15

setvar 0x8005 ITEM\_SILKSCARF 'Item to give

special 0x15 'Give silk scarf

compare LAST\_RESULT 1

If 0x1 goto @Failed

Special 0x16 – Change Species

**Details**: Change the species of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Species to change to.

**Returns:** Nothing.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x0 'First Pokémon

setvar 0x8005 PKMN\_CHARMANDER 'Set to Charmander

special 0x16

Special 0x17 – Change Attacks

**Details**: Set or remove a move for a Pokémon. This one does not work with the PC Selection Hack.

**Input**:

*Var 0x8004:* Pokémon Slot (0-5)

*Var 0x8005:* Move Slot (0-3 for moves 1-4, respectively)

*Var 0x8006:* Move Id (0 to clear move slot)

**Returns:** Nothing.

**Example Script:**

Special 0x9F 'Select a Pokémon from the menu, store slot to Var 0x8004

waitstate

compare LAST\_RESULT 0x6

if 0x4 goto @Cancelled

setvar 0x8005 0 'First move

setvar 0x8006 MOVE\_HYPERBEAM 'Teach Hyper Beam in slot 0

special 0x17

Special 0x18 – Check Species

**Details**: Check the species of a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** Species Id to given var.

**Example Script:**

setvar 0x8003 0x0 'From party

setvar 0x8004 0x0 'Check first Pokémon

special2 LAST\_RESULT 0x18

compare LAST\_RESULT PKMN\_RATTATA 'Check if first Pokémon if Rattata

If 0x0 goto @NotCorrect

Special 0x19 – Check Attack PP

**Details**: Check a Pokémon move’s PP.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

*Var 0x8005:* Move slot (0-3).

**Returns:** PP left to given var.

**Example Script:**

setvar 0x8003 0x0 'From Party

setvar 0x8004 0x0 'First Pokémon

setvar 0x8005 0x0 'First move

special2 LAST\_RESULT 0x19

buffernumber 0x0 LAST\_RESULT

compare LAST\_RESULT 0x0

if 0x1 goto @NoPPLeft

#### Party Specials

Special 0x62 – Erase Pokémon

**Details**: Erase a Pokémon from your party, or the entire party.

**Input**:

*Var 0x8004:* Slot to erase (0xF for entire party).

**Returns: Nothing.**

**Example Script:**

setvar 0x8004 0xF 'Erase entire party

Special 0x62

Special 0x63 – Status Checker

**Details**: Check the primary status of a Pokémon.

**Input**:

*Var 0x8004:* Pokémon Slot

**Returns:** Status inflicted to given var.

|  |  |  |
| --- | --- | --- |
| **Status** | **Bits** | **Hex Value** |
| Sleep | 1  2  3 | 0x1  0x2  0x4 |
| Poison | 4 | 0x8 |
| Burn | 5 | 0x10 |
| Frozen | 6 | 0x20 |
| Paralyzed | 7 | 0x40 |
| Badly Poisoned | 8 | 0x80 |

**Example Script:**

setvar 0x8004 0x0 'First party Pokémon

special2 0x8004 0x63 'Get statuses to var 0x8004

setvar 0x8005 0x80 'Badly poisoned

special2 LAST\_RESULT 0x42 'Var 0x8004 & Var 0x8005

compare LAST\_RESULT 0x1 '& will return 1 if the Pokémon has this status

If 0x1 goto @BadlyPoisoned

Special 0x64 – Status Inducer

**Details**: Inflict a primary status on a party Pokémon.

**Input**:

*Var 0x8004:* Pokémon slot, or 0xF for entire party.

*Var 0x8005:* Status(es) to induce (see Special 0x63 for values).

*Var 0x8006*: 1 if status should only be given to Pokémon that can be afflicted with it (ie. No paralysis on Electric-types), 0 otherwise.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0xF 'First party Pokémon

setvar 0x8005 0x20 'Freeze entire party

setvar 0x8006 0x1 'Don’t freeze Ice-types of Pokémon with Magma Armor

special 0x64

Special 0x65 – Check Pokémon HP

**Details**: Check the amount of HP remaining for a party Pokémon.

**Input**:

*Var 0x8004:* Pokémon slot.

**Returns:** HP to given var.

**Example Script:**

setvar 0x8004 0x0 'First Pokémon

special2 LAST\_RESULT 0x65

compare LAST\_RESULT 0x0

if 0x1 goto @DeadPoke

Special 0x66 – Inflict Party Damage or Heal

**Details**: Inflict damage on/heal a Pokémon, or entire party.

**Input**:

*Var 0x8004:* Pokémon slot, 0xF for entire party.

*Var 0x8005:* Damage to inflict/heal.

*Var 0x8006:* 1 to heal, otherwise inflict damage.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0xF 'Entire party

setvar 0x8005 20 '20 damage to party

setvar 0x8006 0x0 'Damage party

Special 0x66

Special 0xB0 – Load Party Pokémon Types

**Details**: Gets the types of the requested party Pokémon.

**Input**:

*Var 0x8000:* Party Pokémon number.

**Returns:**

*Var 0x8000:* Type 1.

*Var 0x8001:* Type 2.

**Example Script:**

Special 0x9F 'Select a Pokémon from the menu, store slot to Var 0x8004

waitstate

compare LAST\_RESULT 0x6

if 0x4 goto @Cancelled

copyvar 0x8000 LAST\_RESULT

special 0xB0

Special 0xB2 –Pokémon Type In Party

**Details**: Gets the types of the requested party Pokémon.

**Input**:

*Var 0x8000:* Pokémon type.

**Returns:** *Last Result:* TRUE if type was found in party. 0 otherwise.

**Example Script:**

setvar 0x8000 0xA 'Check Fire-type in party

special 0xB2

compare LAST\_RESULT 0x0

if == goto @NotFound

Special 0xCC –Pokémon in Party that Can Learn Draco Meteor

**Details**: Checks if any Pokémon in the player's party can learn Draco Meteor. It does not include a friendship check. That is handled in the tutor part.

**Input**: None.

**Returns:** *Last Result:* TRUE if any Pokémon can learn Draco Meteor. 0 otherwise.

**Example Script:**

special 0xB2

compare LAST\_RESULT 0x0

if == goto @NoDragonTypesInParty

Special 0xD0 - Pokémon in Party that Can Learn TM/HM

**Details**: Checks if any Pokémon in the player's party can learn the given TM/HM number.

**Input**: Var8000: The TM/HM id to check (1 to NUM\_TM + NUM\_HMS).

**Returns:** *Given Var:* Slot Id of the Pokémon the can learn the TM/HM. 6 if none can learn.

**Example Script:**

setvar 0x8000 0x1 'Check Focus Punch

special2 LAST\_RESULT 0xD0

compare LAST\_RESULT 0x0

if == goto @NoFocusPunchPotentialInParty

#### Key Specials

Special 0x2B – Check AB Buttons

**Details**: Check if *A* or *B* has been pressed

**Input**: Nothing.

**Returns:** To the given var:

0x0: Neither pressed.

0x1: *A* was pressed.

0x2: *B* was pressed.

0x3: Both *A* & *B* were pressed.

**Example Script:**

#org @Loop

special2 LAST\_RESULT 0x2B

compare LAST\_RESULT 0x1 'Check for A

if 0x0 goto @Loop 'Player cannot continue onwards until they press A

'Continue

Special 0x2C – Check D-Pad

**Details**: Check *D-Pad* presses.

**Input**: Nothing.

**Returns:** To given var:

0x0: No direction is pressed.

0x1: *Up* is pressed.

0x2: *Left* is pressed.

0x3: *Down* is pressed.

0x4: *Right* is pressed.

0x5: *Up-left* is pressed.

0x6: *Up-right* is pressed.

0x7: *Down-left* is pressed.

0x8: *Down-right* is pressed.

**Example Script: Wait Until Player Presses Down Button**

#org @start

goto @loop

#org @loop

special2 LAST\_RESULT 0x2C

compare LAST\_RESULT 0x3 'Down pressed

if 0x1 goto @continue 'exit infinite loop if player pressed down

goto @loop

Special 0x2D – Check Start/Select

**Details**: Check if *Start*/*Select* are pressed

**Input**: Nothing

**Returns:** To given var:

0x0: Nothing pressed.

0x1: *Select* pressed.

0x2: *Start* pressed.

0x3: Both *Start* & *Select* pressed.

**Example Script:**

special2 LAST\_RESULT 0x2D

compare LAST\_RESULT 0x2 'Start pressed

If 0x1 goto @PressedStart

Special 0x2E – Check L/R

**Details**: Check if *L*/*R* are pressed.

**Input**: Nothing.

**Returns:** To given var:

0x0: Nothing pressed.

0x1: *R* pressed.

0x2: *L* pressed.

0x3: *L* & *R* pressed.

**Example Script:**

special2 LAST\_RESULT 0x2E

compare LAST\_RESULT 0x1 'R pressed

if 0x1 goto @PressedR

Special 0x2F – Dump Keys

**Details**: Dump any and all keys that have been pressed.

**Input**: Nothing.

**Returns:** Key presses to Var 0x800D (LASTRESULT):

|  |  |  |
| --- | --- | --- |
| **Key** | **Bit** | **Hex** |
| A | 1 = 0001 | 0x1 |
| B | 2 = 0010 | 0x2 |
| Select | 3 = 0100 | 0x4 |
| Start | 4 = 1000 | 0x8 |
| Right | 5 = 0001 0000 | 0x10 |
| Left | 6 = 0010 0000 | 0x20 |
| Up | 7 = 0100 0000 | 0x40 |
| Down | 8 = 1000 0000 | 0x80 |
| R | 9 = 0001 0000 0000 | 0x100 |
| L | 10 = 0010 0000 0000 | 0x200 |

**Example Script:** Wait until player presses A

#org @loop

special 0x2F

Compare LAST\_RESULT 0x1 'A pressed

If 0x1 goto @pressedA

goto @loop

Special 0xC9 – Force Key Input

**Details**: Force a key input from the user.

**Input**:

*Var 0x8004:* Key(s) to force (bitfield) (see *Special 0x2F* for bits).

*Var 0x8005:* Number of times to press it. For held buttons, a good rule of thumb is 0x10 times the number of times you want to move (see Script Example)

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0x10 'Right

setvar 0x8005 0x30 'Step Right 3 Times

special 0xC9 'Force player to move right 3 steps

Special 0xCA – Prevent Key Press

**Details**: Prevent player from being able to press button(s).

**Input**:

*Var 0x8004:* Key(s) to prevent (bitfield). 0 to allow all keys.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0x104 'Prevent *R* and *Select* from doing anything

Special 0xCA

Special 0xCB – Assign Key Script

**Details**: assign a specific script to a key

**Input**:

*Var 0x8004:* Key to assign script to (0 to remove).

*Loadpointer 0x0:* Script pointer.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0x100 'Assign to R

loadpointer 0x0 @DoSomething

special 0xCB 'Now, when in the overworld, pressing R will launch @DoSomething

#### Variable Math Specials

Special 0x3E – Add Variables

**Details**: Add the values of two variables together.

**Input**:

*Var 0x8004:* Variable holding the first value.

*Var 0x8005:* Variable holding the second value.

**Returns:**

*Variable Inside Var 0x8004:* Sum of two values.

*Given Var:* 1 if sum overflows 0xFFFF, 0 otherwise.

**Example Script:**

setvar 0x4050 50

setvar 0x4051 25

setvar 0x8004 0x4050

setvar 0x8005 0x4051

special2 LAST\_RESULT 0x3E 'Addition result to first variable, 0x4050

compare LAST\_RESULT 0x1

if 0x1 goto @overflow

Special 0x3F – Subtract Variables

**Details**: Subtract the values inside two variables.

**Input**:

*Var 0x8004:* Variable holding first value.

*Var 0x8005:* Variable holding second value.

**Returns:**

*First Variable:* Difference of values.

*Given Var:* 1 if difference underflows 0x0, 0 otherwise.

**Example Script:**

setvar 0x8004 0x4059

setvar 0x8005 0x4050

special2 LAST\_RESULT 0x3F '[var 0x4059] = [var 0x4059] – [var 0x4050]

compare LAST\_RESULT 0x1

If 0x1 goto @overflowed '[var 0x4059] > [var 0x4050]

Special 0x40 – Multiply Variables

**Details**: Multiply the values of two variables together.

**Input**:

*Var 0x8004:* Variable holding first value.

*Var 0x8005:* Variable holding second value.

**Returns:**

*First Variable: [Var 0x8004]* [*Var 0x8005]*.

*Given Var:* 1 if product overflows 0xFFFF, 0 otherwise.

**Example Script:**

setvar 0x8006 400

setvar 0x8007 200

setvar 0x8004 0x8006

setvar 0x8005 0x8007

special2 LAST\_RESULT 0x40 '400\*200 = 80000 = 0x13880 = (0xFFFF) + 0x3881

Special 0x41 – Divide Variables

**Details**: Divide the values in two variables.

**Input**:

*Var 0x8004:* Numerator.

*Var 0x8005:* Denominator.

**Returns:**

*Var 0x8004:* Integer result of Var8004 / Var8005

*Given Var:* Remainder (modulus) of the division.

**Example Script:**

setvar 0x8004 50

setvar 0x8005 6

special2 0x8006 0x41 'Var8004 = 50 / 6 = 8

buffernumber 0x0 0x8006 'Remainder = 2

Special 0x42 – AND Variables

**Details**: [Bitwise AND](https://en.wikipedia.org/wiki/Bitwise_operation#AND) two variables.

**Input**:

*Var 0x8004:* First value.

*Var 0x8005:* Second value.

**Returns:** AND result of two variables to given var.

**Example Script:**

setvar 0x8004 0xCB '1100 1011

setvar 0x8005 0xAA '1010 1010

special2 0x8004 0x42 '[1100 1011] & [1010 1010] = 1000 1010 = 0x8A

Special 0x43 – OR Variables

**Details**: [Bitwise OR](https://en.wikipedia.org/wiki/Bitwise_operation#OR) two variables

**Input**:

*Var 0x8004:* First value.

*Var 0x8005:* Second value.

**Returns:** OR result of two variables to given variable

**Example Script:**

setvar 0x8004 0x4 '0000 0100

setvar 0x8005 0x10 '0001 0000

special2 0x8004 0x43 'Var8004 = [0000 0100] | [0001 0000] = 0001 0100 = 0x14

Special 0x44 – XOR Variables

**Details**: [Bitwise XOR](https://en.wikipedia.org/wiki/Bitwise_operation#XOR) two variables.

**Input**:

*Var 0x8004:* First value.

*Var 0x8005:* Second value.

**Returns:** XOR result of two variables to given variable

**Example Script:**

setvar 0x8004 0x12 '0001 0010

setvar 0x8005 0x18 '0001 1000

special2 0x8007 0x44 'Var8007 = [0001 0010] ^ [0001 1000] = 0000 1010 = 0xA

#### Frontier Specials

Special 0x52 – Generate Frontier Trainer Id

**Details**: Generates a random battle facility id in gTowerTrainers or gSpecialTowerTrainers, or a preset frontier brain in gFrontierBrains found in **src/Tables/Frontier\_Trainers.c**, and a name for the trainer.

**Input**:

*Var 0x8000:* 0 = Load data for trainer Opponent 1.

1 = Load data for trainer Opponent 2.

2 = Load data for partner Trainer.

*Var 0x8001:* 0 = Load from gTowerTrainers.

1 = Load from gSpecialTowerTrainers.

2 = Load from gFrontierBrains.

*Var 0x8002:* 0 = If *Var 0x8001* is 2, then the frontier brain id num.

**Returns:** To given var the OW sprite Id of the chosen trainer.

Also buffers the trainer name to *gStringVar1*.

**Example Script:** See below.

Special 0x53 – Load Frontier Intro Battle Message

**Details**: Loads the battle intro message of the requested trainer.

**Input**:

*Var 0x8000:* 0 = Load data for trainer Opponent 1.

1 = Load data for trainer Opponent 2.

*Var 0x8001:* 0 = Load from gTowerTrainers.

1 = Load from gSpecialTowerTrainers.

2 = Load from gFrontierBrains.

**Returns:** Nothing

**Example Script:** For battle against a normal single opponent:

#define VAR\_RUNTIME\_CHANGEABLE 0x4080

setvar 0x8000 OPPONENT\_1 '0

setvar 0x8001 REGULAR\_TRAINER '0

special2 VAR\_RUNTIME\_CHANGEABLE SPECIAL\_GENERATE\_TOWER\_TRAINER '0x52

reappear FOE\_NPC\_ID 'The Person Id of the dynamic overworld person event

setvar 0x8000 OPPONENT\_1 '0

setvar 0x8001 REGULAR\_TRAINER '0

special BUFFER\_TOWER\_TRAINER\_INTRO\_MSG '0x53

callstd MSG\_NORMAL 'Displays the buffered intro text

Special 0x54 – Get Frontier Streak

**Details**: Gets the streak for the requested Frontier format.

**Input**:

*Var 0x8000:* 0 = Current Streak.

1 = Max Streak.

*Var 0x8001:* 0xFFFF = Load style from var BATTLE\_TOWER\_BATTLE\_TYPE.

0 = Single battle.

1 = Double battle.

2 = Multi battle.

3 = Link multi battle.

*Var 0x8002:* 0xFFFF = Load tier from var BATTLE\_TOWER\_TIER.

0 - 0xFF = See BATTLE\_TOWER\_TIER description for available tiers.

*Var 0x8003:* 1 - 5 = Party size option 1.

6 = Party size option 2.

*Var 0x8004:* 0 = Load level from BATTLE\_TOWER\_POKE\_LEVEL.

1-MAX\_LEVEL = Level bracket 1.

MAX\_LEVEL = Level bracket 2.

**Returns:** The requested streak to the given var.

**Example Script:**

setvar 0x8000 CURRENT\_STREAK '0

setvar 0x8001 LOAD\_STYLE\_FROM\_VAR '0xFFFF

setvar 0x8002 LOAD\_TIER\_FROM\_VAR '0xFFFF

setvar 0x8003 LOAD\_INVERSE\_FROM\_FLAG '2

setvar 0x8004 LOAD\_LEVEL\_FROM\_VAR '0

special2 LASTRESULT SPECIAL\_GET\_TOWER\_STREAK '0x54

Special 0x55 – Update Current Frontier Streak

**Details**: Updates the streak for the current Frontier format.

**Input**:

*Var 0x8000:* 0 = Increment streak by 1.

1 = Reset streak.

**Returns:** Nothing.

**Example Script:**

setvar 0x8000 INCREMENT\_STREAK\_BY\_1 '0

special SPECIAL\_UPDATE\_TOWER\_STREAK '0x55

Special 0x56 – Determine Battle Points To Give

**Details**: Gets the number of battle points to give for the player’s previous frontier win.

**Input**: None.

**Returns:** Given var:

|  |  |
| --- | --- |
| Streak Length | Num Battle Points |
| 1 - 10 | 2 |
| 11 - 19 | 3 |
| 20 | 20 (Against Brain) |
| 21 - 30 | 4 |
| 31 - 40 | 5 |
| 41 - 49 | 6 |
| 50 | 50 (Against Brain) |
| 51+ | 7 |

**Example Script:**

special2 LASTRESULT SPECIAL\_DETERMINE\_BATTLE\_POINTS '0x56

'Add vars here to update total BP amount

Special 0x6B – Replace Player Team With Multi Trainer Team

**Details**: Temporarily replaces the player's team with the team of the given multi battle trainer to allow the player to choose which Pokémon they want the partner to use.

**Input**:

*Var 0x8000:* Given multi trainer id of trainer in gFrontierMultiBattleTrainers.

**Returns:** Nothing.

**Example Script:** See *Special 0x6C*.

Special 0x6C – Splice Frontier Multi Trainer Team With Player’s Team

**Details**: To be used after special 0x6B. Merges the player's choice of partner Pokemon onto their team.

**Input**: None.

**Returns:** Nothing.

**Example Script:**

setvar 0x8000 0x0 'First trainer in gFrontierMultiBattleTrainers

special 0x6B 'Replace player’s team

special 0xF5 'Choose Pokémon for battle

waitstate

special 0x6C 'Splice result, even if player cancelled.

Special 0x6D – Load Frontier Multi Trainer Data By Id

**Details**: Loads any relevant multi trainer data by the given Id value.

**Input**:

*Var 0x8000:* 0-0xFE: Given multi trainer id of trainer in gFrontierMultiBattleTrainers.

0xFF: Random multi trainer id.

**Returns:** Given Var: The OW sprite Id of the chosen trainer.

Also buffers the multi trainer name to gStringVar2 ([BUFFER2] / bufferstring 0x1).

**Example Script:**

setvar 0x8000 0x0 'First trainer in gFrontierMultiBattleTrainers

special2 VAR\_RUNTIME\_CHANGEABLE 0x6D

Special 0x6E – Buffer Battle Sands Records

**Details**: Buffers text relating to Battle Sands records.

**Input**:

*Var 0x8000:* 0 = Previous Streak.

1 = Max Streak.

**Returns:** Last Result: TRUE if the requested record exists.

Also buffers strings to the following:

gStringVar1: Tier name.

gStringVar2: Battle format name.

gStringVar3: Level.

gStringVar7: Inverse on or off.

gStringVar8: Species 1.

gStringVar9: Species 2.

gStringVarA: Species 3.

gStringVarB: Streak length.

gStringVarC: "previous" or "max"

**Example Script:**

setvar 0x8000 0 'Previous record

special 0x6E 'Buffer records

msgbox @TellRecords MSG\_NORMAL

Special 0x6F – Can Team Participate in Battle Mine

**Details**: Checks if the player's team can enter the Battle Mine. Also sets the BATTLE\_TOWER\_TIER to the chosen tier.

**Input**:

*Var 0x8000:* 0 = Check Battle Mine Format 1.

1 = Check Battle Mine Format 2.

2 = Check Battle Mine Format 3.

**Returns:** LastResult: TRUE if the team can participate.

**Example Script:**

setvar 0x8000 0 'Check eligibility in Battle Mine Format 1

special 0x6F

compare LASTRESULT 0x0

if == goto @CantParticipate

Special 0x70 – Randomize Battle Mine Options

**Details**: Randomizes various battle options for a battle in the Battle Mine.

**Input**: None.

**Returns:** Buffers strings to the following:

gStringVar7: Tier name.

gStringVar8: Battle format name.

gStringVar9: Level.

gStringVarA: Party size.

gStringVarB: Inverse on or off.

**Example Script:**  special 0x70

Special 0x71 – Load Battle Mine Record Tier

**Details**: Sets the BATTLE\_TOWER\_TIER var to the correct tier the Battle Mine streaks are recorded in. To be used after a battle.

**Input**: None.

**Returns:** Nothing.

**Example Script:** special 0x71

Special 0x72 – Load Battle Circus Effects

**Details**: Loads random effects for Battle Circus battles.

**Input**: None.

**Returns:** Last Result: TRUE if all effects loaded.

gStringVarC: A description of the effect just activated.

**Example Script:**

setweather 0x0 'Remove overworld weather data

#org @SetEffectsForBattle

special 0x72 'Try to add effect

compare LASTRESULT 0x0

if != goto @Return

msgbox @BattleWillTakePlaceIn MSG\_NORMAL

doweather 'Try to update overworld weather

goto @SetEffectsForBattle

#org @Return

return

Special 0x67 – Generate Random Battle Frontier Team

**Details**: Generates a random Battle Frontier ready team using BATTLE\_TOWER\_POKE\_LEVEL. Teams are generated based on Pokémon found in **src/Tables/Frontier\_Spreads.h**

**Input**: Technically BATTLE\_TOWER\_POKE\_LEVEL.

**Returns**: Nothing.

**Example Script:**  special 0x67

#### Battle Specials

Special 0x51 – Can Team Participate in a Sky Battle

**Details**: Checks if the player has at least one Pokémon on their team that can participate in a [Sky Battle](https://bulbapedia.bulbagarden.net/wiki/Sky_Battle). The Pokémon species banned from participating can be found under gSkyBattleBannedSpeciesList in **src/Tables/Pokémon\_Tables.c**.

**Input**: None.

**Returns:** To given var 0 if team can’t participate, 1 if it can.

**Example Script:**

special2 0x51

compare LASTRESULT 0x0

if == goto @CantParticipate

setflag SKY\_BATTLE\_FLAG 'Defined in config.h

trainerbattle 0x3 0x20 0x0 @lose

Special 0x58 – Buffer Swarm Text

**Details**: Buffers the map name where there is currently a swarm to *buffer1* and the species name where there is currently a swarm to *buffer2*.

**Input**: None.

**Returns:** Nothing.

**Example Script:**

special 0x58

msgbox @Saw 0x6 '[buffer1]! They said there’s\na whole bunch of [buffer2] there!

Special 0x59 – Buffer Species Roaming Text

**Details**: Buffers the map name where the given roamer can be found to buffer1, and the species name of the roamer to buffer2.

**Input**:

*Var 0x8000:* Species

**Returns:** To given var 0 if the requested species isn’t found roaming. 1 otherwise.

**Example Script:**

setvar 0x8000 PKMN\_CHARMANDER 'Charmander should roam

setvar 0x8001 25 'Level 25

setvar 0x8002 0x1 'Can roam on land

setvar 0x8003 0x0 'Cannot roam on water

special 0x129 'Create roaming Pokemon

compare LASTRESULT 0x0

if 0x1 goto @TooManyRoamers

setvar 0x8000 PKMN\_CHARMANDER 'Find map where Charmander is roaming

special2 LASTRESULT 0x59 'Buffer roaming text

compare LASTRESULT 0x0

if 0x1 goto @NotRoaming

msgbox @Saw 0x6 '[buffer1]! They said a\n[buffer2] appeared there!

Special 0x5A – Wild Data Switch

**Details**: Overwrites the wild data on all maps until *special 0x5B* is used.

**Input**:

*Loadpointer 0x0:* New wild data header pointer.

**Returns:** Nothing.

**Example Script:**

loadpointer 0x0 0x83C9CB8 'Pointer to Route 1 wild data

special 0x5B

Special 0x5B – Cancel Wild Data Switch

**Details**: Cancels the wild data switch set by *special 0x5A*.

**Input**: None.

**Returns:** Nothing

**Example Script:**

special 0x5C

Special 0x97 – Random Grass Battle

**Details**: Initiates a random grass battle from the map wild data. Does nothing if no land data exists.

**Input**: None.

**Returns:** Nothing.

**Example Script:** See *Special 0x98*.

Special 0x98 – Random Sea Battle

**Details**: Initiates a random water battle from the map wild data. Does nothing if no water data exists.

**Input**: None.

**Returns:** Nothing.

**Example Script:**

#org @start

special 0x8F 'The trainer position special

special2 LAST\_RESULT 0x7F 'Returns 1 if ground battles, 2 if water battle

compare LAST\_RESULT 0x1

if 0x1 goto @grass

compare LAST\_RESULT 0x2

if 0x1 goto @water

release

end

#org @grass

special 0x97 'Generates a random grass battle.

release

end

#org @water

special 0x98 'Generates a random water battle.

release

end

Special 0x156 – Ghost Battle

**Details**: Initiate a ghost battle with a given Pokémon, level, and held item.

**Input**:

*Var 0x8004:* Ghost species. Setting to 0 initiates a battle with default Marowak.

*Var 0x8005:* Ghost level.

*Var 0x8006:* Ghost hold item.

**Returns**: Nothing.

**Example Script:**

setvar 0x8004 PKMN\_CHARIZARD

setvar 0x8005 100

setvar 0x8006 ITEM\_LEFTOVERS

Special 0x156

Special 0xAC - Load Second Trainer Defeat Message

**Details**: If a battle against two opponents is being started from a flag, this special will load in the defeat text for the second trainer.

**Input**:

*Loadpointer 0x0:* Pointer to defeat text.

**Returns**: Nothing

**Example Script**: See [here](#_Multi_Battles).

Special 0x9C – Old Man Battle

**Details**: Initiate an old man battle with a specific Pokémon species and level.

**Input**:

*Var 0x8004:* Species.

*Var 0x8005:* Level.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 PKMN\_BEEDRILL

setvar 0x8005 50

special 0x9C

waitstate



#### Timer Specials

Another feature from JPANs engine, which allows the player to utilize the game timer for timed events.

Special 0x46 – Start Timer

**Details**: Start the timer. If called after it started running, it resets the timer.

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x46

Special 0x47 – Pause Timer

**Details**: Pauses the already-started timer. Stores the timer value to *gTimerValue*

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x47

Special 0x48 – Resume Timer

**Details**: Resume a paused timer.

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x48

Special 0x49 – Stop Timer

**Details**: Stops the timer and returns the value. Timer needs to be started anew, resuming the timer will cause the value to be inaccurate.

**Input**: None.

**Returns:** The timer value to the given var.

**Example Script:** special2 LAST\_RESULT 0x49

Special 0x4A – Get Timer Value

**Details**: Just return the time on the timer.

**Input**: None.

**Returns:** The timer value to the given variable.

**Example Script:** special2 LAST\_RESULT 0x4A

Special 0x4B – Stop and Update Playtime

**Details**: Stop the timer and update playtime value.

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x4B

Special 0x4C – Update Playtime

**Details**: Update the playtime. This is meant for functions that take a while to process that cause delay in playtime.

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x4C

Special 0x4D – Check Timer Value

**Details**: Check if timer has reached a value stored in *Var 0x8010*.

**Input**:

*Var 0x8010:* Value to check against.

**Returns:** *1* if timer is greater or equal, *0* otherwise.

**Example Script:**

setvar 0x8010 100

special2 LAST\_RESULT 0x4D

Compare LAST\_RESULT 0x1 'Is timer >=

If 0x1 goto @timeReached

Special 0x4E – Save Timer Value

**Details**: Store the timer value to a free RAM address, *gTimerValue* to allow you to later reset it to this value.

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x4E

Special 0x4F – Start Timer at a Time

**Details**: Restart the timer at the value stored with Special 0x4E (value in *gTimerValue*).

**Input**: None.

**Returns:** Nothing.

**Example Script**: special 0x4F

Special 0x50 – Store Timer Value to Variable

**Details**: Store the timer value from *gTimerValue* to a given variable

**Input**:

*Var 0x8006:* Variable to store timer value to

**Returns:** Nothing.

**Example Script**:

Special 0x4e 'save timer value to *gTimerValue*

Setvar 0x8006 0x8000

special 0x50 'store timer value to variable 0x8000

Special 0x61 – Load Timer Value from Variable

**Details**: Set the timer value RAM, *gTimerValue*, from a variable

**Input**:

*Var 0x8006:* Variable holding timer value to set

**Returns:** Nothing.

**Example Script**:

Setvar 0x8000 50

setvar 0x8006 0x8000

special 0x61 'set *gTimerValue* to 50

#### Safari Specials

Special 0x86 – Get Safari Balls

**Details**: Check Safari Ball quantity.

**Input**: None.

**Returns:**

*Given Var:* Number of Safari Balls

**Example Script:**

special2 LAST\_RESULT 0x86

buffernumber 0x0 LAST\_RESULT

msgbox @numSafariBalls 0x6

#org @numSafariBalls

= You have [buffer1] Safari Balls remaining!

Special 0x87 – Change Safari Balls

**Details**: Increase or decrease the safari ball count, maximum up to MAX\_SAFARI\_BALLS.

**Input**:

*Var 0x8004:* Number to increase/decrease by (up to 100)

0x1XX decreases by XX, 0x1YY increases by YY.

**Returns:** Nothing.

**Example Script:**

setvar 0x8004 0x011E 'Remove 30 safari balls

special 0x87

Special 0x88 - Get Safari Pedometer

**Details**: Get the value of the safari pedometer.

**Input**: None.

**Returns:** Safari pedometer value to given variable.

**Example Script:**

special2 LAST\_RESULT 0x88

compare LAST\_RESULT 50 'Check if taken at least 50 steps

if 0x4 goto @OverFifty

Special 0x89 – Set Safari Pedometer

**Details**: Set a specific number of steps in the safari.

**Input**:

*Var 0x8004:* Pedometer Value to set.

**Returns: Nothing.**

**Example Script:**

special2 LAST\_RESULT 0x88 'Get current pedometer

Compare LAST\_RESULT 50

If 0x3 goto @Continue

setvar 0x8004 50 'Pedometer can only reach 50 steps

Special 0x89 'Set the safari step counter to 50

#### Walking Specials

Special 0x7E – Get Tile Number

**Details**: Get the tile number at a specified location on the current map.

**Input**:

*Var 0x8004:* Tile x-coordinate.

*Var 0x8005:* Tile y-coordinate.

**Returns:** Tile number to the given var.

**Example Script:**

getplayerpos 0x8004 0x8005 'Player’s current position

addvar 0x8004 0x2 'Check tile 2 steps to the right of the player

special2 LAST\_RESULT 0x7E

compare LAST\_RESULT SOME\_TILE\_NUMBER

if 0x1 goto @CorrectTile

Special 0x7F – Get Tile Behaviour

**Details**: Get a specific tile set of attributes

**Input**:

*Var 0x8004:* Tile x-coordinate.

*Var 0x8005:* Tile y-coordinate.

**Returns:**

*Var 0x8004:* Tile background byte.

*Var 0x8005:* Tile behaviour bytes.

*Given Var:* Tile background byte.

**Example Script:**

Getplayerpos 0x8004 0x8005

special2 LAST\_RESULT 0x7F 'Get tile attributes on player’s current tile

Special 0x81 – Set Walking Script

**Details**: Load a walking script to run each step. Setting to zero removes any walking script.

**Input**:

*Loadpointer 0x0:* Script pointer.

**Returns:** Nothing.

**Example Script:**

Loadpointer 0x0 @WalkingMsg

Special 0x81

release

end

#org @walkingMsg

Msgbox @msg 0x6

end

#org @msg

= This msgbox will play every single step.

Special 0x8A – Read Pedometer Value

**Details**: This special is not in JPAN’s original engine. It reads the value of one of the extra pedometers included in the engine, which are set with flags (see customization).

**Input**:

*Var 0x8004:* Pedometer to read

0: always active pedometer (32bit)

1: large valued-pedometer (32bit)

2: medium valued-pedometer (16bit)

3: first small pedometer (8bit)

4: second small pedometer (8bit)

**Returns:** Pedometer value to the given variable.

**Example Script:**

setvar 0x8004 0x0 'Pedometer that’s always on

special2 LAST\_RESULT 0x8A 'Get number of steps player has walked

buffernumber 0x0 LAST\_RESULT

#### PC Selection Specials

A couple new specials are added, as well as a few existing specials changed to allow data manipulation of boxed Pokémon.

Special 0x1A – Store/Return Party Pokémon Data

**Details**: Save or Return party/boxed Pokémon Data

**Input**:

*Var 0x8002*:

0 For store to free ram.

1 For return to party from free ram.

2 For store from free ram to box.

3 For store from box to free ram.

*Var 0x8005:* Party slot number (for special 0xFE inputs).

**Returns:** 0 or 1 to LAST\_RESULT for success/failure, respectively

**Script Example:**

setvar 0x8002 0 'Store

setvar 0x8005 0 'Save first party mon data

Special 0x1A 'First party Pokémon data now in Enemy data slot 5

Special 0x1B – Swap Party/Boxed Pokémon Data

**Details**: Swap party and box data

**Input**:

*Var 0x8000:* Box Number *.*

*Var 0x8001:* Box Position.

*Var 0x8002*: 0 for withdraw from box, 1 for store to box.

*Var 0x8005:* Party slot number (for special 0xFE inputs).

**Returns:** 0 or 1 to LAST\_RESULT for success/failure, respectively.

**Script Example – Swap Party and Boxed Mon**

Msgbox @ask 0x6 'Select party mon to deposit

Special 0x9F

waitstate

copyvar 0x8005 0x8004

setvar 0x8002 0 'From party to free ram

Special 0x1A ' Store from party to free ram

msgbox @ask2 0x6 'Select boxed mon to withdraw

writebytetooffset 0x1 0x0203b7ac

special 0x3C 'Select boxed mon, box stored to Var8000, slot to Var8001

waitstate

setvar 0x8002 0x0 'Withdraw

special 0x1B 'Selected boxed mon to selected party slot (Var8005)

setvar 0x8002 0x2 'Free ram to box

special 0x1A 'Free ram (eg. Original selected party mon) to same box slot

**NOTE**: Rather than use a single special for this, the dynamic inputs of these specials allow for swapping party/boxed mon, trading, and more.

Special 0x7C – Buffer nickname

**Details**: Buffer a Pokémon’s nickname to [buffer1].

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns: Nothing.**

**Example Script:** See *Special 0x9E*.

Special 0x7D – Check Traded Pokémon

**Details**: Check if Pokémon is traded.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** 0 if traded, 1 if not.

**Example Script:** See *Special 0x9E*.

Special 0x9E – Nickname Pokémon

**Details**: Nickname a Pokémon.

**Input**:

*Var 0x8003:* From party (0), or box (1).

From Box: *Var 0x8000*, *Var 0x8001* hold the box num/slot, respectively.

From Party: *Var 0x8004* holds the party slot number.

**Returns:** 0 if traded, 1 if not.

**Example Script – Nickname a Boxed Pokémon**

writebytetooffset 0x1 0x0203B7AC

special 0x3C 'Select boxed mon, box stored to Var8000, slot to Var8001

waitstate

setvar 0x8003 0x1

special 0x7C 'Buffer nickname

Msgbox @AskNickname 0x5 'Nickname [buffer1]?

Compare LAST\_RESULT 0x1

If 0x0 goto @Nope

Special 0x7D 'Check traded mon

Compare LAST\_RESULT 0

If 0x1 goto @Traded

Special 0x9E 'Nickname boxed mon

waitstate

#### Other Specials

Special 0x24 – Add Multichoice Text By Variable

**Details**: Add a dynamic multichoice option by variables

**Input**:

*Var 0x8004:* Upper halfword of pointer.

*Var 0x8005:* Lower halfword of pointer.

*Var 0x8006:* Multichoice Index.

**Returns: Nothing.**

**Example Script:**

setvar 0x8004 0x0890

setvar 0x8005 0x5040

setvar 0x8006 0x0

Special 0x24 'Multichoice index 0 is string pointer 0x08905040

**NOTE:** Special 0x25 is easier to use, you don’t have to worry about upper/lower sections of a word.

Special 0x25 – Add Multichoice Text by Pointer

**Details**: Add a dynamic multichoice option by a pointer.

**Input**:

*Var 0x8006:* Multichoice Index.

*Loadpointer 0x0:* Pointer to string.

**Returns: Nothing.**

**Example Script:**

setvar 0x8006 0x0

loadpointer 0x0 @option1

special 0x25

setvar 0x8006 0x1

loadpointer 0x0 @option2

special 0x25

preparemsg @msg

waitmsg

multichoice 0x0 0x0 0x20 0x0 'See note below

compare LAST\_RESULT 0x0

if 0x1 goto @selectedOption1

compare LAST\_RESULT 0x1

if 0x1 goto @selectedOption2

**NOTE:** multichoice 0xX 0xY 0xWW 0xZ

0xX: X position of box.

0xY: Y position of box.

0xWW: Multichoice box index.

0x20: 2 options (min).

…

0x25: 7 options (max).

0xZ: 0x0 if *B* can cancel box, 0x1 if not

Special 0x74 – Is Pokémon Storage System Full

**Details**: Checks if there is space in the PC for a new Pokémon.

**Input**: None.

**Returns:** To Given Var: 1 if no space left in PC. 0 otherwise.

**Example Script:**

special 0x74

compare LAST\_RESULT 0x0

if == goto @YesSpace

Special 0x75 – Buffer Species

**Details**: Buffer a Pokémon’s species to [buffer3] and size to [buffer1].

**Input**:

*Var 0x8005:* Holds the variable that stores measurements.

*Var 0x8006:* Species to evaluate.

**Returns:** Nothing.

**Example Script:** See below

Special 0x76 – Measure Pokémon

**Details**: Play the measure Pokémon game.

**Input**:

*Var 0x8004:* Party slot of Pokémon.

*Var 0x8005:* Holds the variable that stores measurements.

*Var 0x8006:* Species to evaluate.

**Returns:** To the given var:

*1*: Pokémon is not of selected type.

*2*: Pokémon is smaller.

*3*: Pokémon is bigger, also stores biggest value in variable in *Var 0x8005.*

*4*: Sizes are equal.

Special 0x18B – Show Fossil Image

**Details**: Load a custom image into the fossil image window.

|  |  |  |
| --- | --- | --- |
| Pointer Table - Defined in **src/config.h**  Table can also be generated by the engine in **src/script\_specials.c**. Search for *gFossilImageTable*. | | |
| Name | Bytes | Description |
| Fossil Pointer | 4 | Pointer to a section of data that has the needed information for the function to work. |
| Palette Pointer | 4 | A simple uncompressed palette must be on the other end. |
| Fossil Data (Pointed to from “Fossil Pointer”) | | |
| Image Pointer | 4 | Pointer to the actual image. Uncompressed, 64x64 pixel, so each should take 4kb worth of space. |
| Constant bytes | 4 | Don’t know what they do, but when changed they mess up the whole picture. Must be 0008581b. |
| Null bytes | 8 | 8 zeros. Changing them will result in the same as messing with the above bytes. |

**Input**:

*Var 0x8004:* Image number.

*Var 0x8005:* X coordinate on screen.

*Var 0x8006:* Y coordinate on screen.

**Returns: Nothing.**

**Example Script:**

setvar 0x8004 0x2 'Show third image in table

setvar 0x8005 0x0 'At (0x0)

setvar 0x8006 0x0 'At (0x0)

Special 0x18B

Special 0x9A – Stop Sounds

**Details**: Stops all sound effects currently playing

**Input**: None.

**Returns:** Nothing.

**Example Script:**

sound 0x15

special 0x9A 'Stops the sound

Special 0xAE – Dynamic Clearflag

**Details**: Clears the flag loaded.

**Input**:

*Var 0x8000:* Flag to clear.

**Returns:** Nothing.

**Example Script:**

setvar 0x8000 0x2D 'Clear flag 0x2D

special 0xAE

Special 0xAF – Dismount Bicycle

**Details**: If the player is on the bicycle, they dismount.

**Input**: None.

**Returns:** Nothing.

**Example Script:** special 0xAF

Special 0xB1 – Choose Item From Bag

**Details**: Opens the bag and lets the player select an item.

**Input**:

*Var 0x8000:* 0 = Any pocket, Open from Item's pocket.

1 = Any pocket, Open from Key Item's pocket.

2 = Any pocket, Open from Poke Ball pocket.

3 = Any pocket, Open from Item's pocket.

4 = Berry Pouch

5 = TM Case

**Returns:** Var 0x800E: The item the player chose. 0 if they chose nothing.

**Example Script:**

setvar 0x8000 0x4 'Choose berry

special 0xB1

Special 0xB3 – Do Choose Number Screen

**Details**: Opens the naming screen and allows the player to input a number.

**Input**: None.

**Returns**:Last Result: The number entered. 0xFFFF if number was invalid.

**Example Script**:

special 0xB3

waitstate

compare LASTRESULT 0xFFFF

if == goto @NotValidNumber

Special 0x196 – Try To Copy TM/HM Name To Buffer1

**Details**: If the given item is a TM/HM, buffers its name.

**Input**:

*Var 0x8000:* Item id of TM/HM.

**Returns**: Given Var: 0 if couldn’t copy name. 1 if name copied.

gStringVar1: TM/HM Name.

**Example Script**:

setvar 0x8000 TM01

special 0x196

#### Time-Based Specials

Special 0xA0 - Check And/Or Set Daily Event

**Details**: Checks if a daily event has been run. It can also simultaneously set a daily event to “done”.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

*Var 0x8001*: Set to 0 if you just want to check if the event has been done. Any other value sets the daily event as “done”.

**Returns**: 0 if the event has already been completed. 1 otherwise.

Special 0xA1- Update Time in Vars

**Details**: Updates the time stored in a pair of vars to the current time.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

Special 0xA2 - Get Time Difference

**Details**: Gets the time difference between the data stored in a pair of vars and the current time. These vars should be set by *special 0xA1* or *special 0xA2*.

**Input**:

*Var 0x8000*: The first of a pair of vars containing the daily event data. Note that the var after this var is used as well (hence why “pair” was mentioned).

*Var 0x8001*: Set to one of the following values:

* *0* - Get the minute difference.
* *1* - Get the hour difference.
* *2* - Get the day difference.
* *3* - Get the month difference.
* *4* - Get the year difference.

**Example Script**:

#define SP\_DAILY\_EVENT 0xA0

#define SP\_UPDATE\_TIME\_IN\_VARS 0xA1

#define SP\_GET\_TIME\_DIFFERENCE 0xA2

#define DAILY\_EVENT\_VAR 0x50D2 'Also uses 0x50D3

#dynamic 0x740000

#org @start

setvar 0x8000 DAILY\_EVENT\_VAR

setvar 0x8001 0x0 'Don’t set daily event var to done

special2 LASTRESULT SP\_DAILY\_EVENT

compare LASTRESULT 0x0

if == goto @AlreadyDid

setvar 0x8000 DAILY\_EVENT\_VAR

special SP\_UPDATE\_TIME\_IN\_VARS

msgbox @havenot ‘“I have not done it.”

callstd MSG\_FACEPLAYER

end

#org @AlreadyDid

setvar 0x8000 DAILY\_EVENT\_VAR

setvar 0x8001 0x0 'Minute difference

special2 LASTRESULT SP\_GET\_TIME\_DIFFERENCE

buffernumber 0x0 LASTRESULT

msgbox @already ‘“\v\h02 minutes ago I did it.”

callstd MSG\_FACEPLAYER

end

Special 0xAD- Get Time Of Day

**Details**: Gets the current time of day.

**Input**:

*Var 0x8001*: 0 = Distinct times.

1 = Merge morning, day, and evening into day.

**Returns**: Given Var:

0 = Morning.

1 = Day.

2 = Evening.

3 = Night.

# Creating New Battle Mechanics

The following sections detail how to set up new moves, abilities, poke balls, and items to the battle engine. Because there is no cookie-cutter methodology for writing code for each ability or item effect, these sections only tell you how to set up the necessary data. The actual logic and coding are up to the user.

## Moves

1. Declaration
   1. Add a definition in ***include/constants/moves.h*** following the same format, *#define MOVE\_<your move name>*. **It should ideally go after *Hold Hands* before the z-moves.**
   2. This index must stay the same in all table positions! This is easily done by not editing any move IDs and adding new moves after *Hold Hands*.
2. Move Name
   1. add to the end of ***strings/Attack\_Name\_Table.string*** following the *#org @NAME\_<your name here>* format. **The move name “table” is generated directly from the ordering of the *#org* declarations in this file, so make sure the ordering is correct!**
3. Move Description
   1. Add to the end of ***strings/Attack\_Descriptions.string*** following the *#org @DESC\_<your name here>* format, **following the ordering of your move ID definitions from step 1!**
   2. Add a pointer in the *AttackDescriptionTable* table in ***assembly/data/Attack\_Description\_Table.s***, following the *.word DESC\_<your name here>* format, **following the ordering of your move ID definitions from step 1!**
4. Animation
   1. Add the animation pointer in the respective location in *AttackAnimationTable,* typically called *ANIM\_<your move name>*. F**ollow the ordering of your move ID definitions from step 1!**
   2. Create your animation. This is most easily done with animation command macros. See examples after the table in ***assembly/data/Attack\_Anim\_Table.s*** as well as the macros in ***Anim\_Defines.asm***
   3. If you are adding new backgrounds:
      1. Add the indexed image to ***graphics/Backgrounds/Animation\_Backgrounds/*** for a 256x112 image. 256x256 may go in ***Larger Anim BGs***. Other sizes you’ll need to create your own directory and *gritflags.txt* file for.
      2. Add the new BG index to .***/Anim\_Defines.asm***. It is easiest to add just after *BG\_SNUGGLE\_FOREVER*.
      3. Add the definition to ***assembly/data/Anim\_Backgrounds\_Graphics\_Defines.s*** following the same format. This step isn’t 100% necessary, but helps keep formatting consistent. If your file name is NewBg, your image declaration would be NewBgTiles, your palette would be NewBgPal, and the Map would be NewBgMap.
      4. Add the background to the animation background table in ***assembly/data/Anim\_Background\_Table.s***, following the ordering of your indices from step *4.b.ii.* The table goes [IMG\_PTR, PAL\_PTR, MAP\_PTR], so follow your definitions from *4.b.iii*.
      5. Your background can now be called in animations by your definition from *4.b.ii*, **eg. loadBG1 BG\_<your\_bg\_name>**
   4. If you are adding new particles:
      1. Add the indexed .png file to ***graphics/Attack\_Particles/***
      2. Define a new particle tag at the end of ***./Anim\_Defines.asm***. It is easiest to keep all current definitions and add to the end after *ANIM\_TAG\_MUD\_BOMB*
      3. Add the particle definitions to ***assembly/data/Anim\_Particle\_Graphics\_Defines.s***. Similar to the backgrounds, this isn’t necessary but keeps the naming convention consistent. For a file called *NewParticle.png*, your image header will be called *NewParticleTiles* and your palette will be *NewParticlePal*.
      4. Add the above definitions to *gBattleAnimPicTable* and *gBattleAnimPaletteTable* in ***assembly/data/Anim\_Particle\_Table.s***. These are two separate tables, the palettes following the images. So make sure you add *animparticle <your\_particle\_IMG>, (64 \* 64) / 2, ANIM\_TAG\_<your\_particle\_tag>* and *animparticlepal <your\_particle\_PAL, ANIM\_TAG\_<your\_particle\_tag>, 0x0* in the appropriate spots in each respective table.
      5. You can now call your particles by the defined particle tag in any *objtemplate* (follow examples in ***assembly/data/Attack\_Anim\_Table.s***)
5. Battle Data
   1. Add your static definition in the respective location inside *gBattleMoves*, found in ***src/Tables/battle\_moves.c***, following any existing move for each structure element definition.
6. You now have a fully defined move! You can add it to a Pokemon’s learnset inside ***src/Tables/level\_up\_learnsets.c***, give it to a trainer’s pokemon, add in some AI logic, and whatever else you desire.

## Abilities

1. Ability ID – add your *#define ABILITY\_<your\_ability>* at the end of ***include/constants/abilities.h***. Then adjust *ABILITIES\_COUNT* to the last ability index +1.
2. Ability Name – add *#org @NAME\_<your\_ability>* and the ability name in ***strings/Ability\_Name\_Table.string***. Make sure the positioning of your ability name matches the index in Step 1.
3. Ability Description – add the header, *#org @DESC\_<your\_ability>* to ***strings/Ability\_Descriptions.string***, as well as the description itself.
4. Add a pointer to your description inside *AbilityDescriptionTable*, found in ***assembly/data/Ability\_Description\_Table.s***. Make sure the position matches the index defined in Step 1!
5. Your new ability is now defined and can be assigned to a pokemon with the Ability ID defined in Step 1. Any ability coding/logic must be written by the user.

## Poke Balls

1. Define a Ball ID – add a new ball type inside the *enum* *BallTypes* via *BALL\_TYPE\_<ball\_name>* inside ***include/new/catching.h***. You can only have up to 32 ball types, due to the restriction on bits.
   1. Update *LAST\_BALL\_INDEX* and *NUM\_BALLS* as needed.
2. Throwing Image – add the indexed 16x48 .png file to ***graphics/Poke\_Balls/***, and any new particles to ***graphics/Poke\_Balls/Opening\_Particles/***
   1. Note that the bag image is excluded because item data is untouched in this engine
3. Open ***src/Tables/Ball\_Graphics\_Tables.c***.
   1. Declare the throwing image and palette with *extern const u8 gInterfaceGfx\_<ball\_name>Tiles[];* and *extern const u8 gInterfaceGfx\_<ball\_name>Pal[];*
   2. Any new opening particles can be declared underneath the image/pal declarations, eg. *extern const u8 gBattleAnimSpriteSheet\_Particles<your\_particle>Tiles/Pal[];* You can define the desired starting frame (for particle animations) similar to *BALL\_OPEN\_WHITE\_STARS*
   3. Ball Tag – add *GFX\_TAG\_<ball\_name>* at the end of *enum BallTags*
   4. Ball Particle Tags – add *TAG\_BALL\_OPEN\_<ball\_name>* at the end of *enum BallOpenParticleTags*
   5. Add the sprite sheet info to *gBallSpriteSheets* following the same format as the other entries
   6. Add the sprite palette info to *gBallSpritePalettes*, again following the same table format.
   7. Add the sprite template info to *gBallSpriteTemplates*, emulating another table entry. The *tileTag* and *paletteTag* should be the only elements that change.
   8. Add an entry to the particle sprite sheet, *gBallOpenParticleSpritesheets*, following the format of the other entries. *gBattleAnimSpriteSheet\_Particles* is used for any existing opening particles (eg. the sticks/stars defined in *enum BallOpenParticles*). If you’ve included custom opening particles, assign them similar to the dusk ball or heal ball.
   9. Add an entry for the particle palette table, *gBallOpenParticlePalettes*, again following the table format. *gBattleAnimSpritePalette\_136* is for the given opening particles (sticks, stars, etc). Custom particle palettes can be defined similar to the dusk or heal ball.
   10. Add an entry to *gBallOpenParticleAnimNums*, linking your ball ID to the opening particle. It is easiest to assign existing particles, eg. *BALL\_OPEN\_STICKS*. Custom particles are defined similar to the dusk or heal ball, eg. follow *BALL\_OPEN\_PURPLE\_CIRCLES* and its definitions/uses.
   11. Create a particle trajectory animation entry in *gBallOpenParticleAnimationFuncs*. The existing animation trajectories are hard to describe, just test them yourself 😊
   12. Add an entry to the *gBallOpenMonFadePal* table, which is the color the pokemon fades to when it exits a pokeball. You can play around with RGB values in paint to find one you like. Keep the definitions after BALL\_TYPE\_DREAM\_BALL at the end.
   13. Create a sprite template entry in *gBallParticleSpriteTemplates* following the structure of one of the new balls such as Dusk Ball or Dream Ball. Again, your tileTag and paletteTag should be the only differences between entries.
4. Ball Catch Multiplier
   1. open ***src/catching.c*** and find the function, ***atkEF\_handleballthrow***.
   2. Add a *case* for your new ball inside the *switch (ItemType)* statement to include logic for your specific ball catch multiplier. A *ball\_multiplier* value of 10 corresponds to a 1x catch multiplier, as the value is divided by 10 later on. You can also add or subtract values by modifying the variable *catch\_rate¸*as in the heavy ball’s case.
5. You have now fully defined your new pokeball. You can assign it to specific trainer’s pokemon (eg. Battle Frontier). You will still need to create the item data. This is entirely up to the user, as this engine does not directly modify item data.

## Items

NOTE: Defining a new battle item is easily done with this engine. The logic for the hold item effect is not generic, and must be done by the user.

1. Define an Item ID – add a define: *#define ITEM\_<your\_item\_name>* inside ***include/constants/items.h***.This is your item’s index. Update *ITEMS\_COUNT*
2. Define a hold item effect index - *#define ITEM\_EFFECT\_<your\_effect\_name>* in ***include/constants/hold\_effects.h*.** Update*ITEM\_EFFECT\_COUNT* as needed.
3. Create your new item – this is handled entirely by the user, as item data is untouched in this engine. Just make sure the item index (step 1) matches. Assign the item the hold effect defined in Step 2.
4. There is no hold effect information needed. The user must find an appropriate location to include logic for their new battle item effect. The function ***ItemId\_GetHoldEffect*** is an effective way to obtain the pokemon in question’s hold item effect from its hold item ID.

# Code Files

Below is a list of all code files which can be found in **src**, along with features contained in each file. Each file’s respective header file can be found in **include/new**.

|  |  |
| --- | --- |
| File | Description |
| **ability\_battle\_effects.c** | Functions that introduce or modify battle effects via abilities or otherwise. Includes terrain effects and ability pop-up |
| **accuracy\_calc.c** | Rewrites how accuracy is calculated, including all relevant abilities, effects, etc. Includes protection logic |
| **attackcanceler.c** | Handles any logic for discerning if a move can be used or is effective, eg. mold breaker, flinch status, truant, etc. |
| **battle\_anims.c** | Functions and structures to modify attack animations. |
| **battle\_contoller\_opponent.c** | Handles the functions responsible for the user moving between battle menus, choosing moves, etc. |
| **battle\_script\_util.c** | General functions that aide in battle scripting via callasm. |
| **battle\_start\_turn\_start.c** | Handles the logic for determining which pokemon attacks first. Also handles setting up battle data. |
| **battle\_strings.c** | Modifies the strings displayed in battle. |
| **battle\_terrain.c** | Functions responsible for checking/loading/removing battle terrain. |
| **battle\_transition.c** | Handles the transition into battle, eg. trainer mugshots. |
| **battle\_util.c** | General functions for aiding in battle logic for everything. |
| **build\_pokemon.c** | Modifies the data that is set for generated pokemon, eg. for battle tower/frontier team generation and others. |
| **catching.c** | Handles the catch probability logic, expands pokeballs, etc. |
| **character\_customization.c** | Functions for altering the player's sprite based on the current sprite/palette selections |
| **cmd49.c** | Handles a ton of battle logic at the end of each turn |
| **damage\_calc.c** | Functions responsible for calculating damage, including modifications from abilities, effects, etc... |
| **daycare.c** | Functions that handle all daycare functions, including attribute inheritance and step counts. |
| **dexnav.c** | Functions for the simplified dexnav system. |
| **dns.c** | Handles functions and palette changes for the day, night, and seasons feature. |
| **dynamic\_ow\_pals.c** | Handles the dynamic loading and tracking of overworld sprite palettes. |
| **end\_battle.c** | Handles all battle termination logic and data resetting/saving. |
| **end\_turn.c** | Handles all effects that happen at the end of each turn. |
| **evolution.c** | Handles old and new evolution methods. |
| **exp.c** | Functions that handle the exp share and exp gain. |
| **form\_change.c** | Functions/structures/arrays that handle species that change form. |
| **frontier.c** | All supporting and master functions for developing a battle frontier. |
| **frontier\_records.c** | Handle player's status in the battle frontier. |
| **general\_bs\_commands.c** | Functions that support the battle scripting in assembly/battle\_scripts. |
| **item.c** | Handles all item related functions, such as returning hold effects, tm/hm expansion, etc. |
| **item\_battle\_effects.c** | Handles functions that deal with battle effects of specific items. |
| **learn\_move.c** | Handles functions for pokemon trying to learn moves. |
| **link.c** | Handles data transfer. |
| **mega.c** | Functions that support mega evolution logic and execution. |
| **move\_menu.c** | Functions for displaying move data and z moves in the battle menu. |
| **multi.c** | Handles partner battle logic. |
| **new\_bs\_commands.c** | Functions for any additional battle scripting commands that are used inside battle scripts. |
| **options\_menu.c** | Functions for secondary options menu. |
| **overworld.c** | Functions for anything regarding the overworld, such as trainer spotting, whiteout, step counters, etc. |
| **party\_menu.c** | Handles anything related to the party menu, such as field moves, new party menu GUIs, etc... |
| **pokemon\_storage\_system.c** | Handles pokemon storage expansion and related functions |
| **read\_keys.c** | Emulated JPANs keypad hacks, allowing the designer to force key presses, prevent them, or map functions onto them, among other uses. |
| **roamer.c** | Handles roaming pokemon and relevant functions/hooks. |
| **save.c** | Handles save block expansion functions/structures. |
| **scripting.c** | Handles all scripting specials or other functions associated with scripts. |
| **set\_effect.c** | Handles move effects. |
| **set\_z\_effect.c** | Handles the logic for z move effects, including special z moves. |
| **start\_menu.c** | Functions to redo how the start menu is generated, and associated functions such as safari steps/ball count. |
| **stat\_buffs.c** | Adjusts stat-related functions to include abilities and effects that change their operations (eg. contrary). |
| **switching.c** | Handles battle switching logic. |
| **trainer\_sliding.c** | Handles mid-battle trainer sliding and related message. |
| **util.c** | General utility functions. |
| **wild\_encounter.c** | Handles functions related to wild encounter probability and associated features. |

The following files can be found in **src/Battle\_AI** are contain code for the updated battle AI.

|  |  |
| --- | --- |
| File | Description |
| **ai\_advanced.c** | Advanced logic for the AI, including move prediction and fight classes |
| **ai\_master.c** | The master function(s) for the AI logic |
| **ai\_negatives.c** | All possible subtractions to an AIs move viability. |
| **ai\_partner.c** | Partner AI logic function(s) |
| **ai\_positives.c** | All possible additions to an AIs move viability |
| **ai\_util.c** | Commonly used functions in AI logic |

The following files can be found in **src/Tables** and contain various data tables for the user to modify as they see fit.

|  |  |
| --- | --- |
| File | Description |
| **Attack\_Data\_Table.c** | Contains data for all the different battle moves. |
| **Ball\_Graphics\_Tables.c** | Contains data for new in-battle Poke Ball graphics. |
| **Class\_Based\_Poke\_Ball\_Table.c** | Used in conjunction with the definition TRAINER\_CLASS\_POKE\_BALLS to assign unique Poké Balls toe each trainer class. |
| **Experience\_Tables.c** | Contains larger Exp yield data for each species (see GEN\_7\_BASE\_EXP\_YIELD). Also contains experience per level tables to help account for a possible increased max level. |
| **Frontier\_Trainers.c** | Contains [data](#_Battle_Frontier) for various trainers which can appear in *Battle Facilities*. |
| **Learnsets.c** | Contains data for the level-up movepool of each Pokémon to support the increased number of moves the engine has to offer. |
| **Music\_Tables.c** | Contains [data](#_Battle_Music) to set trainer encounter music by class (see ENCOUNTER\_MUSIC\_BY\_CLASS), battle music by class, and battle music by wild species. |
| **Pickup\_Items.c** | Contains the different items that can be picked up using the ability [Pickup](#_Pickup). |
| **Pokemon\_Tables.c** | Contains data tables for alternate species height and weights (for example, if a mega form is a different size), and several Pokémon ban lists. |
| **Terrain\_Table.c** | Contains [data](#_Battle_Terrain) for each battle background and how it influences various effects in battle such as *Camouflage*, *Nature Power*, *Secret Power*, and *Burmy*. |
| **Wild\_Encounter\_Tables.c** | Contains data for setting up [time-based wild encounters](#_Time_of_Day) and [swarms](#_Swarms). |

The following files can also be found in **src/Tables** and also contain various data tables for the user to modify, however, as they are header files, a change must be made in the file that includes them in order for them to recompile without cleaning the whole engine.

|  |  |
| --- | --- |
| File | Description |
| **battle\_tower\_spreads.h** | Set up EVs, IVs, ability types, items, moves, ball type, etc, for trainers in the battle tower. |
| **frontier\_multi\_spreads.h** | Set up battle frontier multi battle spreads: EVs, IVs, ability type, item, moves, pokeball type, and more! |
| **frontier\_special\_trainer\_spreads.h** | Set up battle frontier species for special trainers, including EVs, IVs, nature, ability, ball type, etc! |
| **frontier\_trainer\_names.h** | Set up tables of names for battle frontier trainers |
| **trainers\_with\_evs\_table.h** | Defining the EV/IV/ability/ball type to be assigned to trainer pokemon |

The following files can be found in **assembly/data** and contain various data tables for the user to modify as they see fit.

|  |  |
| --- | --- |
| File | Description |
| **Ability\_Description\_Table.s** | Contains a table with pointers to the different ability descriptions. |
| **Ability\_Tables.s** | Contains several tables with abilities mainly used by the battle engine to see if certain abilities are in a given list. |
| **Anim\_Background\_Table.s** | Contains a table of all the in-battle animation backgrounds. |
| **Anim\_Backgrounds\_Graphics\_Defines.s** | Contains defines for  **assembly/data/Anim\_Background\_Table.s** |
| **Anim\_Particle\_Graphics\_Defines.s** | Contains defines for  **assembly/data/Anim\_Particle\_Table.s** |
| **Anim\_Particle\_Table.s** | Contains a table of all the in-battle attack particles. |
| **Attack\_Anim\_Table.s** | Contains a table and data for all the new attack animations. |
| **Attack\_Description\_Table.s** | Contains a table with pointers to the attack descriptions. |
| **Battle\_Script\_Commands\_Table.s** | Contains both the original battle scripting command table, as well as a second battle scripting command table. |
| **Item\_Tables.s** | Tables with items that are checked during the execution of certain moves. |
| **Move\_Effect\_Table.s** | Contains a table with the battle scripts for each move effect, as well as certain tables of move effects which are used by the AI. |
| **Move\_Tables.s** | Contains several tables with moves mainly used by the battle engine to see if certain moves are in a given list. |
| **Playanimation\_Table.s** | Contains a table and data for special in-battle animations. |
| **Trainer\_Backsprite\_Table.s** | Contains various tables relating the trainer backsprites shown at the start of battle. |
| **Type\_Tables.s** | Contains tables for move type effectives and graphics data for the different type icons. |
| **Z-Move\_Name\_Table.s** | Contains a table with pointers to the names of all the different Z-Moves. |

The following files can be found in **assembly/battle\_scripts** and contain several different scripts that run in-battle.

|  |  |
| --- | --- |
| File | Description |
| **ability\_battle\_scripts.s** | Contains battle scripts for various ability effects. |
| **attackcanceler\_battle\_scripts.s** | Contains battle scripts for the battle scripting command *attackcanceler*.  See **src/attackcanceler.c**. |
| **battle\_start\_turn\_start\_battle\_scripts.s** | Contains battle scripts that can run at the beginning of the battle or each turn.  See **src/battle\_start\_turn\_start.c**. |
| **cmd49\_battle\_scripts.s** | Contains battle scripts for the battle scripting command *cmd49* (*moveend*).  See **src/CMD49.c.** |
| **et\_battle\_scripts.s** | Contains battle scripts that can run at the end of each round.  See **src/end\_turn.c**. |
| **fainting\_battle\_scripts.s** | Contains the battle scripts that run when a Pokémon faints. |
| **general\_attack\_battle\_scripts.s** | Contains various battle scripts for the different move effects. |
| **item\_battle\_scripts.s** | Contains various battle scripts for item effects.  See **src/item\_battle\_effects.c**. |
| **mega\_battle\_script.s** | Contains battle scripts for Mega Evolution, Primal Reversion, and Ultra Burst.  See **src/mega.c**. |
| **move\_menu\_battle\_scripts.s** | Contains battle scripts which print strings for the player while they are selecting moves.  See **src/move\_menu.c**. |
| **set\_effect\_battle\_scripts.s** | Contains battle scripts for various secondary move effects.  See **src/set\_effect.c**. |
| **standard\_damage\_battle\_script.s** | Contains battle scripts that are called for nearly every damaging move and moves that fail. |
| **switch\_battle\_scripts.s** | Contains battle scripts that run when Pokémon are switched out.  See **src/switching.c**. |
| **trainer\_sliding\_battle\_scripts.s** | Contains battle scripts for when the opposing trainer sliding back onto the screen to give a message.  See **src/trainer\_sliding.c**. |
| **z\_effect\_battle\_scripts.s** | Contains battle scripts for various secondary Z-Status move effects.  See **src/set\_z\_effect.c**. |

The following files can be found in **assembly/overworld\_scripts** and contain a few different scripts that run in the overworld.

|  |  |
| --- | --- |
| File | Description |
| **system\_scripts.s** | Contains scripts that are run by the engine rather than as an event script (such as overworld poison, repels, etc.). |
| **trainer\_battle\_scripts.s** | Contains scripts that help initiate new types of trainer battles. |

The following files can be found in **assembly/hooks** and contain different sets of hooks for updated functions.

|  |  |
| --- | --- |
| File | Description |
| **bag\_expansion\_hooks.s** | Contains hooks relating to expanding the bag size.  See **src/item.c**. |
| **follow\_me\_hooks.s** | UNUSED |
| **general\_hooks.s** | Contains various general hooks that didn’t fit into one specific category. |
| **illusion\_hooks.s** | Contains hooks for the ability [Illusion](https://bulbapedia.bulbagarden.net/wiki/Illusion_(Ability)).  See **src/ability\_battle\_effects.c**. |
| **mega\_hooks.s** | Contains hooks for Mega Evolution, Primal Reversion, and Ultra Burst.  See **src/mega.c**. |
| **multi\_hooks.s** | Contains hooks that help implement *multi battles* against two opponents.  See **src/multi.c** and  **src/battle\_controller\_opponent.c**. |
| **multi\_partner\_hooks.s** | Contains hooks that help implement *multi battles* with a partner  See **src/multi.c**. |
| **pokemon\_data\_hooks.s** | Contains hooks for the alteration of various Pokémon data.  See **src/build\_pokemon.c**. |

The following files can be found in **assembly** and contain different assembly routines.

|  |  |
| --- | --- |
| File | Description |
| **main.s** | Similar to **assembly/hooks/general\_hooks.s**, this file was mainly used by **ghouslash**, while the former was mainly used by **Skeli**. |
| **pokedex\_screen\_stats.s** | Contains a routine the replace the Pokédex size comparison with a screen showing the stat values of the Pokémon. |
| **rtc.s** | Contains a routine originally created by [prime-dialga](https://www.pokecommunity.com/member.php?u=188013) that implements the real-time clock using whatever hardware the emulator is running on. |
| **thumb\_compiler\_helper.S** | Contains various functions to helper the C compiler. |

# Engine Scripts

## Clean.py

Clean.py’s role is to clean up the repository. The following commands can be used from the command line:

|  |  |
| --- | --- |
| Command | Description |
| **python scripts/clean.py** | Removes all object files (not including those from images), generated repoints, generated offsets, and generated roms. |
| **python scripts/clean.py all** | Removes all object files, generated repoints, generated offsets, and generated roms. |
| **python scripts/clean.py build** | Removes all object files (not including those from images), generated offsets, and generated roms. |
| **python scripts/clean.py graphics** | Removes all object and header files from images. |
| **python scripts/clean.py file FILE\_PATH** | Removes only the object file for the given file path. Make sure "/" and not "\" is used to separate the directory names. |

## String.py

String.py’s role is to compile *.string* files (found in the **strings** directory). The following rules apply to compiling *.string* files:

* Placing *MAX\_LENGTH=XX* at the top of the file where *XX* is a number of your choosing will force all strings compiled in that file to have a maximum length of *XX* (not including the terminator character, 0xFF).
* When used in conjunction with *MAX\_LENGTH*, placing *FILL\_FF=True* at the top of the file will force all strings shorter than *MAX\_LENGTH* to have FFs appended onto the end to make them the correct size. This is how arrays of strings are made possible (such as *gMoveNames* and *gAbilityNames*).
* Each string name starts with the directive *#org @* and is followed by the title of the string. Several *#org*’s made be piled on top of one another to make several string defines point to the same string. For example:



* The line after the *#org* contains the string. It can be written in plain text with no quotation marks on either side and with no terminator character at the end. The string may be spread onto multiple lines if you so wish. The string is terminated when the next *#org* is parsed.
* There are different escape characters which can be used with the string files:

|  |  |
| --- | --- |
| Character | Becomes |
| \n | Newline |
| \p | New textbox (display arrow) |
| \l | Scroll line |
| \e | é |
| \$ | ₽ |
| \" | ” |

* Text buffers can be used too:

|  |  |
| --- | --- |
| Buffer | Description |
| [.] | … |
| [BUFFER] | Start a buffer. Usually followed by a hex buffer. |
| [ATTACKER] | IN-BATTLE: Loads the attacker’s name (found in *gBankAttacker*). |
| [TARGET] | IN-BATTLE: Loads the target’s name (found in *gBankTarget*). |
| [EFFECT\_BANK] | IN-BATTLE: Loads the bank found in *gEffectBank*. |
| [SCRIPTING\_BANK] | IN-BATTLE: Loads the bank found in *gBattleScripting->bank*. |
| [CURRENT\_MOVE] | IN-BATTLE: Loads the move found in *gCurrentMove*. |
| [LAST\_ITEM] | IN-BATTLE: Loads the item found in *gLastUsedItem*. |
| [LAST\_ABILITY] | IN-BATTLE: Loads the ability found in *gLastUsedAbility*. |
| [ATTACKER\_ABILITY] | IN-BATTLE: Loads the attacker’s ability (from *gBankAttacker*). |
| [TARGET\_ABILITY] | IN-BATTLE: Loads the target’s ability (from *gBankTarget*). |
| [SCRIPTING\_BANK\_ABILITY] | IN-BATTLE: Loads the ability of *gBattleScripting->bank*. |
| [PLAYER\_NAME] | IN-BATTLE: Loads the player’s name. |
| [BLACK] | OVERWORLD: Changes the text colour to black. |
| [RED] | OVERWORLD: Changes the text colour to red. |
| [GREEN] | OVERWORLD: Changes the text colour to green. |
| [BLUE] | OVERWORLD: Changes the text colour to blue. |
| [XX]  Eg. [FA] or [52] | Any two hex characters to represent that byte exactly (hex buffer). Can be used in conjunction with [BUFFER]. For example, in the overworld, [BUFFER][02] is the same as saying [buffer1] in XSE. |

# Credits

**Main Contributors:**

Skeli

Ghoulslash

**Code**:

*Lixdel* - Attack Animations

*Pret* - PokeRuby, PokeFireRed, PokeEmerald

*Sagiri* - Trainer Class Poke Balls, Pickup Update, Move Item, Summary Screen Wrapping

*DizzyEgg* - Emerald Battle Engine Upgrade V1 & V2, Dizzy's Emerald Hacked Engine

*FBI* - Expanded Saveblock, Dexnav

*Touched –* Follow Me, Mega Evolution source

*Navenatox* – Dynamic Overworld Palettes

*Doesntknowhowtoplay & Squeetz* - Pokedex Screen Stats

*Azurile13* - Hidden Abilities

*Squeetz* - Footstep Noises

*JPAN –* Fire Red Hacked Engine Source

*Diegoisawesome –* Triple Layer Tiles

*Jiangzhengwenjz –* Linux compatibility

**Graphics**:

*Golche* - Attack Particles, Battle Backgrounds, Other Graphics

Criminon - Ultra Burst indicator

*Bela* - Poke Balls

*Solo993* - Backsprites

*canstockphoto.ca* - Battle Backgrounds

**Testers:**

Criminon

Dionen

Gail

Recko Juice

Patrickz

If you actually read all the pages up until this point you deserve a medal. No, wait, a cookie. You deserve a cookie. Good job!



THE END

1. Credits to [Squeetz](http://www.pokemonclover.com/) for the original routine. [↑](#footnote-ref-1)
2. Credits to the [Emerald Battle Engine Upgrade V2.0](https://github.com/DizzyEggg/pokeemerald/tree/battle_engine_v2) for the original source code. [↑](#footnote-ref-2)
3. Credits to [Sagiri](https://github.com/Sagiri/bpre-field-pickup) for the original code. [↑](#footnote-ref-3)
4. Credits to [FBI](https://www.pokecommunity.com/showthread.php?t=417493) for this feature. [↑](#footnote-ref-4)
5. Credits to [FBI](https://github.com/EternalCode/Dexnav) and the [Community Hack](https://github.com/Touched/community-hack) for this feature. [↑](#footnote-ref-5)
6. Credits to [Navenatox](https://github.com/Navenatox/DynamicOverworldPalettes) for this feature. [↑](#footnote-ref-6)
7. Credits to [DizzyEgg](https://github.com/DizzyEggg/pokeemerald/tree/battle_engine_v2) for this feature. [↑](#footnote-ref-7)
8. Credits to [azurile13](https://www.pokecommunity.com/showthread.php?t=355328) for this feature. [↑](#footnote-ref-8)
9. Credits to [DoesntKnowHowToPlay](https://www.pokecommunity.com/member.php?u=300067) and [Squeetz](https://www.pokecommunity.com/member.php?u=400825) for this feature.

   10Credits to [Diegoisawesome](https://www.pokecommunity.com/member.php?u=70562) for this feature [↑](#footnote-ref-9)